School of Electrical Engineering, Computing and
Mathematical Sciences

Efficient Semantic Segmentation for Resource-Constrained Applications with Lightweight Neural Networks

Tanmay Singha
0000-0001-6924-057X

This thesis is presented for the Degree of
Doctor of Philosophy
of
Curtin University

July 2023

Declaration

To the best of my knowledge and belief this thesis contains no material previously published by any other person except where due acknowledgment has been made. This thesis contains no material which has been accepted for the award of any other degree or diploma in any university.

Tanmay Singha
School of Electrical, Computing,
and Mathematical Sciences
Curtin University

Signature:.....

Date: 14-July-2023

Abstract

Semantic segmentation plays a crucial role in computer vision by providing comprehensive scene understanding through the precise labeling of each pixel in an image. While deep convolutional neural networks (DCNNs) have demonstrated remarkable performance in tasks like image classification and object detection, they often lack contextual details and background understanding, making semantic segmentation essential. However, existing semantic segmentation models are typically unsuitable for resource-constrained mobile devices due to their large architecture. This creates a demand for efficient real-time segmentation models in applications such as autonomous cars, robotics, medical imaging, agriculture, and surveillance. To address the challenge of achieving high accuracy while maintaining efficiency in lightweight semantic segmentation, this thesis introduces a series of novel model design techniques that tackle issues including scalability, parameter reduction, semantic gap reduction, object shape capture, boundary preservation, gradient preservation, and memory footprint optimization. These advancements pave the way for practical and resource-efficient semantic segmentation models, enabling their deployment on resource-constrained devices.

Firstly, to address scalability and large size issues, this thesis employs a compound scaling technique for designing an end-to-end segmentation model. The literature demonstrates that the performance of DCNNs is highly influenced by the model's width, depth, and input resolution. Therefore, instead of solely scaling the model's depth, the compound scaling technique uniformly scales all dimensions to achieve better semantic results. In addition, a family of efficient

segmentation models called ESPNets is introduced. The thesis also highlights the benefits of incorporating a feature pooling module on top of the encoder. While the proposed ESPNet achieves competitive accuracy, it exhibits slightly lower efficiency in real-time environments due to its 7.6 million parameters. To enhance efficiency, the thesis designs a lightweight backbone with fewer than 1 million parameters. The resulting segmentation model, named FANet, not only improves model accuracy compared to ESPNet but also significantly enhances efficiency.

Secondly, to address the challenge of large semantic gaps and capture objects with varied geometrical shapes in complex scenes, this thesis introduces several techniques. The proposed FANet model modifies the bi-directional feature pyramid network (Bi-FPN) to reduce semantic gaps and enhance the feature hierarchy. Building upon this, a novel model named M2FANet is introduced with an optimized feature fusion module (FFM) consisting of skip connections. Additionally, a feature scaling module (FSM) is deployed at the decoder side to improve receptive fields for capturing objects of different sizes. However, the existing FSM contributes a large number of parameters and FLOPs, resulting in moderately higher memory usage. To address this issue, the thesis introduces optimized scaling techniques, including a new FSM design and Feature Refinement (FR) at the decoder end. Furthermore, a new bottleneck block called short-term dense bottleneck (SDB) is introduced to enhance the encoder's capturing ability, providing a significantly larger field-of-view compared to the existing bottleneck blocks. Segmentation models incorporating SDB blocks demonstrate exceptional performance across various datasets while maintaining high efficiency.

Thirdly, to tackle the issue of boundary degeneration effect, the thesis introduces a novel knowledge-sharing technique among multiple branches of the encoder. A shared two-branch encoder design is presented, where both deep and shallow branches contribute to learning at different stages. Additionally, a context mining module is introduced for coarse-to-fine refinement of the shared fea-

ture map. Leveraging this innovative knowledge sharing architecture, an efficient structural crack detection technique named SC-CrackSeg is developed specifically for real-time applications. This technique outperforms existing models and achieves an impressive frame rate, leading to superior results.

Finally, to tackle the issue of gradient vanishing in the encoder, this thesis introduces a novel design called the shared-branch multiple sub-encoders design. This design filters the shared semantic feature maps through multiple sub-encoders. The top global feature map of each sub-encoder is passed to the next sub-encoder, enabling coarse-to-fine refinement, while the lateral connections at the same stages in multiple sub-encoders preserve the gradients. Moreover, for accurate object localization and improved semantic representation, a hybrid path attention mechanism is introduced at the decoder side. Building upon these innovations, an efficient model called SFRSeg is presented, which achieves state-of-the-art performance on various indoor and outdoor datasets in both structured and unstructured environments.

The novel techniques developed in this thesis have made significant contributions to the research progress on lightweight semantic segmentation. These advancements enable the development and deployment of practical models for resource-constrained applications. These models exhibit lower memory footprints and lower power consumption while achieving competitive segmentation performance.

Acknowledgements

I would like to take this opportunity to express my heartfelt gratitude to all the individuals and organizations who have provided invaluable support throughout my journey of pursuing a Doctor of Philosophy degree.

Firstly, I would like to express my deepest gratitude to my parents and siblings for their support, motivation, and blessings. Their guidance has always steered me in the right direction and played a significant role in shaping who I am today. I cannot find the right words to fully describe how much they have helped me in shaping a meaningful and satisfying life. I am incredibly grateful for their contributions.

I owe my sincere gratitude to both of my research supervisors, Dr. Aneesh Krishna, Associate Professor,Curtin University and Dr. Duc-Son Pham, Senior Lecturer, Curtin University. Working under your supervision has been an incredible privilege and has had a profound impact on both my personal and professional growth. Your expertise, dedication, and commitment to excellence have continually inspired me to push the boundaries of my capabilities and strive for academic excellence. Your insightful feedback and constructive criticism have helped me refine my ideas and approach, enabling me to navigate the challenges and complexities of the research process. Your ability to provide clear direction while also fostering independent thinking has empowered me to explore new avenues and think critically about my work. To build my research network, you have given me the opportunity to work on multiple summer internship projects. These experiences have not only expanded my research domains but also enhanced my

collaborative research skills.You have given me the opportunity to co-supervise honors students in their research projects, which has greatly improved my supervision skills. Beside your role as a supervisor, I am grateful for the genuine interest you have shown in my overall well-being. Your door has always been open for discussions, whether they were related to my research progress or personal challenges. Your empathetic and supportive approach has made me feel valued as a person, not just as a student, and I am truly appreciative of that.

The School of Electrical Engineering, Computing and Mathematical Sciences at Curtin University has awarded me a tuition fee waiver scholarship, which has greatly supported my research work. I am immensely grateful to the school for their trust in my abilities and their provision of various forms of support.

I am also thankful to the Graduate Research School, Curtin University for giving me six months Higher Degree of Research (HDR) scholarship.

I would like to thank the Pawsey super-computing centre for giving me hardware support to run some of my experiments in their system.

I am thankful to all my co-authors who have made valuable contributions to the research work. Their input has substantially improved the experimental results and the quality of the papers.

I would like to express my gratitude to all my AI team members for generously sharing their research works. Our collaborative research platform has been truly exceptional in enhancing our research skills. This platform has provided us with numerous opportunities to work together on multiple projects.

I am also deeply grateful to all the academic and administrative staff members of my school. They have consistently demonstrated professionalism and provided unwavering support. Their efforts have created a conducive research and learning environment that greatly enhances our research abilities.

I would like to thank Dr. Sreenithya Sumesh and her family. Dr. Sreenithya was my first PhD office mate. From the first day, she extended her support towards me by sharing her journey in PhD. Whenever, I went through bad patches

of my research journey, she was there like a mentor. I am deeply grateful for all kind of supports she has given to me.

Last but not least, I would like to thank the almighty God for pouring His blessings upon me and my family. Due to His blessings, I have had the privilege of meeting all these wonderful people and receiving their valuable support.

List of publications included as part of the thesis

The following list includes the publications which form part of this thesis:

Publication 1:

Singha, T., Pham, DS., Krishna, A., Dunstan, J. (2020). **"Efficient Segmentation Pyramid Network"**. In: ICONIP 2020. vol 1332. Springer, Cham. https://doi.org/10.1007/978-3-030-63820-7-44.

Publication 2:

Singha, T., Pham, DS., Krishna, A., **"FANet: Feature Aggregation Network for Semantic Segmentation,"** In: Proc. DICTA, Melbourne, Australia, 2020, pp. 1-8, doi: 10.1109/DICTA51227.2020.9363370.

Publication 3:

Singha, T., Pham, DS., Krishna, A., Gedeon, T. (2021). **"A Lightweight Multi-scale Feature Fusion Network for Real-Time Semantic Segmentation"**. In: Proc. ICONIP 2021. vol 13109. Springer, Cham. https://doi.org/10.1007/978-3-030-92270-2-17.

Publication 4:

Singha, T., Bergemann, M., Pham, DS., Krishna, A., **"SCMNet: Shared Context Mining Network for Real-time Semantic Segmentation,"** In: Proc. DICTA, Gold Coast, Australia, 2021, pp. 1-8, doi: 10.1109/DICTA52665.2021.9647401.

Publication 5:

Singha, T., Pham, DS., Krishna, A.. **"Urban Street Scene Analysis Us-**

ing Lightweight Multi-level Multi-path Feature Aggregation Network". Multiagent and Grid Systems, vol. 17, no. 3, pp. 249-271, 2021. DOI: 10.3233/MGS-210353

Publication 6:

Singha, T., Bergemann, M., Pham, DS., Krishna, A., **"SC-CrackSeg: A Real-time Shared Feature Pyramid Network for Crack Detection and Segmentation,"** In: Proc. DICTA, Sydney, Australia, 2022, pp. 1-8, doi: 10.1109/DICTA56598.2022.10034629.

Publication 7:

Singha, T., Pham, DS., Krishna, A., **"SDBNet: Lightweight Real-time Semantic Segmentation Using Short-term Dense Bottleneck,"** In: Proc. DICTA, Sydney, Australia, 2022, pp. 1-8, doi: 10.1109/DICTA56598.2022.10034634.

Publication 8:

Singha, T., Pham, DS., Krishna, A., **"A real-time semantic segmentation model using iteratively shared features in multiple sub-encoders"**, Pattern Recognition, Volume 140, 2023, 109557, ISSN 0031-3203, https://doi.org/10.1016/j.patcog.2023.109557.

Publication 9:

Singha, T., Pham, DS., Krishna, A., **"Multi-encoder Context Aggregation Network for Structured and Unstructured Urban Street Scene Analysis,"** in IEEE Access, vol. 11, pp. 66227-66244, 2023, doi: 10.1109/ACCESS.2023.3289968.

Contents

List of Figures

Chapter 1

Introduction

In the modern era, digital scene understanding is an important research topic in the field of computer vision. To analyze and comprehend a visual scene, it requires interpreting and extracting meaningful information from the images or videos for gaining a deeper understanding of the objects, relationships, and context present in the scene. A complete scene understanding typically involves several tasks, including object recognition, object detection, region identification, semantic segmentation, depth estimation, and spatial understanding. Hence, the ultimate goal of automated scene understanding is to make the machine intelligent enough for comprehending and interpreting the visual scenes in a way that is similar to how humans perceive and understand their environment. For achieving this goal, deep Convolutional Neural Networks (CNNs) have played an important role.

This thesis primarily focuses on real-time semantic segmentation models for resource-constrained applications. However, in this section, for the benefit of the readers, a comprehensive discussion about two other computer vision tasks, namely image classification and object detection, is provided. By doing so, readers will gain a clear understanding of the application of deep CNNs in the field of image classification and object detection for scene understanding. They will also become aware of the limitations of these tasks, which further emphasizes the need

for semantic segmentation to achieve better scene understanding.

Hence, this chapter begins by discussing the background, including the architecture and limitations of image classification, object detection, and semantic segmentation tasks. It then provides a detailed explanation of the necessity of semantic segmentation for resource-constrained applications. Subsequently, a thorough literature review, identification of research gaps, formulation of the problem statement, presentation of the proposed research, and an in-depth explanation of the experimental setup are provided.

1.1 Background

1.1.1 Image classification

Image classification is a fundamental task in computer vision for recognizing the object in the scene. By studying the visual features of the object in the scene, machine accurately identifies the object and assigns a label or class to the image. For understanding the visual features by the machine, deep CNNs have shown a revolutionized way of artificially fusing the intelligence into the machine. It has become the state-of-the-art approach for achieving high accuracy in this domain. It extracts meaningful features from images and learn hierarchical representations that capture both low-level and high-level visual patterns.

Purpose

The main objectives of image classification are:

- **Object Recognition:** It enables the identification and recognition of objects or patterns within images. It allows computers to understand and interpret the visual content present in an image, which is essential for scene understanding.
- **Content Organization:** It facilitates the organization and categorization of large collections of images. By automatically assigning labels to images,

they can be grouped, sorted, and indexed based on their content, making it easier to search and retrieve specific images from a database.

- **Visual Search:** It plays a crucial role in visual search engines. By categorizing images into different classes, users can perform searches based on visual similarity. For example, searching for"dogs" would retrieve images that have been classified as dogs, even if they were not explicitly tagged with the word "dog."

- **Autonomous Systems:** It is vital for enabling autonomous systems, such as self-driving cars and drones, to perceive and understand the visual information in their surroundings. By classifying objects and scenes in real-time, these systems can make informed decisions and navigate their environments safely.

- **Medical Diagnosis:** It is extensively used in medical imaging for the detection and diagnosis of diseases. By classifying medical images such as X-rays, and MRIs healthcare professionals can identify abnormalities, tumors, or specific medical conditions, aiding in accurate diagnosis and treatment planning.

Overall, image classification is a fundamental building block in computer vision and has broad applications across various domains. It enables machines to understand and interpret visual information, opening doors to numerous innovative solutions and advancements.

Architecture

DCNNs are mainly used for image classification due to their effectiveness in capturing complex visual features and learning hierarchical representations from images. LeNet-5 LeCun et al. (1998) was one of the pioneering CNN architectures for image classification. It has shown a revolutionary approach of designing DCNN by staging several convolutional and pooling layers followed by fully connected

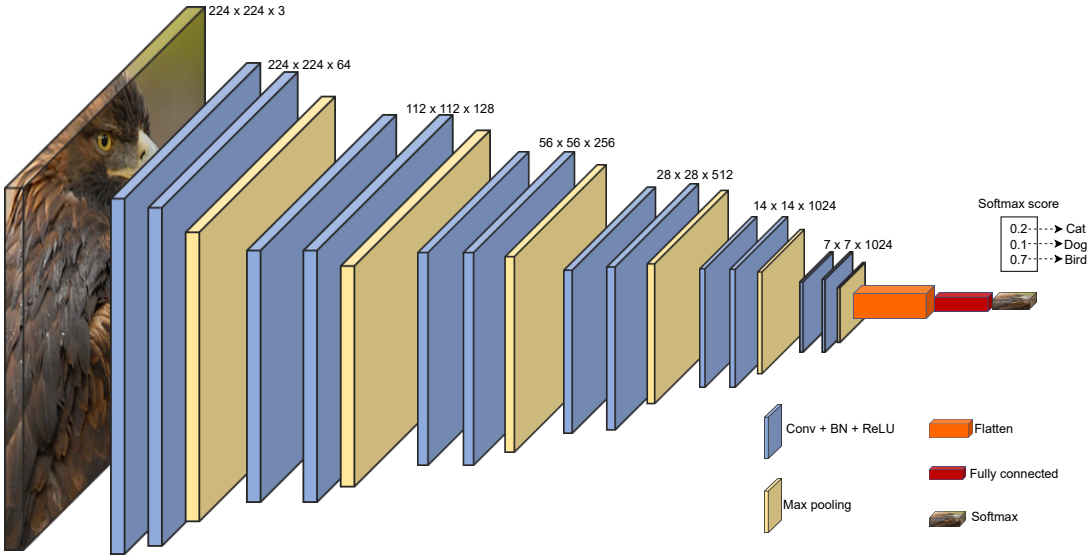

Figure 1.1: An illustration of deep convolutional neural architecture (Pyramidal shape).

layers. Followed by this novel approach, several other models such as VGGNet Simonyan & Zisserman (2014), ResNet Targ et al. (2016), Xception Chollet (2017), and EfficientNet Tan & Le (2019) are introduced which have shown an outstanding performance growth in this domain. The general architecture of deep CNN models is displayed in Figure 1.1. It consists of several convolution layers, max pooling layers which filter the input image and gradually downsamples the spatial dimensions of the image while increasing the number of channels of the feature map. Thus, it creates a rich global feature map with lowest spatial dimensions. The classification head consists of flatten, dense and softmax layer which transforms the global feature into a suitable format and based on the classification score, it assigns a class or label to the image.

Limitation

While image classification is a powerful technique for categorizing individual images into specific classes or labels, it has certain limitations when it comes to scene understanding, which involves a deeper understanding of the context and semantic relationships within a scene. Some of the limitations of image classification

for scene understanding are:

- **Lack of Contextual Information:** Image classification algorithms typically focus on analyzing individual images in isolation, without considering the broader context or relationships with other objects within a scene. This limitation makes it challenging to understand the overall scene and the interactions between different elements.

- **Inability to Capture Spatial Relationships:** Image classification algorithms often treat images as a collection of independent pixels or regions, neglecting the spatial relationships between objects. Understanding scene semantics and structure requires capturing the spatial arrangement, relative positions, and interactions between objects, which image classification alone cannot accomplish.

- **Limited Semantic Understanding:** Image classification is primarily concerned with assigning a single label or class to an image. However, scene understanding requires a more fine-grained and detailed understanding of the different objects, their attributes, and their semantic relationships. Image classification may overlook these nuances and fail to capture the richness of scene semantics.

- **Difficulty in Handling Ambiguity:** Images can contain ambiguous or complex scenes where objects or regions may have multiple interpretations or belong to multiple classes. Image classification algorithms may struggle to handle such ambiguity and may assign incorrect labels or struggle with uncertain cases, leading to reduced scene understanding accuracy.

1.1.2 Object detection

Identifying an object in a scene can be be achieved by image classification. However, positioning the objects in the scene can not be done by image classification. Thus, for localizing a object in a scene, Object detection task is introduced in

computer vision. It aims to locate and classify objects of interest within an image or a video. Unlike image classification that assigns a single label to the entire image, object detection goes a step further by identifying and localizing individual objects within the scene. It provides both the class or category of the object and the precise bounding box coordinates that enclose the object.

Purpose

The purpose of object detection is to enable machines to identify and locate objects of interest within images or videos. Object detection serves as a fundamental building block for many applications and tasks, including:

- **Object localization:** Object detection provides precise information about the location and extent of objects within an image or video. This is crucial for tasks that require accurate object positioning, such as robotic manipulation, augmented reality, and autonomous navigation systems.
- **Object recognition and classification:** Object detection allows for the recognition and classification of objects into different categories or classes. It provides information about the type or identity of objects present in a scene. This is useful in applications like visual search, content-based image retrieval, and inventory management.
- **Scene understanding:** Object detection contributes to a better understanding of visual scenes by identifying and analyzing the objects within them. It helps in extracting meaningful information about the composition, layout, and interactions of objects, leading to higher-level scene understanding and analysis.
- **Video analysis and surveillance:** Object detection techniques play a crucial role in video analysis and surveillance systems. They enable the identification and tracking of objects over time, allowing for applications like video-based security monitoring, activity recognition, and abnormal event detection.

- **Autonomous driving:** Object detection is essential for autonomous driving systems as it enables the detection and tracking of pedestrians, vehicles, and other objects on the road. It provides the necessary information for making real-time decisions and ensuring safe navigation in complex traffic environments.

Overall, the purpose of object detection techniques is to enable machines to perceive and understand the visual world by detecting and localizing objects, facilitating tasks that require object recognition, scene understanding, tracking, and decision-making based on visual information.

Architecture

Object detection techniques typically involve the following steps:

- **Region proposal:** In this step, potential object regions or bounding box proposals are generated based on the presence of objects in different areas of the image. This is done using algorithms such as Selective Search, EdgeBoxes, or Region Proposal Networks (RPNs).

- **Feature extraction:** Convolutional Neural Networks (CNNs) are commonly used to extract meaningful features from the proposed regions. The CNN model is typically pre-trained on a large dataset (e.g., ImageNet Deng et al. (2009)) and can capture hierarchical representations of visual features.

- **Object classification:** The extracted features from each proposed region are fed into a classifier, such as a softmax classifier, to determine the class or category of the object within that region. This step involves assigning a class label to each proposed region, indicating the presence of a specific object category.

- **Bounding box refinement:** The initially proposed bounding boxes are refined to better fit the object's precise location within the region. Techniques

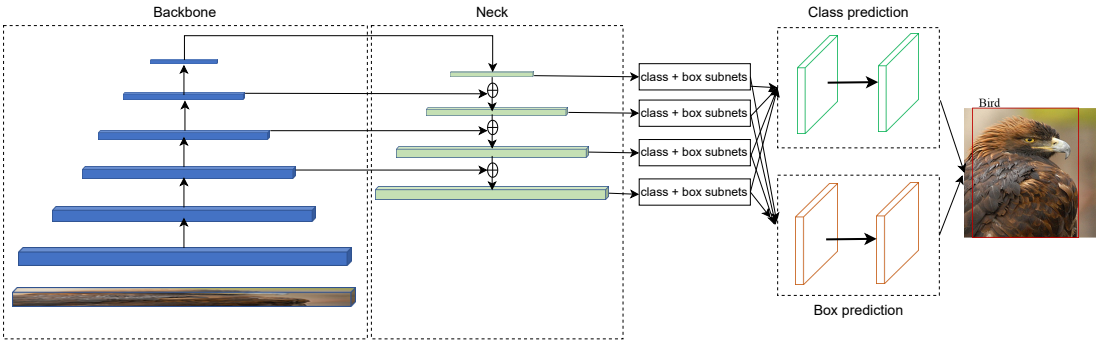

Figure 1.2: An illustration of general architecture of object detection model

such as bounding box regression or non-maximum suppression (NMS) are commonly employed to refine the bounding box coordinates and remove duplicate or overlapping detections.

Over the years, several object detection models such as R-CNN Girshick et al. (2014), Faster R-CNN Ren et al. (2015), YOLO Redmon et al. (2016), EfficientDet Tan et al. (2020) have been proposed. These techniques have enabled a wide range of applications, including autonomous driving, surveillance systems, object tracking, and augmented reality. They provide a more comprehensive understanding of visual scenes by identifying and localizing objects within them.

Limitation

Object detection techniques have made significant advancements in the field of machine learning, but they still have some limitations. Here are a few common limitations of object detection:

- **Accuracy and precision:** Object detection algorithms may have limitations in accurately detecting and localizing objects, especially in complex scenes, cluttered backgrounds, or when objects are occluded or have similar appearances. Achieving high accuracy and precision remains a challenge, particularly for small or partially visible objects.

- **Scale and variability:** Object detection models may struggle to handle objects at different scales and viewpoints. Detecting objects at extreme scales (very small or very large) or dealing with significant variations in object appearance, pose, illumination, or occlusion can pose challenges for detection algorithms.

- **Semantic understanding:** Object detection focuses on identifying and localizing objects in an image but may not capture higher-level semantic understanding. While detecting objects is an important step, it may not provide sufficient context or understanding of the relationships between objects, their interactions, or the overall scene semantics.

- **Limited contextual information:** Object detection algorithms often operate on individual frames or images and may not leverage contextual information from neighboring frames or wider temporal context. Incorporating richer contextual information could improve object detection performance and robustness in complex scenarios.

1.1.3 Semantic segmentation

Even though for better scene understanding, object detection techniques have shown an outstanding performance, but they still fail to provide higher level semantic details such as the relationships between objects, their interactions, or the overall scene semantics. Thus, there is a need for a new computer vision technique which can provides such details.

Semantic segmentation is a computer vision task that involves classifying and labeling each pixel in an image with its corresponding semantic category. In other words, it aims to partition an image into multiple regions or segments by clustering the pixels of having similar features and assign a class label to each segment. As it is a pixel-wise classification task which involves considering the characteristics of neighboring pixels before assigning a class label to each pixel, so

it allows for a comprehensive analysis of the overall semantics of a scene, leading to a better understanding of complex scenes.

Purpose

The main purposes of semantic segmentation include:

- **Scene understanding:** Semantic segmentation provides a fine-grained analysis of an image, allowing for a better understanding of the objects, boundaries, and relationships within a scene. This information is crucial for tasks such as scene understanding, object recognition, and contextual understanding.

- **Object localization:** Semantic segmentation can be used to precisely localize objects within an image. By assigning class labels to individual pixels, it enables the delineation of object boundaries, making it easier to detect and locate specific objects of interest.

- **Image parsing:** Semantic segmentation aids in parsing an image into its constituent parts or regions, providing a higher level of understanding beyond simple object detection. It allows for the identification and extraction of specific regions based on their semantic meaning.

- **Augmented reality and virtual reality:** Semantic segmentation contributes to realistic and immersive experiences in augmented reality and virtual reality applications. By segmenting the scene into different objects and regions, it enables realistic virtual object placement and interaction with the real-world environment.

- **Visual perception for autonomous systems:** In applications such as autonomous driving, and robotics, semantic segmentation plays a vital role in the perception pipeline. It helps autonomous systems understand the surrounding environment by segmenting the scene into categories such as

road, pedestrians, vehicles, and obstacles and navigates them through complex scenes.

- **Medical imaging:** Semantic segmentation is extensively used in medical imaging for tasks such as tumor detection, organ segmentation, and anomaly identification. It assists healthcare professionals in accurate diagnosis, treatment planning, and monitoring of diseases.

- **Other applications:** In recent days, semantic segmentation technique is utilized for designing various applications such as plant diseases detection, crack detection on road and building, and urban planning and geo-spatial analysis from aerial or satellite imagery.

Overall, the purpose of semantic segmentation is to provide a detailed understanding of an image at a pixel level, enabling various applications in computer vision, artificial intelligence, and other domains where precise scene understanding is required.

Architecture

Traditional approaches to semantic segmentation involved using handcrafted features and algorithms, such as graph cuts, random forests, or conditional random fields. However, with the advancements in deep learning, convolutional neural networks (CNNs) have become the dominant method for semantic segmentation.

Deep learning-based semantic segmentation models typically employ an encoder-decoder architecture. The encoder network, often based on pre-trained CNNs like VGG Simonyan & Zisserman (2014), ResNet Targ et al. (2016), MobileNet Sandler et al. (2018), or EfficientNet Tan & Le (2019) extracts high-level features from the input image. The decoder network then upsamples these features to generate a dense pixel-wise prediction map, which represents the class labels for each pixel in the input image. Basically, the encoder convolves the input image and creates a pyramidal shape feature hierarchy whereas the decoder deconvolves

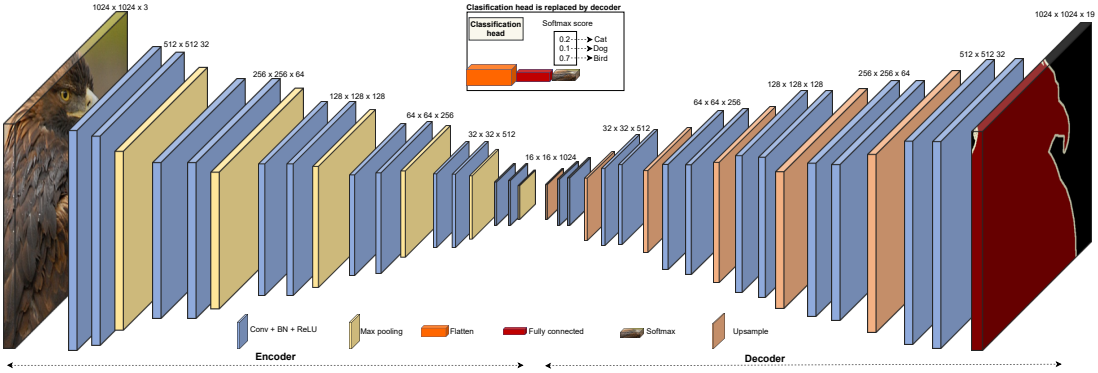

Figure 1.3: An illustration of general architecture of semantic segmentation model

the global feature maps and creates an inverted pyramidal shape feature hierarchy. The layered architecture of the encoder-decoder network is displayed in Figure 1.3. It can be seen that a series of convolution and max pooling layers are used to filter and downsample the input image at the encoder side. The top layers of an image classification model which consists of a flatten, a dense, and a softmax layer, are replaced by a decoder network for generating a dense pixel-wise prediction map in order to achieve semantic segmentation of the input image. This revolutionized approach of replacing the top layers of an existing DCNN is introduced by FCN Long et al. (2015). Later on, several models such as U-Net Ronneberger et al. (2015), SegNet Badrinarayanan et al. (2017), PSPNet Zhao et al. (2017) and DeepLab Chen et al. (2017) were introduced following the same novel approach. These models have shown an performance in this domain.

Limitation and challenges

While semantic segmentation provides better semantic details of a complex scene, it still has several limitations and challenges that researchers and practitioners continue to address. Some of the key limitations and issues of semantic segmentation include:

- **Fine-grained object boundaries:** Semantic segmentation algorithms may struggle to accurately delineate fine-grained object boundaries. Due to the nature of pixel-level classification, objects with intricate shapes or re-

gions with ambiguous boundaries can be challenging to segment accurately.

- **Occlusion handling:** Occlusions, where one object partially obscures another, pose challenges for semantic segmentation. Algorithms may struggle to correctly segment objects that are occluded or only partially visible in the image.

- **Scale and resolution variations:** Semantic segmentation models trained on a specific resolution may have difficulty handling images with different scales or resolutions. Scaling up or down an image can affect the model's ability to capture fine details or maintain spatial relationships accurately.

- **Computational complexity:** Semantic segmentation, particularly with deep learning models, can be computationally expensive and resource-intensive. Training and inference can require significant computational power and memory, limiting real-time performance on resource-constrained devices or in scenarios with strict latency requirements.

- **Limited generalization:** Semantic segmentation models trained on one dataset or domain may struggle to generalize well to unseen or different datasets or domains. Models often exhibit a bias towards the characteristics of the training data, leading to reduced performance on new and diverse data.

- **Class imbalance:** In many semantic segmentation datasets, there can be a significant class imbalance, where certain classes are over-represented or underrepresented. This class imbalance can bias the learning process and result in poor performance for minority classes.

- **Availability of limited datasets** Generating pixel-level annotations for training data can be laborious, time-consuming, and expensive. The process often requires manual annotation by human experts, and the subjective

interpretation of semantic classes can introduce annotation inconsistencies and errors.

- **Real-time performance:** Achieving real-time semantic segmentation in applications such as autonomous driving or robotics can be challenging. Balancing accuracy and speed is crucial, as real-time systems require fast and efficient segmentation algorithms to process high-resolution images or video streams in real-time.

Researchers are actively working on addressing these limitations and developing new techniques to improve the accuracy, efficiency, and robustness of semantic segmentation methods. The field continues to evolve with advancements in deep learning architectures, data augmentation techniques, and the integration of additional contextual information to overcome these challenges.

1.1.4 Resource-constrained applications

As semantic segmentation provides rich contextual details about the scene, it usually requires a large model as discussed previously. These large models usually have large memory footprints, high computational cost, and consequently large energy consumption. In many practical situations, however, there are constraints on memory, computational power and low energy usage requirements, such as mobile, IoT and edge devices. Likewise, some other situations may require the processing of multiple video streams for a fixed computational resource. As such, there is a real need for specialised semantic models that can perform well whilst meeting these constraints and provide real-time performance with desirable accuracy. Some applications that could benefit from lightweight semantic segmentation models include

- **Mobile Augmented Reality:** Augmented reality (AR) Zhang et al. (2020); Tanzi et al. (2021) applications on mobile devices often require real-time semantic segmentation for tasks such as object detection and scene

understanding. Lightweight semantic segmentation models enable efficient inference on mobile devices, allowing AR applications to run smoothly and provide interactive and responsive user experiences.

- **Robotics Navigation:** Robotics navigation refers to the process by which robots autonomously navigate and move in their environment. It involves perceiving and understanding the surroundings, planning a path or trajectory, and executing the necessary actions to reach a desired location or perform a specific task. Semantic segmentation Kim & Seok (2018) enables robots to navigate and interact with their environment effectively. By segmenting objects and obstacles in real-time, robots can plan their movements, avoid collisions, and perform tasks more efficiently in dynamic and cluttered environments. For instance, a blind patient with wheelchair assistance can navigate efficiently in an indoor environment with the help of real-time semantic segmentation model.

- **Smart Surveillance Systems:** Processing videos in a real-time environment is always a challenging task, particularly for surveillance systems Lai et al. (2021); Abdullah & Jalal (2023) that analyze live video streams from multiple feeds for object detection, tracking, and anomaly detection. These systems can benefit from a lightweight real-time semantic segmentation model. A lightweight segmentation model enables real-time processing and efficient utilization of computational resources, enabling the surveillance system to operate continuously without overwhelming the available hardware.

- **Medical Image Analysis:** Medical imaging tasks, such as tumor segmentation Zheng et al. (2022), organ localization Cai & Wang (2023), and tissue classification Progga & Shatabda (2023), often require semantic segmentation models to extract detailed information from images. In resource-constrained healthcare settings, lightweight models are essential for efficient

processing on devices with limited computational power, enabling faster diagnoses and treatment planning.

- **Environmental Monitoring:** Resource-constrained applications in environmental monitoring, such as remote sensing or satellite imagery analysis Wu et al. (2019a); Yang & Tang (2021); Khan et al. (2022), often require semantic segmentation models to identify land cover types, monitor changes, and detect anomalies. Lightweight models enable processing on edge devices or in remote locations with limited connectivity, facilitating timely and cost-effective analysis of environmental data. In modern-day applications, unmanned aerial vehicles (UAVs) Stache et al. (2023) are widely utilized for various tasks such as identifying agricultural lands, residential areas, commercial zones, and streets for urban planning. Similarly, in the agriculture field, low altitude flying drones Revanasiddappa et al. (2020) are employed for plant detection. However, these devices often have limited hardware capabilities. The integration of a lightweight semantic segmentation model can empower these embedded devices to efficiently perform detection and segmentation tasks.

- **Structural Monitoring:** Structural monitoring Nayyeri & Zhou (2021), such as building or road crack detection Choi & Cha (2019), often requires a real-time detection technique that efficiently detects cracks and notifies the authorities if further maintenance is required. A lightweight semantic segmentation model embedded in a resource-constrained mobile device can effectively accomplish this in a real-time environment.

The aforementioned applications highlight the significant demand for developing real-time semantic segmentation models. While numerous scene segmentation models exist, it is important to note that many of these models are not suitable for resource-constrained applications. The Figure 1.4 shows the plot of existing models' validation mIoU on Cityscapes Cordts et al. (2016) dataset against frame per

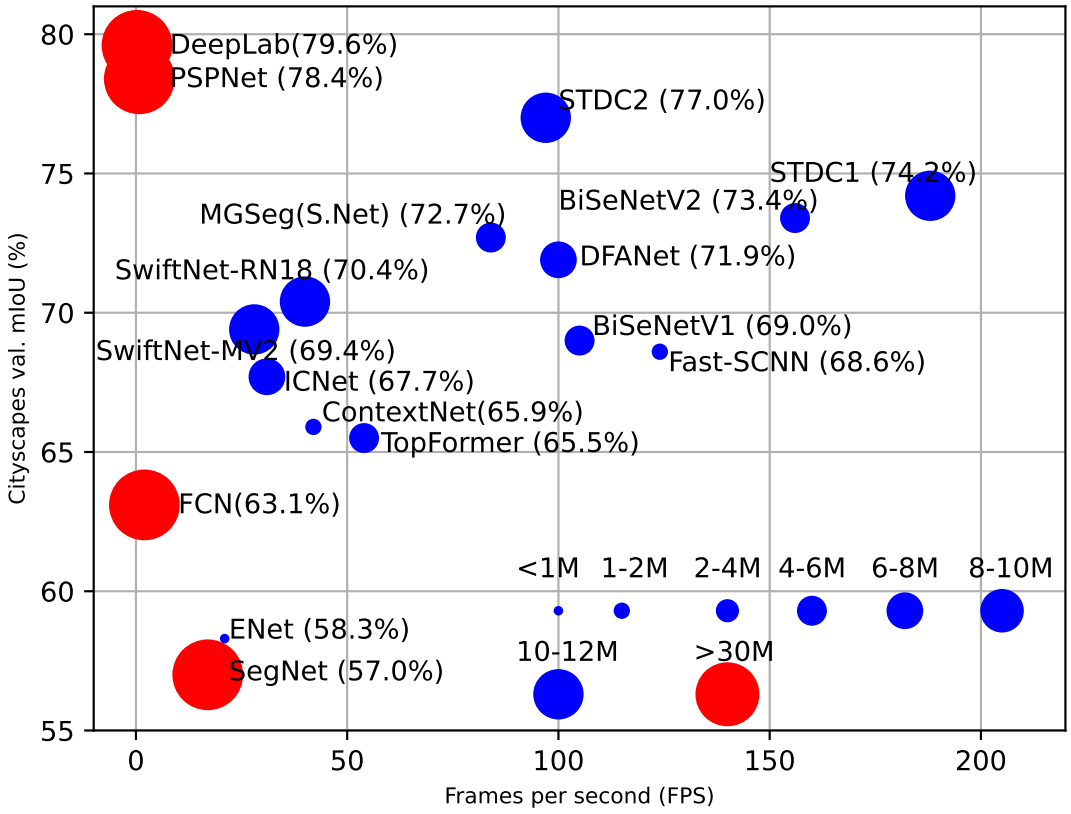

Figure 1.4: Cityscapes validation mIoU Vs frame per second (FPS) of existing offline and real-time models. The varying sizes of the dots define the different numbers of parameters of the models.

second (FPS). The red dot defines the existing offline scene segmentation models and the blue dot defines real-time semantic segmentation models. The size of each dots illustrates the size of the models. Generally offline models have more than 30 million parameters, whereas real-time semantic segmentation models have 0.4 to 16 million parameters. As offline model focuses on improving accuracy than efficiency, hence most of the offline models do not present their FPS count in the literature. The thesis presented four offline models in the plot and it can be clearly seen that the state-of-the-art offline models such as DeepLabV3+ Chen et al. (2018), PSPNet Zhao et al. (2017) not even process 1 frames per second. In comparison to offline models, existing real-time models can process more number of frames. However, in order to achieve a desirable trade-off between model's

accuracy and efficiency, a continuous research work has been carried out in the field of semantic segmentation.

The thesis also focusing on developing on real-time semantic segmentation model which is suitable for resource constrained mobile devices and can handle high resolution input images with less computational cost. The key reasons for developing real-time semantic segmentation models for resource constrained applications are:

- **Computational Efficiency:** Resource-constrained applications often operate on devices with limited computational power, such as mobile devices or embedded systems. Very large neural network models require significant computational resources, making them impractical for deployment on such constrained devices. Developing specific models allows for the optimization of computational efficiency, ensuring that the models can run smoothly within the given constraints.

- **Memory Footprint:** In addition to computational efficiency, the memory footprint of a model is a critical consideration for resource-constrained applications. Large neural network models tend to have a high number of parameters, which increases their memory requirements. On devices with limited memory, it becomes necessary to develop models that are compact and can fit within the available memory constraints while maintaining acceptable performance levels.

- **Latency and Real-Time Processing:** Some applications, such as real-time processing or edge computing scenarios, require low latency and fast inference times. Very large neural network models often have complex architectures that introduce higher computational and memory requirements, leading to increased inference times. Specific models can be designed with optimized architectures that minimize latency and enable real-time processing, meeting the needs of resource-constrained applications.

- **Energy Efficiency:** Energy consumption is a critical concern for devices that run on batteries or have limited power sources. Large neural network models with high computational requirements can quickly drain the device's battery or exceed the available power budget. Developing specific models allows for the design of energy-efficient architectures and algorithms, reducing the computational load and prolonging the device's battery life.

- **Adaptability to Task Requirements:** Resource-constrained applications often have specific requirements and constraints unique to their use cases. By developing specific models, researchers and engineers can tailor the model architecture, size, and complexity to best suit the application's needs. This customization allows for efficient use of available resources and ensures that the model performs optimally in the specific context, without unnecessary overhead from a large, general-purpose model.

In essence, creating specialized models tailored for resource-constrained applications offers advantages such as enhanced computational efficiency, reduced memory usage, minimized latency, improved energy efficiency, and adaptability to the specific needs of the application. These factors play a vital role in ensuring successful deployment and optimal performance in limited environments.

1.2 Literature review - Existing approaches for semantic segmentation

1.2.1 Conceptual foundation

Semantic segmentation is an important active research topic in the field of computer vision. Before the success of convolutional neural networks, the best ways of classifying the pixels independently were handcrafted feature-based methods such as graph cuts Freedman & Zhang (2005), Ramdom forests Shotton et al. (2008), Conditional Random Fields Plath et al. (2009). These methods often require expert domain knowledge and extensive manual feature engineering. They

rely on the design of appropriate features and the development of algorithms to extract and process these features effectively. While these methods were widely used in the past, the advent of deep learning and CNNs has significantly surpassed their performance, allowing for more accurate and end-to-end learning-based approaches for semantic segmentation.

In semantic segmentation, deep CNNs are firstly utilized by FCN Long et al. (2015). It has shown a revolutionary approach to deploying a deep CNN as an encoder of segmentation model. It adopted the contemporary classification models VGGNet Simonyan & Zisserman (2014) and GoogLeNet Szegedy et al. (2015) as feature extractor and replaced the top fully connected layers of the CNN by a convolution layer to generate a spatial map and then the spatial map is given to the decoder network in order to produce a segmentation output. Thus, an encoder-decoder architecture is designed. The general architecture of an encoder-decoder network can be seen in Figure 1.3. Following this novel design, later on several semantic segmentation models are introduced. The literature review on semantic segmentation using deep CNNs is designed based on the Figure 1.5. All the existing deep CNN based semantic segmentation models follow one of these network architectures.

1.2.2 One-branch encoder design

The most popular approach of designing a semantic segmentation model is to adopt an existing deep image classification model as an encoder and relying on the extracted deep feature maps by the encoder, design a decoder for generating a dense semantic map for pixel labelling. Most of the existing segmentation models which have adopted an existing deep CNN model as feature extractor, are following one-branch encoder design. The generale layout of the one-branch encoder design can be seen in Figure 1.5. The pioneer semantic segmentation model, FCN Long et al. (2015) also followed one-branch encoder design. It adopted VGGNet Simonyan & Zisserman (2014) or GoogLeNet Szegedy et al. (2015) as an encoder

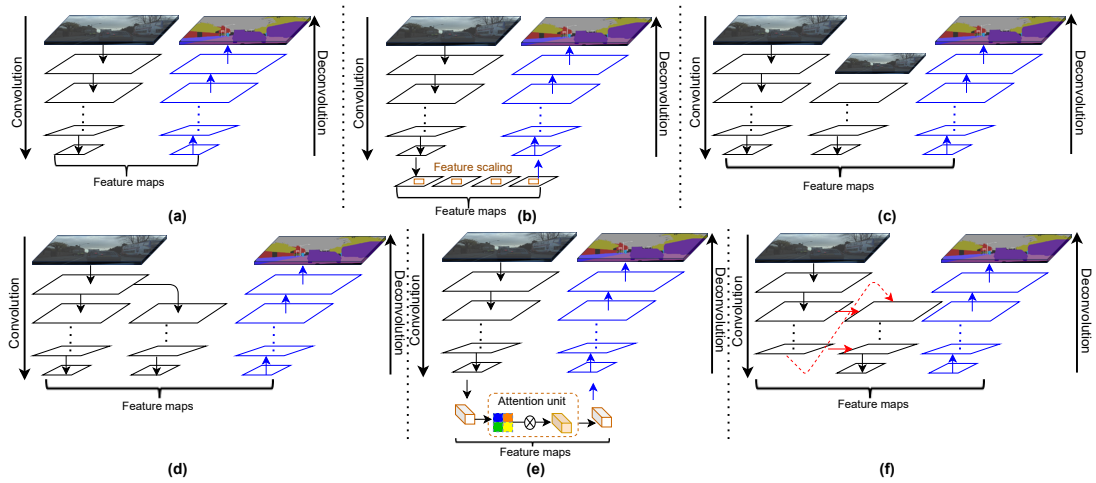

Figure 1.5: Different existing architectures used for semantic segmentation. (a) One branch encoder design, (b) One branch encoder using feature scaling technique, (c) Multi-branch encoder design, (d) Two-branch encoder design, (e) Encoder-decoder design with attention mechanism, (f) Feature reuse at encoder.

and replaces the fully connected layers by a convolution decoder. Followed by this design, a model called U-Net Ronneberger et al. (2015) was introduced for biomedical image segmentation. Along with a convolution path for capturing contextual details, it deployed a symmetric deconvolution path that enables precise objects localization. This symmetry helps in capturing both local and global context information for accurate segmentation. It also deployed skip connections from encoder to decoder to preserve the spatial details. Following this symmetrical architecture, another model named SegNet Badrinarayanan et al. (2017) is designed. It uses convolutional layers in the encoder for down-sampling and stores the max-pooling indices for each pooling operation. In the decoder, it uses upsampling layers that utilize the stored indices to efficiently reconstruct the segmentation map. Unlike, U-Net Ronneberger et al. (2015), it did not utilize skip connections which makes SegNet computationally efficient than U-Net.

1.2.3 One-branch encoder using feature scaling technique

Relying on encoder-decoder architecture later on different variants of DeepLab models Chen et al. (2017, 2018), PSPNet Zhao et al. (2017), DenseASPP Yang

et al. (2018) were introduced which proposed a new feature scaling technique for capturing rich contextual details at various scales from the deep global feature maps. DeepLabV3+ Chen et al. (2018) proposed Atrous Spatial Pyramid Pooling (ASPP) module which uses dilated convolution branches with higher dilation rates to have an effective receptive field of various sizes. It enables the model to capture both local and global context, leading to improved segmentation accuracy. The model demonstrated state-of-the-art performance in semantic segmentation due to its' large backbone (ResNet-101 Wu et al. (2019b) or Xception Chollet (2017)) and ASPP module. Motivated by the feature scaling technique, literature Yang et al. (2018) introduced another semantic segmentation model named DenseASPP. It is built upon on DenseNet Huang et al. (2017) classification model. It deploys ASPP on top of the DenseNet backbone for capturing multi-scale contextual information. The dense architecture facilitates feature reuse and enhances gradient flow, which leads to improved feature representation. It does not explicitly include a separate decoder module like DeepLabV3+. Other feature scaling technique such as Pyramid Pooling Module (PPM) is introduced by PSPNet Zhao et al. (2017). Instead of dilated convolution branches, it deploys multiple image pooling branches with different pool sizes. Thus, it provides various sizes of receptive field for better capturing of contextual details.

1.2.4 Multi-branch encoder design

Figure 1.5(c) depicts the multi-branch encoder design. The reason behind of this approach is to reduce the computational power and improve model's performance. Processing high resolution input images through a deep CNN such as ResNet-101 Wu et al. (2019b) generates a large computer overhead. Moreover, as the input image passes through a deep encoder, it loses the local spatial details, but produces rich contextual feature maps. However, for better semantic output, local information such as textures, boundary details are important. Keeping these thing in mind, a model called RefineNet Lin et al. (2017a) is introduced. The core

idea of this model is to deploy a deep branch for low resolution input images and employ multiple parallel shallow branches for varied higher resolution images. In this way, deep branch will not create large computer overhead and at the same time, multiple shallow branches extract local spatial details and fuse it with the deep feature map at the end of the encoder pipeline. Thus, model's performance is enhanced. Even though, this modle is designed to reduce the computational cost, but due to the deployment of ResNet-101 or ResNet-152 deep CNN model as the deep branch, model still produces a large computer overhead.

All these models, discussed above are basically offline semantic segmentation models due to their large backbone, large number of parameters, large FLOPs and large memory usage. However, growing demand of developing real-time applications in this domain has shown the need for resource friendly semantic segmentation model. To fulfill the needs, the literature Zhao et al. (2018) introduced an optimized multi-branch semantic segmentation model name ICNet. It reduced the number of parameters to 6.7 million and achieved a good balance between speed and accuracy. In order to reduce the model size further and makes the model more resource friendly, different multi-branch models such as ContextNet Poudel et al. (2018), and SwiftNet Orsic et al. (2019) were introduced. Among these, ContextNet is the smallest model, having only 1.1 million parameters. It deploys only two branches: one deep and one shallow branch. ContextNet is more resource friendly among the multi-branch models. However, in terms of accuracy, SwiftNet, which has more than 12 million parameters, produces better results among the multi-branch models. All these models are designed based on multi-branch approach where all branches are independently extracting the feature maps at various scales.

There are some shallow semantic networks such as ENet Paszke et al. (2016), CGNet Wu et al. (2020) which have only one-branch at the encoder. These models have very less parameters (around 0.5 million). However, due to the shallow architecture, the accuracy of these models dropped dramatically.

1.2.5 Two-branch encoder design

Among all the aforementioned models designed based on multi-branch encoder design, ContextNet Poudel et al. (2018) has utilized only two branches, one for deep global feature maps and another for shallow feature maps. This model optimized the design of multi-branch encoder. However, model still accepts two different sizes of input which increases the training time. For on-the-fly data augmentation techniques during the training process, input of both sizes need to be processed separately which causes the increase of time during the training. To overcome this issue, researchers proposed a new designed called two-branch design where the shallow branch, parallel to the main branch or deep branch, is created after initial few stages. In this design, model accepts the input at one resolution. Hence, all the data processing operations can be done only on one resolution input images, causes less training time. This designed can be seen in Figure 1.5(d). Following this approach, several models such as Fast-SCNN Poudel et al. (2019), BiSeNetV1 Yu et al. (2018), and BiSeNetV2 Yu et al. (2021) were introduced. Among these models, Fast-SCNN is the smallest two-branch segmentation model which has 1.1 million parameters. BiSeNet proposes two paths: Spatial Path (SP) which is a shallow branch designed for affluent spatial details and Context Path (CP) which is a deep branch, consisting of Xception Chollet (2017) deep CNN network and a global average pooling layer on top of the CNN, designed for contextual granularity. With ResNet18 Wu et al. (2019b), the baseline BiSeNet has 49.0 million parameters and with Xception39 Chollet (2017), it has 5.8 million parameters. Recently a new model called STDC Fan et al. (2021) is introduced which inherits the context path (CP) of the BiSeNet and excludes the spatial path (SP) in their backbone design. It introduced a detailed guidance technique which generates the ground-truth of the boundaries of each object in the scenes from the actual semantic segmentation ground-truth and provide this additional spatial details at the third stage of the context path. Thus, it fulfills the need of spatial details. This additional pre-processing operation is

only required during training. Relying on BiSeNet context path, it proposed two backbones: STDC1 and STDC2. STDC1 backbone has 8.4 million parameters, whereas STDC2 backbone has 12.5 million parameters. However, their end-to-end segmentation models have 12.0 and 16.1 million parameters, respectively. In comparison to BiSeNetV1 Yu et al. (2018) and BiSeNetV2 using Yu et al. (2021) Xception39 Chollet (2017) as backbone, both STDC versions have 2 to 3 times more parameters. As the model size is increased, hence the accuracy is improved by 4% to 7%. Moreover, the additional boundary knowledge also helps to improve the model's performance.

1.2.6 Encoder-decoder design with attention mechanism

Like feature scaling technique, attention mechanism is one of the useful techniques used by the computer vision researchers to improve the performance of the semantic segmentation model by focusing on important regions or features of an input image. It allows the model to allocate more computational resources and attention to regions that are semantically rich and contribute more to the segmentation task. Relying on attention module, several deep CNN models are designed for image classification. For example, MobileNetV3 Howard et al. (2019) uses squeeze-and-excitation (SE) module inside each inverted residual block for focusing on re-calibrating channel-wise feature responses. It consists of two main operations: squeeze and excitation. The squeeze operation globally aggregates spatial information by applying global average pooling over the input feature maps. The excitation operation then models channel dependencies by learning channel-wise weights using fully connected layers. This allows the model to adaptively emphasize informative channels and suppress less relevant ones. Likewise, Convolutional Block Attention Module (CBAM), proposed by the literature Woo et al. (2018), combines both channel attention and spatial attention mechanisms in order to capture inter-dependencies between different channels and finds the relationships between different spatial locations in the feature maps. CBAM shows

slightly better performance than SE when it is deployed with MobileNet Sandler et al. (2018) image classification model.

Motivated by these attention mechanisms, researchers started exploring different attention modules in the domain of semantic segmentation. Literature Fu et al. (2019) introduced an offline segmentation model called Dual Attention Network (DANet). Like, CABM, it consists of two attention modules: the Position Attention Module (PAM) and the Channel Attention Module (CAM). The PAM captures long-range dependencies by modeling the relationships between different positions in the feature maps. The CAM re-calibrates the importance of different feature channels. Relying on ResNet-101 Wu et al. (2019b) backbone and the proposed attention modules, DANet generates better results than existing offline segmentation models. Generally, attention module is deployed on top of the encoder to filter out the noises from the global feature maps. This general layout can be seen in Figure 1.5. However, it can be deployed in each building block of the encoder like SE and CBAM.

Relying on another attention module called Obejct-Contextual Representations (OCR), the literature Yuan et al. (2020) introduced an improved version of HRNet Sun et al. (2019) for image segmentation. The module OCR focuses on two main aspects: intra-object context and inter-object context. The intra-object context refers to the contextual information within individual objects and The inter-object context deals with the relationships and dependencies between multiple objects in an image. Thus, relying on a large HRNet backbone, OCR produces state-of-the-art performance on various public datasets. However, due to large number of parameters ($>$ 79 million), model's efficiency is dropped. The literature Choi et al. (2020) proposed another attention module called Height-driven Attention Networks (HANet) which is used as a general add-on module to semantic segmentation for urban-scene images. It adopts DeepLabV3+ as a baseline and add HANet at five different stages. The proposed HANet attention module extracts height-wise contextual information from the input feature

map and then computes height-driven attention weights and use it as an attention vector to guide the input feature map. Based on HANet attention module, two semantic models were proposed: MobileNetV2 + HANet and ResNet-101 + HANet. The mobileNet version has 15.4 million parameters, whereas ResNet version has 65.4 million parameters.

All the aforementioned models using attention mechanism are offline model due to their large network architecture. However, there are some shallow models which focus on real-time semantic performance with adequate accuracy. Model like DANet Li et al. (2019) deployed a simple yet effective Fully Connected (FC) attention module on top of the encoder branch to enhance feature representations and improve the accuracy of segmentation predictions. It also proposed two models: DFANet-A and DFANet-B using two different Xception Chollet (2017) backbones. The number of parameters of these two versions ranges from 4.8 to 7.8 million.

1.2.7 Feature reuse at encoder

All aforementioned semantic segmentation models follow a pyramidal shape to obtain global features with high-level semantics. However, due to deep network architecture, the rich global feature maps lack the local spatial details. This phenomena mainly observed in one-branch encoder design where a large CNN is used as an encoder. To mitigate this issue, the literature Li et al. (2019) introduced a concept of reusing the feature maps at the encoder side. For reusing the high-level features extracted from the backbone, successive encoder branches are added which accepts the high-level features from the predecessor branch. The depth of the model is increased in the successive branch. Thus, a Deep Feature Aggregation Network Li et al. (2019) is introduced for image segmentation. This feature reuse design can be seen in Figure 1.5(f).

1.2.8 Transformer based design

Transformer-based models have gained significant attention in natural language processing (NLP) tasks, but they have also been adapted for computer vision tasks, including semantic segmentation. The Vision Transformer (ViT)Dosovitskiy et al. (2020) introduced a CNN free transformer architecture for image recognition where input images are processed as sequences of patch tokens. After observing the excellent performance of ViT in image classification domain, SETR Zheng et al. (2021) introduced a transformer-based image segmentation model which uses ViT as backbone and a standard CNN as decoder. SETR divides the input image into a grid of fixed-size patches and applies a stacked hierarchical structure of transformers to capture both local and global context. The model uses self-attention to model dependencies between patches and generate segmentation predictions. Recently, a new model called Segmenter Strudel et al. (2021) is introduced which is a fully transformer-based encoder-decoder architecture. It deploys a variant of ViT Dosovitskiy et al. (2020) as an encoder to get the output embedding corresponding to image patches and uses a point-wise linear mapping or a mask transformer for obtaining class labels from these embeddings.

All the aforementioned models are very large and produce heavy computational cost which hampers their performance in resource constrained mobile devices. To address this issue, a mobile-friendly transformer-based model called Token Pyramid Vision Transformer (TopFormer) Zhang et al. (2022) is introduced which takes Tokens from various scales as input to produce scale-aware semantic features using ViT, then injects it into the corresponding tokens to augment the representation. The top variant of the TopFormer model has around 5.1 million parameter which is much less than the other transformer-based model. The performance of transformer-based models relies on large datasets.

These transformer-based models demonstrate promising results in semantic segmentation tasks by leveraging the self-attention mechanism to capture contextual dependencies and spatial relationships effectively. However, it's important

to note that the performance of transformer-based architecture relies on large datasets.

1.3 Summary of literature review

In summary, most existing semantic segmentation models utilize a pre-trained deep CNN image classification model as the encoder of their network. The ResNet Targ et al. (2016) architecture is one of the most popular CNN architectures, with ResNet-101 Wu et al. (2019b) being a commonly used variant. Deep variants of ResNet are primarily used for offline image segmentation, while smaller variants like ResNet-18 and ResNet-34 are employed for real-time computation. The MobileNet Sandler et al. (2018) image classification model is also commonly used for real-time computation due to its smaller memory footprint compared to the smaller variants of ResNet.

Various techniques such as feature scaling Zhao et al. (2017); Chen et al. (2018); Yang et al. (2018), feature fusion, feature attention Fu et al. (2019); Yuan et al. (2020); Choi et al. (2020); Li et al. (2019), and pre/post-processing methods Fan et al. (2021) are employed to enhance the performance of semantic segmentation models. These techniques aim to improve the model's ability to capture spatial and contextual information and refine segmentation results. Additionally, transformer-based architectures, which have achieved state-of-the-art performance in natural language processing tasks, have been introduced in the image segmentation field. Transformers process images by dividing them into smaller patches or tokens and utilize self-attention mechanisms to capture global context and model long-range dependencies. However, transformers are computationally more expensive compared to CNNs. Therefore, the choice between these models depends on the specific requirements of the task and the available computational resources.

Table 1.1 provides a summary of existing deep and shallow segmentation models designed based on different approaches. It is evident that the majority of the

existing semantic segmentation models employ a pre-trained deep CNN as an encoder, trained on large datasets. The offline models have more than 29.5 million parameters, resulting in high computational costs. In contrast, real-time models have a parameter range of 0.4 to 16.0 million. Due to the relatively shallower network architecture of the deep branch, most real-time models adopt a multi-branch or two-branch design approach. Some of the existing real-time scene segmentation models achieve better accuracy. However, it's important to note that these models still have a large number of parameters, which can affect the overall performance of the model in resource-constrained embedded devices. That is why ongoing research is focused on developing real-time semantic segmentation models for resource-constrained applications.

Table 1.1: Summary of the literature review

Type	Model	Design approach	Backbone	Parameters (Million)
Offline	FCN (Long et al. (2015))	One-branch	VGGNet	134
	U-Net (Ronneberger et al. (2015))	One-branch	VGGNet	>138
	SegNet (Badrinarayanan et al. (2017))	One-branch	VGGNet	29.5
	DeepLab (Chen et al. (2017))	One-branch with feature scaling	VGGNet	134.0
	DeepLabV3+ (Chen et al. (2018))	One-branch with feature scaling	Xception	54.6
	PSPNet (Zhao et al. (2017))	One-branch with feature scaling	ResNet	250.8
	RefineNet (Lin et al. (2017a))	Multi-branch	ResNet	68.2
	DANet (Fu et al. (2019))	One-branch with dual attention	ResNet	46.0
	HRNet (Sun et al. (2019))	One-branch	HRNet	65.9
	OCR (Yuan et al. (2020))	One-branch	HRNet	>76.4
	HANet (Choi et al. (2020))	One-branch with height-driven attention	ResNet	65.4
	SETR (Zheng et al. (2021))	Transformer-based	ViT	97.6
	Segmenter (Strudel et al. (2021))	Transformer-based	ViT	>307
Real-time	ENet (Paszke et al. (2016))	One-branch	Xception	0.4
	ICNet (Zhao et al. (2018))	Multi-branch	ResNet	6.4
	ContextNet (Poudel et al. (2018))	Multi-branch	Own backbone	1.1
	BiSeNetV1 (Yu et al. (2018))	Two-branch	Xception	5.8
	SwiftNet (Orsic et al. (2019))	Multi-branch	ResNet	12.9
	Fast-SCNN (Poudel et al. (2019))	Two-branch	Own backbone	1.2
	DFANet (Li et al. (2019))	Feature reuse and FC attention	ResNet	8.4
	CGNet (Wu et al. (2020))	One-branch	Xception	0.5
	BiSeNetV2 (Yu et al. (2021))	Two-branch	Xception	5.2
	MGSeg (He et al. (2021))	One-branch with hybrid attention	ResNet	13.3
	STDC (Fan et al. (2021))	Two-branch	BiseNet	16.0
	TopFormer (Zhang et al. (2022))	Transformer-based	ViT	5.1

1.4 Identification of research gaps in semantic segmentation

From Table 1.1, it is evident that deep CNNs have been extensively used in semantic segmentation models to extract meaningful information and improve the quality of segmentation results. In recent years, transformer-based designs have gained popularity in this field. However, they have not surpassed the performance of deep CNNs and require larger datasets and more computational resources. Overall, CNN-based models still outperform transformer-based approaches in terms of both accuracy and efficiency. However, there is still room for improvement in existing deep CNN-based models.

Most existing segmentation models rely on large, deep CNN architectures, which result in high computational overhead and make them unsuitable for real-time computation. While deep CNNs enhance model accuracy, they also increase the memory footprint and reduce efficiency. On the other hand, shallow models optimize the pipeline and reduce the number of parameters, making them suitable for real-time performance on mobile devices. However, their performance is significantly lower compared to deep CNN-based models.

To achieve the desirable trade-off between accuracy and efficiency, this thesis investigates the architectures of existing offline and real-time semantic segmentation models and identifies the following research gaps:

- **Lack of architectural control:** From Table 1.1, it can be clearly seen that most existing models employ pre-trained deep CNN models as encoders. While this provides a performance boost, it limits architectural modifications and optimizations when utilizing the weights of each layer. Moreover, deep CNNs are designed for image classification tasks where contextual details are important for identifying objects in a scene. They do not provide rich spatial details. In semantic segmentation models, both contextual details and spatial local details are required for accurate object localization

with precise object boundaries. Therefore, employing a deep CNN may not be beneficial for resource-constrained semantic applications.

- **Lack of fine-grained detail:** Many segmentation models struggle to capture fine-grained details due to down-sampling operations in the encoder stage. This can result in the loss of spatial information and produce coarse segmentation boundaries.

- **Difficulty in handling object scale variations:** Semantic segmentation models may encounter difficulties in accurately segmenting objects with varying scales. Objects that are smaller or larger than the distribution of the training set pose challenges for proper segmentation.

- **Ambiguity in object boundaries:** Semantic segmentation models often face challenges in accurately delineating object boundaries, especially when boundaries are ambiguous or poorly defined. This can lead to leakage or incorrect segmentation in regions with unclear boundaries.

- **Class imbalance and rare classes:** Imbalanced class distribution can impact the performance of segmentation models, particularly for rare or underrepresented classes. Models tend to favor majority classes, resulting in inadequate segmentation for less frequent classes.

- **Limited contextual reasoning:** Some segmentation models have limitations in capturing long-range dependencies and contextual information across the entire image. This can affect the model's ability to understand global context and make accurate segmentation predictions.

- **High computational and memory requirements:** Table 1.1 highlights that many existing scene segmentation models have a large number of parameters, leading to a significant memory footprint. These models require substantial memory resources for training and inference.

Addressing these research gaps is an active area of research, and ongoing advancements aim to improve the performance and robustness of semantic segmentation models. Techniques such as dilated convolutions, attention mechanisms, multi-scale processing, and data augmentation strategies are being explored to tackle these challenges and enhance the capabilities of semantic segmentation models.

1.5 Problem statement

From the literature review, it is evident that existing semantic segmentation models have several architectural issues that hinder them from achieving the sensible trade-off between model accuracy and efficiency. While these models may achieve outstanding results due to their large network architecture, their efficiency significantly drops in real-time environments. On the other hand, existing shallow scene segmentation models improve efficiency but suffer from a substantial drop in accuracy. Additionally, these models often struggle to capture long-range dependencies and accurately segment objects with varied geometrical shapes and boundary details.

To address these research gaps, this thesis presents several novel backbone architectures that are resource-friendly and capable of handling high-resolution input images (e.g., 1024×2048) in real-time environments. These novel designs provide larger receptive fields than existing architectures, enabling them to capture spatial and contextual details and long-range dependencies more effectively. These designs also reduce the number of parameters and floating-point operations, resulting in a smaller memory footprint and lower power consumption.

Furthermore, this thesis introduces techniques such as knowledge sharing, feature refinement, context mining modules, and hybrid path attention semantic aggregation methods for improved semantic representation and precise object localization. These methods enhance the overall performance of the models and bridge the gap between accuracy and efficiency. Importantly, they do not generate

large numbers of parameters or require a significant memory footprint, making them suitable for real-time performance.

1.6 Proposed research

1.6.1 Research questions

In this thesis, the following research questions are designed:

1. How can this thesis design a real-time semantic segmentation model for resource-constrained applications that can handle high-resolution input images with less computational cost?

2. How can this thesis address model scalability across all dimensions to improve the model's performance?

3. How can this thesis improve the model's performance to extract fine-grained details, capture long-range dependencies, and recognize objects of varied geometrical shapes?

4. How can this thesis address existing issues such as occlusion, rare classes, large semantic gaps, and gradient vanishing?

5. How can this thesis achieve a sensible trade-off between the model's accuracy and efficiency?

1.6.2 Research objectives

Main objective:

The main objective of this thesis is to develop real-time semantic segmentation models and applications that enhance scene understanding. These models and applications are designed to minimize memory consumption and power usage while simultaneously improving model accuracy and efficiency.

Sub-objectives:

In order to achieve the main objective, the following sub-objectives need to be addressed:

- **Model scalability:** To achieve a balance between model accuracy and efficiency, a proper scaling technique is required across all dimensions of the model. Publication 1 demonstrates the effectiveness of a compound scaling technique for extending the segmentation model uniformly along the depth, width, and input resolution, leading to improved performance. However, for real-time scene segmentation, expanding at a higher rate can reduce efficiency.

- **Optimized backbone architecture:** The overall performance of a semantic segmentation model heavily relies on the backbone network. While a deep backbone helps extract more contextual details, it also reduces efficiency. Therefore, an optimized backbone architecture with effective techniques is required to capture both spatial and contextual details. Publication 2 emphasizes the importance of designing a lightweight and resource-friendly backbone and feature aggregation unit for better semantic results.

- **Handling object scale variations:** Existing segmentation models often struggle to identify and locate objects of various shapes in complex scenes due to a lack of receptive field sizes throughout the feature hierarchy. Addressing this issue requires feature scaling techniques at different scales. Publications 3 and 5 introduce optimized scaling methods to handle object scale variations and localize objects of various shapes while considering the computational efficiency.

- **Achieving fine-grained object boundaries:** Local spatial information, such as object boundaries, patterns, and textures, plays a vital role in achieving better semantic results. Deep CNNs can provide contextual details, but the spatial details are often lost due to their deep architecture.

Publications 4 and 6 highlight the importance of an additional shallow branch for collecting spatial details from the feature maps. These publications also emphasize knowledge sharing between deep and shallow branches to enhance the entire feature hierarchy and address the issue of object boundaries.

- **Improving the feature hierarchy at the encoder and addressing gradient vanishing:** Gradient vanishing is a common issue observed in existing image segmentation models. To address this issue and improve the feature hierarchy at the encoder, publications 8 and 9 demonstrate the importance of deploying multiple sub-encoders for feature reuses. This approach helps mitigate the gradient vanishing problem and enhances the model's performance.

- **Class imbalance and rare classes:** Many segmentation datasets suffer from class imbalance, where the class distribution is not uniform, leading to models overlooking objects belonging to rare classes. Addressing this issue requires feature reuses through multiple sub-encoders to ensure that objects from rare classes have a higher chance of detection. Publications 8 and 9 emphasize the importance of feature reuses and its impact on the gradient vanishing issue. Additionally, various techniques such as data augmentation methods, stochastic gradient descent (SGD) optimization, and weighted loss functions (Publication 6) are employed across all publications to address the class imbalance issue.

- **Addressing limited contextual reasoning:** Capturing long-range dependencies and contextual information across the entire scene is challenging for many models due to limited field-of-view. Publication 7 introduces a novel architecture that utilizes filtering the input feature map with various sizes of receptive fields, enabling the model to capture long-range dependencies and contextual details in complex scenes.

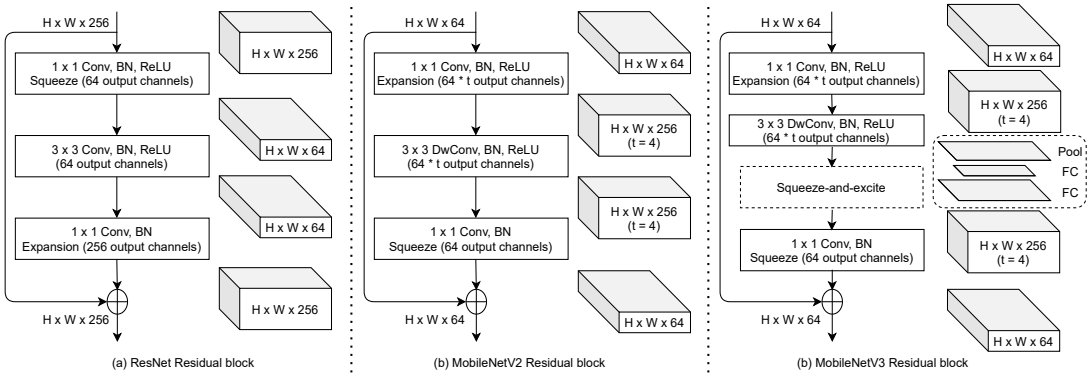

Figure 1.6: Layered architectures of ResNet, MobileNetV2, and MobileNetV3 residual blocks.

- **Reducing computational cost and memory footprint:** Most existing semantic models are not suitable for resource-constrained embedded devices due to high computational costs and large memory requirements. Optimizing the entire segmentation model pipeline becomes crucial to reduce computational costs and memory requirements without sacrificing accuracy. All publications in this thesis propose optimized model architectures, with parameter counts ranging from 1.2 to 2.7 million, significantly lower than existing semantic models, to address this challenge.

1.6.3 Research methodology

In order to achieve the objectives, this thesis developed several optimized architectures that can be deployed for various computer vision tasks. The foundation of each backbone relies on MobileNetV2 Sandler et al. (2018) inverted mobile residual blocks (MBConv). The MBConv block is designed for efficient deployment on resource-constrained devices such as mobile phones and embedded systems, where computational efficiency is crucial. It aims to strike a balance between model size and computational efficiency. Compared to the ResNet Targ et al. (2016) residual block, MBConv produces fewer parameters and FLOPs, drastically reducing the computational overhead.

Unlike the ResNet residual block, the MBConv block first expands the feature

map along the channel. Then, using a depth-wise convolution layer (DwConv), it filters the feature map along the depth. Finally, using a projection layer (1×1 Conv), it squeezes back the output channel of the feature map. MobileNetV3 Howard et al. (2019) introduces a new attention mechanism called the squeeze-and-excite block, placed between the DwConv and projection layer. This mechanism flattens the feature map into a one-dimensional vector using one pooling layer and two dense fully connected (FC) layers. However, dense layers contribute a large number of parameters and FLOPs in each MBConv block, which reduces the model's efficiency. Hence, except for publication 1, this thesis did not use the squeeze-and-excite block inside each MBConv block. The layered architecture of ResNet, MobileNetV2, and MobileNetV3 residual blocks can be seen in Figure 1.6.

The research methodology of each publication is summarized in Chapter 2. Additionally, this thesis provides each published paper in the appendix and demonstrates the methodology of each paper.

1.7 Evaluation approach

1.7.1 Datasets

To evaluate the performance of the proposed models and methods, this thesis utilized several publicly available benchmarks designed for various computer vision tasks. The primary focus of this dissertation is on complex urban street scenes captured in both structured and unstructured environments. These scenes are inherently more complex compared to indoor scenes due to the presence of diverse objects with different geometrical shapes. Furthermore, segmenting outdoor scenes poses additional challenges due to varying atmospheric conditions, environments, and geographic factors.

In addition to the urban street scene datasets, this study also incorporates one indoor scene dataset, as well as several building and road crack segmenta-

tion datasets. These datasets contribute to a comprehensive evaluation of the proposed models across different scenarios and challenges.

The full details of each dataset used in this thesis are provided below:

Cityscapes dataset

Cityscapes Cordts et al. (2016) is a widely utilized dataset for semantic segmentation, which involves categorizing objects in urban street scene images. The dataset offers images with a resolution of 1024×2048, where objects are divided into 35 classes and grouped into 8 categories. In line with the established practices for Cityscapes, we employed 19 classes for pixel annotations. The dataset comprises approximately 5,000 meticulously annotated images. Out of these, 2,975 images were used for training, 500 for validation, and the remaining 1,525 for testing. However, annotations for the test set were not provided with the dataset. To assess the proposed model's performance on the test set, model's test predictions are submitted to the Cityscapes online evaluation server, and the results were subsequently published. In addition to fine-tune dataset, it also provides additional 20,000 coarsely annotated images.

BDD100K dataset

BDD100K Yu et al. (2020) is a newly created dataset that caters to the increasing needs of the autonomous car industry. It stands out as the most extensive collection of driving videos, comprising 100,000 videos. In comparison to Cityscapes, this dataset presents greater challenges due to its diverse content. It offers 8,000 meticulously annotated images at the pixel level, with 7,000 designated for training purposes and 1,000 for validation. The class labeling system in this dataset aligns with that of Cityscapes, and further details can be found on our Github repository. Each image in the dataset possesses dimensions of 720×1280 pixels. However, due to tensor size compatibility, 768×1280 resolution is used for training.

KITTI dataset

Compared to Cityscapes Cordts et al. (2016) and BDD100K Yu et al. (2020), KITTI Alhaija et al. (2018) is a small dataset comprising of 200 urban street scene images with dense semantic information. It also provides an additional set of 200 stereo images without annotations for testing purposes. Each image of the dataset has size of 375×1242 resolution, but for training, 384×1280 resolution is used for better feature size compatibility. Similar to Cityscapes dataset, KITTI provides a test evaluation server. The test set results of all the proposed models which are trained by KITTI are submitted to the online evaluation server and results are published on KITTI leader board. It's worth noting that both KITTI and BDD100K datasets are compatible with the Cityscapes dataset, as they utilize the same class labeling system consisting of 19 classes.

CamVid dataset

CamVid Brostow et al. (2009) is a compact dataset primarily intended for object detection in autonomous driving vehicles. The dataset consists of images extracted from a recorded video, where human operators assigned a class color to each object in the frames. It comprises a total of 267 images for training, 101 for validation, and 233 for testing. For evaluating performance, the present study focuses on 11 classes (excluding the "void" class) out of the 32 available in the dataset. Due to its limited size, models trained on this dataset often face challenges in fully learning the underlying patterns.

IDD and IDD-lite datasets

All aforementioned datasets primarily focus on urban street scenes captured in western countries, specifically Europe or the USA. These environments have well-structured road conditions and fewer variations in the objects present. However, these well-defined traffic environments do not accurately represent the road conditions in Asian countries where road condition is not structured in proper way.

To evaluate the performance of the proposed model in unstructured road conditions, this thesis also explored the IDD-lite dataset (Indian Driving Dataset lite version) Mishra et al. (2020). This dataset comprises 1,404 urban and rural training images, 204 validation samples, and 404 test samples, each with a resolution of 227×320. The dataset classifies objects into seven categories: drivable, non-drivable, living things, vehicles, roadside objects, far objects, and sky.

Additionally, Indian driving dataset also provides a large dataset containing two parts: part 1 and part 2. Together, it contains approximately 14,027 training samples and 2,036 validation samples. This dataset offers a variety of labels, including label1 with 7 classes, label2 with 16 classes, label3 with 26 classes, and label4 with 30 classes. Furthermore, it includes Cityscapes with 19 classes labeling. However, for the evaluation of the model's performance on IDD, this thesis only used 7 classes, similar to IDD-lite. It's worth noting that the class labeling in the Cityscapes dataset is compatible with both IDD and IDD-lite. The resolution of input image of this dataset is 1080×1920.

Indoor object dataset

To analyze indoor scenes, we employed the Indoor Object dataset developed by Mohamed et al. (2022). This dataset is specifically designed for indoor navigation, with a focus on assisting wheelchair users and service robots. The dataset organizes the entire object space into nine distinct classes, which are particularly valuable for gaining a comprehensive understanding of indoor environments. For further information and specific details, readers are encouraged to refer to our supplementary materials.

Crack datasets

For crack segmentation task, this thesis utilizes the crack segmentation dataset sourced from the Kaggle data science community. This dataset, which consists of approximately 11,200 images, comes with detailed annotations for crack seg-

mentation. It combines 12 separate crack segmentation datasets, namely Zhang et al. (2016); Yang et al. (2019); Eisenbach et al. (2017); Shi et al. (2016); Amhaz et al. (2016); Zou et al. (2012), into a unified dataset. The thesis considers this particular dataset to be highly challenging due to its diverse range of crack types found on various surfaces such as roads, pavements, walls, buildings, and concrete structures. The dimensions of all images in the dataset are $448 \times 448 \times 3$. We divide the entire dataset into two subsets: a training set comprising 9,505 images and a test set containing 1,695 images. It is worth noting that many existing crack detection models are trained using private datasets that are not publicly accessible.

System configuration

For conducting experiments with relatively low-resolution input images, such as KITTI, IDD-lite, and Crack datasets, a computerized system equipped with dual Nvidia GeForce RTX 2080Ti GPUs, each having 11GB of memory, is used. This system provides the necessary computational resources for processing the data efficiently.

To handle higher-resolution images, such as Cityscapes and BDD100K, a system equipped with three NVIDIA Titan RTX GPUs, each having 24 GB of memory, is employed. This setup allows for effective processing of the larger and more computationally demanding images.

All scripts are written in the Python language. The deep learning model is developed using the `TensorFlow` 2.1.0 and `Keras` 2.3.1 deep learning software packages. These frameworks provide essential tools for building and training the model, as well as writing the necessary training and evaluating scripts.

To leverage the parallel processing capabilities of the GPUs and accelerate computationally intensive tasks, `CUDA` 10.2 is employed. CUDA is a parallel computing platform that enables efficient utilization of the GPUs for deep learning tasks.

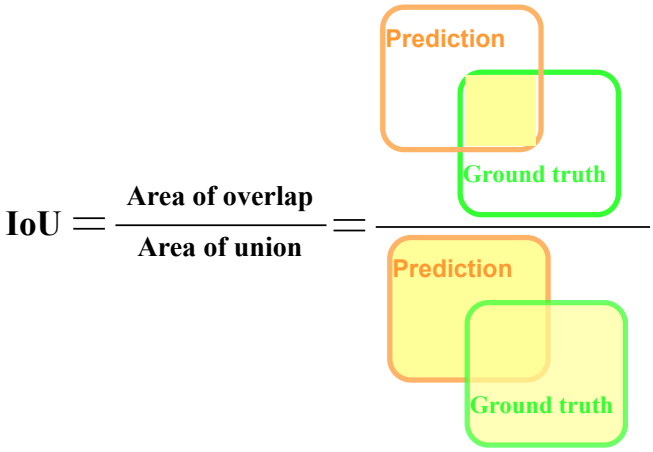

Figure 1.7: Intersection Over Union

For facilitating distributed training of deep learning models across multiple GPUs, the `Horovod` 19.5.0 library is utilized. `Horovod` provides a framework for distributed training, allowing efficient utilization of multiple GPUs and accelerating the training process.

For measuring frame per second (FPS), this thesis utilized `TensorRT` engine which converts the `TensorFlow` into TRT-based model. This is a common approach for measuring the FPS of real-time semantic segmentation models.

Performance measurement metrics

The primary performance measurement metric for semantic segmentation is the mean Intersection Over Union (mIoU) metric, as outlined by Everingham et al. (2015). It stands as a popular choice for evaluating the localization accuracy of objects within a scene. The metric calculates the extent of overlap between the ground truth class and the predicted class, with a visual representation of IoU shown in Figure 1.7. In the context of binary segmentation masks, this metric is computed using a confusion matrix that contains the counts of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN), as illustrated in Figure 1.8 for a two-class scenario.

For multi-class semantic segmentation challenges, the mean IoU (mIoU) is

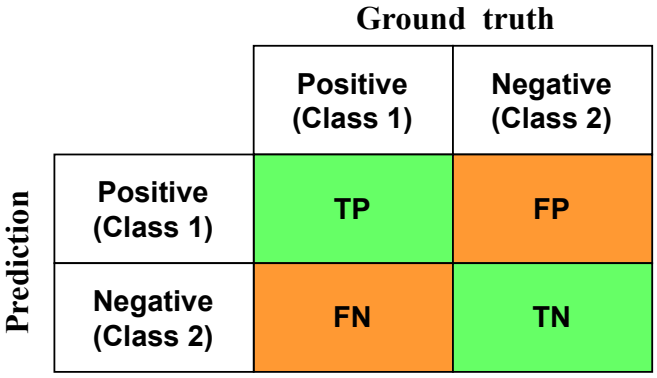

Figure 1.8: Confusion matrix

computed for each class. Evaluation servers like Cityscapes Cordts et al. (2016) and KITTI Alhaija et al. (2018) provide class-based and category-based mIoU. Furthermore, they calculate the instance-level intersection-over-union metric (iIoU) to gauge the representation of individual instances within the labeling. The formulas for IoU and iIoU are shown in Equations 1.1 and 1.2.

$$IoU = \frac{TP}{TP + FP + FN} \tag{1.1}$$

$$iIoU = \frac{iTP}{iTP + FP + iFN} \tag{1.2}$$

In contrast to the conventional IoU measure, iTP and iFN are computed by factoring in the contribution of each pixel, weighted by the ratio of the class's average instance size to the size of the corresponding ground truth instance. To evaluate model efficiency, metrics such as model parameters, Floating Point Operations (FLOPs), and Frames Per Second (FPS) are employed. These metrics provide insights into the computational cost and real-time performance of the model.

For tasks related to crack detection and segmentation, various performance metrics come into play, including F1 score, precision, recall, and accuracy. Much like IoU, these metrics are also calculated based on the confusion matrix, with

their respective formulae depicted in Equations 1.3, 1.4, 1.5, and 1.6.

$$Precision = \frac{TP}{TP + FP} \tag{1.3}$$

$$Recall = \frac{TP}{TP + FN} \tag{1.4}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1.5}$$

$$F1score = \frac{2TP}{2TP + FP + FN} \tag{1.6}$$

1.7.2 Research contributions

Main Contribution:

The main contribution of this thesis is to design real-time semantic segmentation models for resource-constrained mobile devices. This is achieved through the novel design of lightweight backbone networks and the implementation of optimized feature refinement and feature fusion strategies. These strategies aim to strike the sensible balance between model accuracy and model efficiency.

Sub-contributions:

The thesis comprises nine research articles, each addressing different sub-objectives to achieve the main objective. The contributions of all the peer-reviewed research articles are:

- Publication 1 has demonstrated that improved semantic performance can be achieved by uniformly scaling the model in all dimensions. By employing compound scaling techniques, it introduced a family of semantic segmentation models that produce better results than many existing semantic segmentation models.

- Publication 2 introduced a highly lightweight scene parsing model called FANet, which was designed from scratch using MBConv blocks from MobileNet Sandler et al. (2018). By incorporating a modified design of a

bi-directional feature pyramid network in the decoder, this model improved the entire feature hierarchy, enhancing object localization accuracy and providing rich contextual details.

- Publication 3 introduces an efficient feature scaling module for scale-variant objects in the scene and a feature fusion module for reducing the larger semantic gap among the local and global feature maps.

- For real-time structural crack detection and semantic performance, Publication 4 introduces a novel shared two-branch backbone design in which the shallow and deep branches share their knowledge, ultimately enhancing the overall performance of the proposed model. It also presents a context mining module aimed at filtering out noisy data from the shared feature maps. The performance of the proposed SCMNet is evaluated using various structural crack datasets, and it outperforms existing models to achieve the best results.

- Inspired by the neck architecture of object detection techniques, Publication 5 proposed a Multi-level Multi-path Feature Aggregation Network that effectively captures both fine-grained details and high-level semantics from the feature maps, resulting in improved real-time semantic performance. Thanks to its multi-level multi-path feature aggregation technique, complex scenes with various shapes of objects can be segmented efficiently.

- Publication 6 introduces an efficient real-time structural crack detection and segmentation architecture called SC-CrackSeg. The proposed architecture has achieved state-of-the-art performance on various crack datasets compared to existing models.

- Publication 7 introduces a novel dense bottleneck design called the Short-term Dense Bottleneck (SDB) design, which offers a range of receptive fields for capturing various geometrical shapes within complex scenes and en-

hances the model's semantic performance. It also presents feature refinement and semantic aggregation modules for context mining and assimilation.

- To reduce the backbone size of a semantic segmentation model and enhance real-time performance, Publication 8 proposed a novel architecture called the Shared Feature Reuse (SFR) network. This network filters the intermediate and deep feature maps through multiple sub-encoders to address the gradient vanishing issue. To narrow the semantic gap among the feature maps, it deploys the Knowledge Sharing Block (KSB), which fuses feature maps from the same level of various sub-encoders and then filters out noise from the shared feature maps. The article demonstrates an effective approach to designing semantic segmentation models for resource-constrained embedded devices.

- Inspired by the feature reuse design proposed in Publication 8, Publication 9 introduces the Multi-encoder Context Aggregation Network (MCANet) for analyzing both structured and unstructured urban street scenes. This network introduces a Local and Global Context Aggregation (LGCA) module at the decoder end, which fuses spatial and global feature maps through multiple paths in both directions to achieve accurate object localization and improved semantic representation. The proposed MCANet is evaluated using five datasets and demonstrates state-of-the-art performance among existing real-time semantic segmentation models across various datasets.

Additional details on each publication's contributions can be found in Chapter 2.

1.7.3 Research outcomes

The thesis has published nine research articles in top-ranked journals and conferences. Each publication addresses different sub-objectives in order to fulfill the main objective. Novel backbone architectures are introduced that are efficient in real-time environments and produce impressive results. The concept of

knowledge sharing among different branches of the encoder and context mining of the shared feature map is introduced, improving the entire feature hierarchy across all branches. The thesis also presents the concept of feature reusing and re-processing by multiple sub-encoders, which helps capture long-range dependencies and contextual information from complex scenes. Lateral connections among the sub-encoders are utilized to address gradient vanishing issues. Such novel backbone architectures can be deployed for various computer vision tasks.

To achieve better coarse-to-fine refinement of shallow and deep features, several optimized feature scaling and feature refinement methods are introduced. These methods effectively filter out noise from the feature maps and generate impressive semantic results. Other techniques such as feature fusion, hybrid path attention, and feature aggregation are also introduced to enable accurate object localization. With these novel designs, the thesis successfully achieves a sensible trade-off between model accuracy and efficiency.

1.8 Thesis structure

The thesis consists of three chapters, followed by all the published articles.

- **Chapter 1:** In this chapter, the thesis begins by discussing image classification and object detection tasks and their limitations. Based on these limitations, the chapter then introduces semantic segmentation tasks. A comprehensive literature review of various existing semantic architectures, methods, and techniques is provided. The chapter also highlights the existing research gap, presents the problem statement, proposed research, experimental details, research contributions, and the outcomes of the research.

- **Chapter 2:** This chapter provides a summary of all the published articles, including the research work conducted, and presents the overall contributions of the thesis.

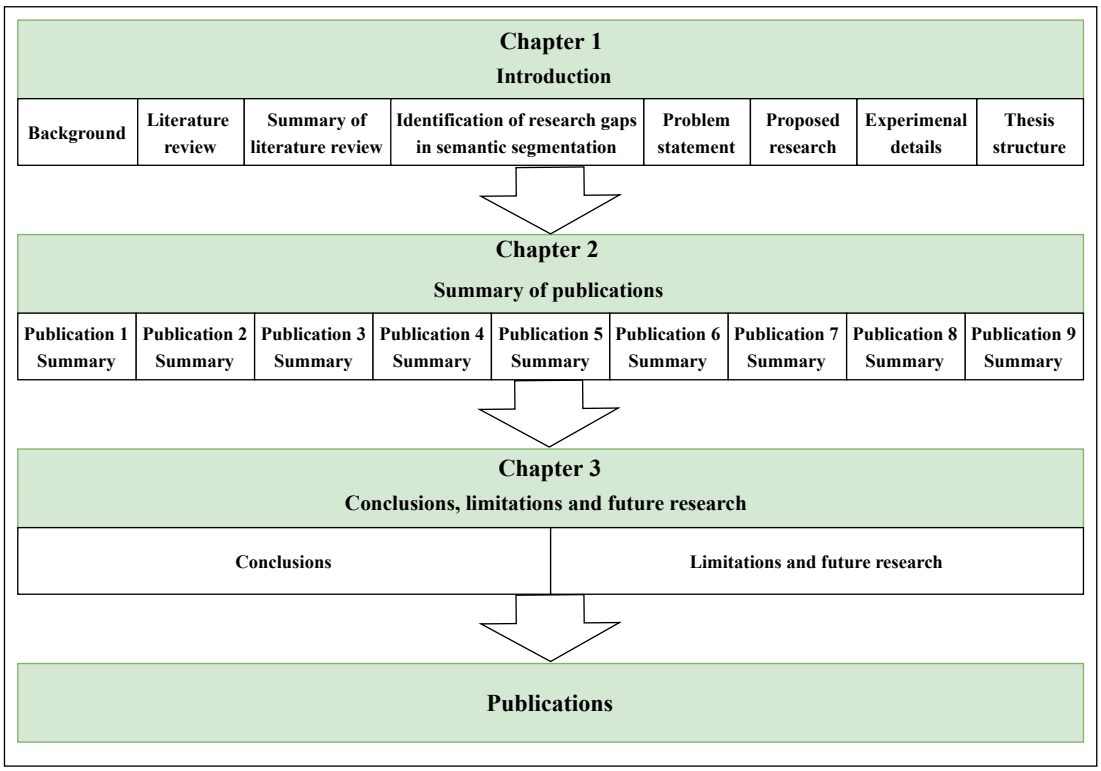

Figure 1.9: Thesis structure

- **Chapter 3:** This chapter offers an overall conclusions of the research methodologies employed, along with the future scope of the research.

The aforementioned chapters are followed by the published articles, represented by the final versions authored by the researchers. The graphical layout of the thesis structure can be seen in the Figure 1.9.

Chapter 2

Summary of publications and contributions

This chapter shows a summary of the eight published research papers as part of this thesis. All publications collectively contribute to addressing the research objectives of the thesis. They are summarised in this chapter, followed by the overall contribution of the thesis.

2.1 Publication 1: Efficient Segmentation Pyramid Network

Bibliographic reference:

Singha, T., Pham, DS., Krishna, A., Dunstan, J. (2020). **"Efficient Segmentation Pyramid Network"**. In: ICONIP 2020. vol 1332. Springer, Cham. https://doi.org/10.1007/978-3-030-63820-7-44.

2.1.1 Abstract

Designing an efficient semantic segmentation model for resource-constrained mobile devices is a challenging task for computer vision researchers. A popular approach for designing a segmentation model is to use an existing deep convolutional neural network (DCNN) as an encoder and design a deconvolution network to generate semantic output. To improve accuracy, a deep DCNN is often deployed as the encoder, with an increased depth for the backbone. However, literature shows that scaling the model in one dimension while neglecting other dimensions does not significantly enhance the model's performance. Additionally, deploying more convolutional layers increases the number of parameters and floating-point operations (FLOPs), resulting in a large memory footprint.

To address these challenges, a new semantic segmentation model called Efficient Segmentation Pyramid Network (ESPNet) is introduced for real-time computation. By leveraging the compound scaling technique and the EfficientNet model Tan & Le (2019), the ESPNet model uniformly scales the model along the depth, width, and input resolution, thereby generating a family of scene parsing models. The study also demonstrates that by incorporating lateral connections from the early stages of the encoder and employing a simple feature fusion technique, it is possible to restrict the model depth and width while achieving better accuracy with a smaller backbone size compared to a larger one. The final ESPNet model yields competitive results on a public benchmark.

2.1.2 Approach

Due to the growing demand for developing autonomous applications, real-time scene parsing has become a key research area in machine learning. Most existing semantic segmentation models utilize either VGG-16 Simonyan & Zisserman (2014), ResNet deep variants Targ et al. (2016), or Xception Chollet (2017) neural networks as encoders. These deep convolutional neural networks (DCNNs) produce impressive semantic results and are well-suited for offline computation.

However, for real-time computation, a mobile-friendly DCNN model is required.

The EfficientNet model Tan & Le (2019) is a DCNN network constructed by leveraging the residual bottleneck blocks of MobileNet Sandler et al. (2018). These blocks have an optimized architecture that makes them more resource-friendly compared to other residual blocks. Inspired by this efficient design, this paper chooses to utilize EfficientNet as the feature extractor of the model.

2.1.3 Methodology and findings

The two main objectives of this paper are as follows: 1) uniformly scaling the model along all dimensions to improve the accuracy of the model, and 2) designing an efficient segmentation model for resource-constrained mobile devices. To achieve the first goal, this study employs a compound scaling method that uniformly scales the network's width, depth, and input resolution. The following equations illustrate the technique, where α, β, and γ are constants determined through a small grid search, and ϕ is a user-specified coefficient that determines the depth, width, and input resolution of the model.

$$depth : d = \alpha^{\phi}, \tag{2.1}$$

$$width : w = \beta^{\phi}, \tag{2.2}$$

$$inputresolution : r = \gamma^{\phi}, \tag{2.3}$$

$$\alpha \times \beta^2 \times \gamma^2 \approx 2, where \alpha \geq 1, \beta \geq 1, \gamma \geq 1 \tag{2.4}$$

Based on this technique, a family of ESPNet models is developed. Table 2.1 illustrates the structure of the ESPNet family. As observed, with the increase in the depth and width coefficients, the model size expands, and the parameters of the ESPNet-S0 model reach 11.6 million (M). To ensure a compact and practical model design for mobile devices, this paper introduces only three extended versions of the segmentation model. For better feature refinement of the top global feature map, The proposed model deploys a feature pooling module on top of the encoder.

Table 2.1: The family of ESPNet

Model	Input size	Width coefficient	depth coefficient	Parameter (M)
ESPNet-S0	512 × 512	1	1	7.6
ESPNet-S1	640 × 640	1	1.1	10.1
ESPNet-S2	768 × 768	1.1	1.2	11.6

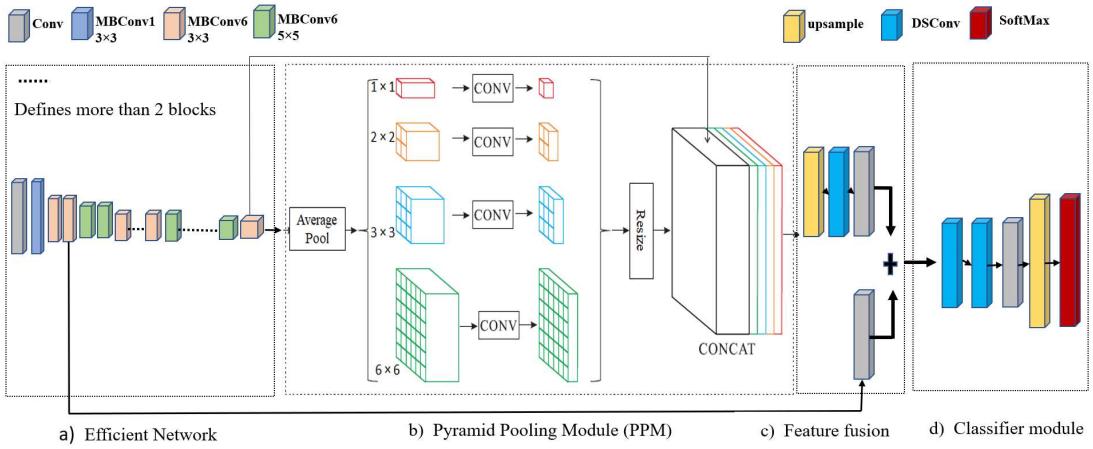

Figure 2.1: Complete architecture of the final ESPNet (this figure is derived from the literature Singha et al. (2020b).)

To accomplish the second objective, this paper demonstrates the effectiveness of incorporating a lateral connection from the early stage of the encoder to the decoder and fusing the shallow feature with the refined global feature map to enhance semantic representation. Building upon this idea, the base ESPNet-S0 achieves superior semantic results compared to ESPNet-S2. Consequently, the final ESPNet model is introduced. The complete architecture of the final ESPNet model is depicted in Figure 2.1.

The key findings of this study are:

- Scaling uniformly along all dimensions is required for better semantic results.
- Lateral connections from the encoder to the decoder enhance model performance.
- Fusing global and shallow features at the decoder improves the semantic

representation.

- Deploying feature scaling techniques on top of the decoder provides better coarse-to-fine refinement of the global feature map.

2.1.4 Contributions

The key contributions of this paper are:

- Relying on the compound scaling technique, the paper presented a family of semantic segmentation models.
- A pyramid pooling module is utilized for filtering the global feature map at different scales, which helps improve the model's performance.
- A simple yet effective feature fusion and classifier unit are introduced. To reduce model parameters and FLOPs, a depth-wise separable convolution (DSConv) layer is utilized in the feature fusion and classifier unit.
- To restrict the depth and width of the model, a lateral connection is introduced, which effectively improves model performance. Based on this, the paper presented the final ESPNet, which has 7.6 million parameters and 6.5 billion FLOPs at a 512×512 input resolution.
- An extensive experiment has been conducted to evaluate the model's performance on public benchmarks. The proposed final ESPNet achieves a competitive result (60.8% mIoU) on the Cityscapes Cordts et al. (2016) validation set.
- Several existing offline and real-time models are also replicated and trained under the same system configuration to ensure a fair comparison. The comparative results demonstrate that the proposed final ESPNet outperforms numerous existing models.
- An official public GitHub repository is created, and the design of the proposed model is uploaded.

2.2 Publication 2: FANet: Feature Aggregation Network for Semantic Segmentation

Bibliographic reference:

Singha, T., Pham, DS., Krishna, A., **"FANet: Feature Aggregation Network for Semantic Segmentation,"** In: Proc. DICTA, Melbourne, Australia, 2020, pp. 1-8, doi: 10.1109/DICTA51227.2020.9363370.

2.2.1 Abstract

Existing semantic segmentation models heavily rely on deep convolutional neural networks (DCNNs), resulting in large overall model architectures. However, such large networks are not suitable for real-time computation, posing a challenge for resource-constrained mobile devices. Additionally, the lack of control over the architectural design of a pre-trained backbone network makes it difficult to optimize the entire pipeline of a semantic segmentation model.

To address these issues and achieve real-time semantic performance, this paper introduces a lightweight semantic segmentation model specifically designed for resource-constrained mobile devices. The proposed model consists of only 1.1 million parameters and 5.8G floating-point operations (FLOPs), enabling it to handle high-resolution input images in real-time environments. The model design leverages the residual blocks of MobileNetV2, which provides an efficient backbone for feature extraction. To enhance precise object localization, a multi-scale feature fusion module is introduced on top of the backbone. This module processes global features through multiple paths, ensuring accurate object localization. The performance of the proposed model is evaluated using two publicly available benchmarks, and the results clearly demonstrate the superior performance of the proposed model compared to many existing models.

2.2.2 Approach

Model optimization and accuracy are crucial factors for achieving real-time semantic performance. While existing semantic segmentation models often achieve high accuracy, their efficiency in real-time environments is often poor. For example, the ESPNet model produces competitive results on the Cityscapes dataset Cordts et al. (2016), but its efficiency is compromised due to its 7.6 million parameters. Additionally, these models heavily rely on existing deep convolutional neural network (DCNN) models.

To address these challenges, this paper adopts a different approach by designing an optimized encoder from scratch to build a lightweight backbone. By utilizing inverted residual bottleneck blocks from MobileNetV2 Sandler et al. (2018), this paper introduces a lightweight backbone with less than 1 million parameters.

To improve context assimilation and object localization, this paper explores different existing feature fusion techniques and aims to design an optimized and efficient technique.

2.2.3 Methodology and findings

The **encoder** of the proposed FANet model is designed by utilizing mobile residual blocks (MBConv). The encoder consists of nine blocks with an expansion ratio of six. These blocks first expand the channels of the input feature map by the expansion ratio and then squeeze it back to the original input size. This squeeze and excitation technique enhances the scale of view along the channels and improves the model's performance.

The paper also introduces a down-sampling technique at the early stage of the encoder to reduce the spatial dimensions of the input image by a factor of 2^3 before passing it to the MBConv block. This downsampling operation helps in reducing the spatial dimensions, resulting in fewer parameters and floating-point operations (FLOPs) for the MBConv blocks. To achieve this, one standard

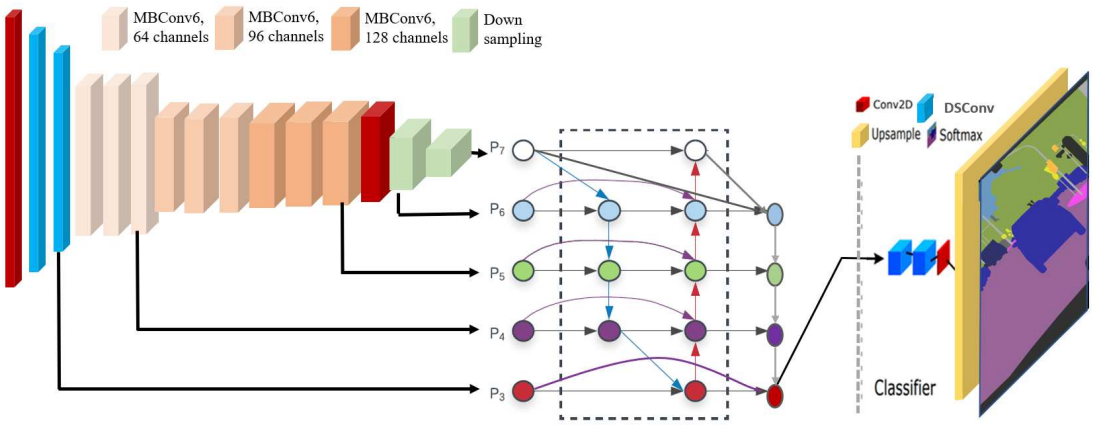

Figure 2.2: Complete architecture of FANet (this figure is derived from the literature Singha et al. (2020b).)

convolution (Conv) layer and two depth-wise separable convolution (DSConv) layers are employed. The DSConv layer generates fewer parameters and FLOPs compared to the standard Conv layer.

The **decoder** design of the proposed FANet is both simple and effective. It consists of two parts: the feature fusion module and the classifier. Taking inspiration from the design of the bi-directional feature pyramid network (Bi-FPN), a modified version of Bi-FPN is introduced by incorporating an additional top-down path and a few skip connections. The design of this modified Bi-FPN is illustrated in Figure 2.2.

In the proposed decoder, the extracted features from different stages of the encoder are passed through multiple paths. This process enhances the entire feature hierarchy by propagating accurate localization signals in both directions. The inclusion of skip connections also helps in preserving semantic details. The final top-down path assimilates contextual details and generates a rich semantic map, which is then passed through the classifier head to produce the final output.

The key findings of this study are:

- Building an encoder is necessary for an optimized real-time semantic segmentation model.

- Down-sampling at early stages of the encoder reduces model parameters and FLOPs.
- Deploying the DSConv layer in the down-sample and classifier module reduces model parameters and FLOPs.
- Multi-scale feature fusion enhances the entire feature hierarchy.
- Adding a few layers in the classifier is useful for better refinement of the final feature map.

2.2.4 Contributions

The key contributions of this paper are:

- A lightweight and efficient semantic backbone is proposed by relying on MBConv blocks of MobileNet Sandler et al. (2018).
- An early-stage down-sampling technique is introduced in the encoder to optimize computational overhead.
- DSConv is utilized instead of standard Conv at various stages of the pipeline to reduce model parameters and FLOPs.
- A modified design of Bi-FPN is introduced, improving the entire feature hierarchy through accurate object localization and rich context assimilation.
- The proposed feature aggregation network (FANet) has only 1.1 million parameters and 5.8G FLOPs.
- The performance of the model is evaluated on two publicly available benchmarks: Cityscapes Cordts et al. (2016) and CamVid Brostow et al. (2009). FANet achieves 65.9% and 57.8% class mIoU on Cityscapes and CamVid validation sets, respectively.

- Several existing offline and real-time models are reproduced and trained under the same system configuration for a fair comparison. The comparative results demonstrate that FANet outperforms many existing models.

- An official public GitHub repository is created, and the project details, including the design of the proposed model, are uploaded.

2.3 Publication 3: A Lightweight Multi-scale Feature Fusion Network for Real-Time Semantic Segmentation

Bibliographic reference:

Singha, T., Pham, DS., Krishna, A., Gedeon, T. (2021). **"A Lightweight Multi-scale Feature Fusion Network for Real-Time Semantic Segmentation"**. In: Proc. ICONIP 2021. vol 13109. Springer, Cham. https://doi.org/10.1007/978-3-030-92270-2-17.

2.3.1 Abstract

Relying on existing Deep Convolutional Neural Networks (DCNNs), the semantic segmentation model retrieves local and deep feature maps and simply passes them through a series of upsample layers at the decoder to generate the segmented output. However, this approach causes a boundary degeneration effect in the output. Moreover, the large semantic gap among the feature maps introduces a noisy effect in the predictions. To overcome these issues, the journal focuses on designing efficient feature scaling and feature fusion modules, which provide a better dense pixel-level representation and improved region identification by alleviating the gap among the feature maps.

Building upon a new feature scaling and feature fusion technique, the thesis proposes a novel architecture called Feature Scaling Feature Fusion Network (FSFFNet), which achieves a 71.8% call mIoU on the Cityscapes validation set while having only 1.3M parameters.

2.3.2 Approach

The primary objective of this study is to improve model accuracy without sacrificing the model's efficiency. To achieve this, the study decided to design the encoder from scratch. Mobile residual blocks from MobileNet Sandler et al. (2018) were chosen as they generate fewer parameters and FLOPs, making them suitable for the encoder design. Additionally, a feature scaling and feature fusion technique is deployed on top of the encoder to enhance coarse-to-fine refinement and improve semantic representation. Finally, a simple classifier is employed to label each pixel based on its softmax score.

2.3.3 Methodology and findings

The complete architecture of the model is exhibited in Figure 2.3. It shows that the whole pipeline has three main parts: the encoder, intermediate stage, and decoder. The encoder consists of one 3×3 Convolution layer and 14 mobile inverted residual blocks (MBConv). Two different types of MBConv blocks are used: MBConv1, which refers to a residual block with an expansion ratio of 1, and MBConv6, which defines a residual block with a 6 expansion ratio. When the input is received by the MBConv block, it first expands the channel of the input by the expansion ratio and then squeezes the channel, effectively filtering out noise from the input feature map. Except for the first block, all MBConv blocks have an expansion ratio of 6.

The intermediate stage demonstrates the multi-scaling technique, which is crucial for complex scene analysis. It consists of one feature scaling module (FSM) and two pooling layers. Motivated by the design of the Atrous Spatial Pyramid Pooling technique (ASPP) Chen et al. (2018), this paper proposes an

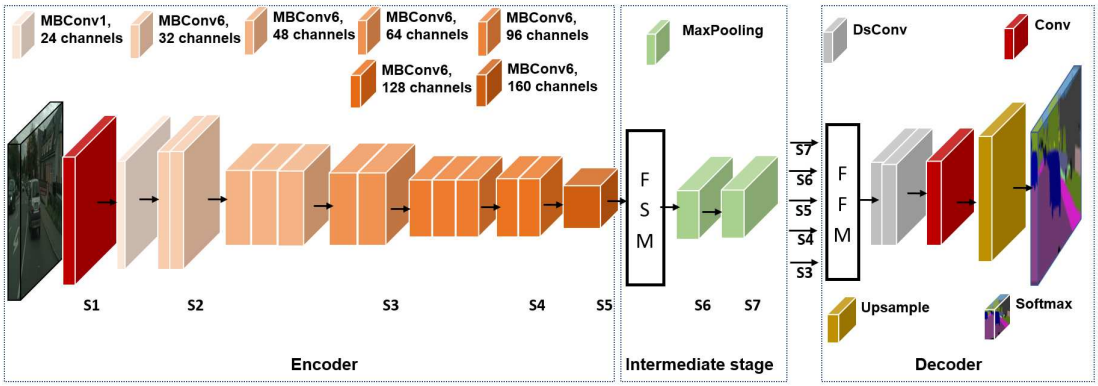

Figure 2.3: Complete architecture of FSFFNet (this figure is derived from the literature Singha et al. (2021c).)

optimized FSM with the following changes:

- The standard convolution (Conv) layers of ASPP are replaced by depth-wise separable convolution (DSConv) layers to reduce memory usage. DSConv produces three times fewer parameters and FLOPs compared to a Conv layer.

- Instead of five branches, the proposed FSM uses four parallel branches: three dilated DSConv branches and one image pooling branch.

- Based on the ablation study, the best dilation rates of (8, 16, 24) are used for an input size of 1024×2048, whereas ASPP uses (6, 12, 18, 24).

- The number of channels inside the FSM is reduced to one-fourth of the channels in the input tensor, reducing the model's parameters and making it more efficient.

The two pooling layers are used to down-sample the spatial dimensions of the feature map without introducing any parameters. Thus, the proposed model creates seven stages in the entire feature hierarchy.

In the decoder, a multi-scale feature fusion module (FFM) is deployed, which takes five feature maps from the last five stages of the encoder and fuses them together. The final feature map from the FSM is successively fused with the

shallow feature maps at the second and first stages of the encoder. This allows for the aggregation of deep, intermediate, and shallow features for better contextualization.

The key findings of this study are:

- At the initial stage, down-sampling can be done using MBConv1 as it does not contribute too many parameters due to a lower number of channels and an expansion ratio of 1.
- Dilated separable convolution is more efficient than standard convolution.
- In the multi-feature scaling, higher dilation rates provide a better receptive field for capturing objects in complex scenes.
- The choice of dilation rates in the feature scaling module depends on the model size and input resolution. In this study, dilation rates of 8, 16, 24 produced better results for an input resolution of 1024×2048.
- For higher-resolution input images, such as 1024×2048, the model can down-sample the input image up to 2^5 to 2^7 times. Additional stages can be created using pooling operations instead of deploying MBConv blocks. Pooling operations do not contribute any parameters. These additional stages create a very low-resolution output with rich contextual details. However, there is a risk of losing spatial details. To address this issue, an extensive feature fusion technique is required.

2.3.4 Contributions

The key contributions of this paper are:

- A simple lightweight backbone is designed by staging 14 MBConv blocks.
- An intermediate stage is introduced on top of the encoder, adding two additional stages to the entire feature hierarchy.

- An optimized and efficient FSM module is introduced, inspired by ASPP.

- A multi-scale FFM is introduced, motivated by the feature fusion technique in FANet Singha et al. (2020a), which aggregates deep, intermediate, and shallow features for improved semantic representation.

- The model's performance is evaluated on two publicly available benchmarks: Cityscapes Cordts et al. (2016) and BDD100K Yu et al. (2020). BDD100K is considered the most challenging dataset due to its diverse nature.

- The proposed model achieves 69.4% and 71.8% class mIoU on the Cityscapes test and validation sets, respectively.

- The proposed model achieves state-of-the-art performance on the BDD100K validation set (55.2%) among real-time models.

- An official public GitHub repository is created, and the project details, including the design of the proposed model, are uploaded.

2.4 Publication 4: SCMNet: Shared Context Mining Network for Real-time Semantic Segmentation

Bibliographic reference:

Singha, T., Bergemann, M., Pham, DS., Krishna, A., **"SCMNet: Shared Context Mining Network for Real-time Semantic Segmentation,"** In: Proc. DICTA, Gold Coast, Australia, 2021, pp. 1-8, doi: 10.1109/DICTA52665.2021.9647401.

2.4.1 Abstract

Different backbone architectures, such as one-branch, multi-branch, and two-branch designs, have been adopted in designing segmentation models. The multi-branch design is particularly introduced to reduce computer overhead and make the model suitable for real-time computation while improving performance. In the multi-branch and two-branch approaches, there is a dedicated deep branch and one or more shallow branches. However, despite fusing shallow and deep features from their respective branches at the decoder side, a boundary degeneration effect can be observed in the output, as all these branches learn independently throughout the training process.

To address this issue, this paper introduces a novel design called knowledge shared two-branch design for the encoder. Firstly, a shallow branch, parallel to the deep branch, is introduced on the encoder side. Then, both branches share their learning at different stages of the encoder. After each sharing point, a context mining module (CMM) is introduced for better coarse-to-fine refinement of the shared feature map. This improves the entire feature hierarchy in both branches, which is then passed to the decoder for better semantic representation. A deep shallow feature fusion module (DS-FFM) is deployed on top of the shared two-branch encoder for precise object localization. Relying on this novel shared branches design, the proposed model achieves state-of-the-art performance on the CamVid dataset while having only 1.2 million parameters.

2.4.2 Approach

To improve the entire feature hierarchy in the encoder, this paper studies different network architectures used for designing a lightweight encoder. In order to optimize computer overhead without sacrificing the model's performance, a multi-branch design is introduced. In this design, a dedicated deep branch is utilized to extract contextual details from low-resolution input images. Keeping the low resolution input at this branch helps reduce overhead. However, relying

solely on the deep branch for model performance would be insufficient due to the reduced input resolution. Therefore, multiple shallow branches are introduced, each taking inputs at different higher resolutions to capture boundary and texture details at different scales. Despite the shallow design of the multiple branches, the overall model still produces a large number of parameters and FLOPs, resulting in a large memory footprint.

Later, the multi-branch design is replaced with a two-branch design to improve model efficiency. However, a drop in model accuracy is observed, which could be attributed to the independent design of the deep and shallow branches in the encoder. If both branches share their knowledge at different stages in the encoder, it can enhance the entire coarse-to-fine refinement process. Knowledge from the other branch can serve as additional information for each branch. For example, the model often struggles to assign the correct class to pixels belonging to the boundaries of objects in the scene. Therefore, if the shallow branch shares shallow features with the deep branch, the semantic representation of the deep feature map can be improved.

Based on this hypothesis, this paper adopts a shared two-branch design approach for the backbone.

2.4.3 Methodology and findings

The design of the shared two-branch encoder can be seen in Figure 2.4. The deep branch of the proposed backbone is designed by utilizing one standard convolution (Conv) layer and 11 mobile inverted residual (MBConv) blocks. In contrast, the shallow branch has a very shallow design, consisting of one Conv layer and 4 depth-wise separable convolution (DSConv) layers. Each layer down-samples the input by a factor of 2, resulting in 5 stages to maintain similar spatial dimensions between the two channels. Since the input size of the deep branch is half that of the shallow branch, the deep branch creates 4 stages in the entire feature hierarchy.

Figure 2.4 illustrates that features from both branches are shared at three

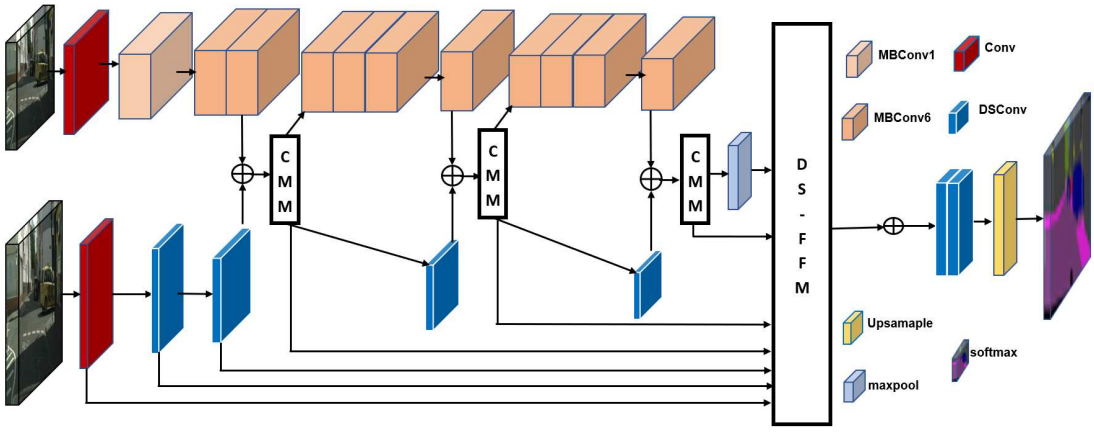

Figure 2.4: Complete architecture of SCMNet (this figure is derived from the literature Singha et al. (2021a).)

stages and pass through the context mining module (CMM). The shared knowledge can introduce noise and negatively influence the performance of the other branch. Therefore, deploying the CMM is necessary for better refinement of the shared feature map. The CMM takes the shared feature map and processes it through four parallel branches. The output of each branch is then concatenated for coarse-to-fine refinement. The CMM design includes one point-wise Conv, one dilated DSConv, and two image pooling branches with different pool sizes.

At the decoder side, a deep shallow feature fusion module is introduced, which takes 3 deep, 1 intermediate, and 3 shallow features from the encoder. Deep features contain rich contextual details of the entire objects, while shallow features capture boundary and texture details. The intermediate feature helps bridge the gap between deep and shallow features by fusing the top-down and bottom-up signals at the intermediate stage of the deep shallow feature fusion module (DS-FFM).

The key findings of this study are as follows:

- Knowledge sharing between the deep and shallow branches improves the entire feature hierarchy in both branches.

- Shared knowledge can introduce some noise in both branches, which can be addressed by providing a better coarse-to-fine refinement process.

- The depth of the deep branch can be reduced due to the presence of the shallow branch and shared feature maps.
- Extensive feature fusion among the shallow and deep feature maps is required to reduce the semantic gaps between deep and shallow feature maps.
- The size of the input image has a significant influence on the model's performance.

2.4.4 Contributions

The key contributions of this paper are as follows:

- Introduction of a novel shared two-branch backbone, where both branches share their knowledge for improved semantic representation.
- Deployment of a new module called the context mining module to filter out noise from the shared feature maps.
- Optimization of the design of the deep branch to reduce computational overhead, leveraging the presence of the shallow branch and shared feature map.
- Introduction of a deep shallow feature fusion module at the decoder, providing guidance for fusion and enabling precise object localization.
- Introduction of an efficient semantic segmentation model called the Shared Context Mining Network (SCMNet) with 1.2 million parameters.
- Evaluation of the model's performance on three publicly available benchmarks: Cityscapes Cordts et al. (2016), BDD100K Yu et al. (2020), and CamVid Brostow et al. (2009).
- State-of-the-art performance achieved by SCMNet on CamVid validation (78.6%) and test (71.3%) sets among existing real-time models.

- Creation of an official public GitHub repository and uploading of project details, including the design of the proposed model.

2.5 Publication 5: Urban Street Scene Analysis Using Lightweight Multi-level Multi-path Feature Aggregation Network

Bibliographic reference:

Singha, T., Pham, DS., Krishna, A.. **"Urban Street Scene Analysis Using Lightweight Multi-level Multi-path Feature Aggregation Network"**. Multiagent and Grid Systems, vol. 17, no. 3, pp. 249-271, 2021. DOI: 10.3233/MGS-210353

2.5.1 Abstract

Urban street scene analysis is an important problem in computer vision due to its complexity. The presence of numerous small objects such as traffic lights, traffic signs, poles, and distant objects makes it challenging for a Deep Convolutional Network (DCNN) to accurately identify and localize these objects in the output. Often, the model overlooks small and distant objects due to the presence of larger objects in the scene. To address this issue, this paper proposes a novel model called M2FANet, which incorporates a lightweight backbone, a neck section similar to an object detection model, and a segmentation head for pixel classification.

The neck section is primarily responsible for fusing or integrating features from different levels of the backbone. The proposed neck aims to enhance the representational power of the model by combining features with varying levels of spatial information and abstraction. It bridges the gap between low-level and

high-level features from different resolutions, enabling the model to capture both fine-grained details and high-level semantics, which are crucial for accurate object detection. This paper refers to this method as Multi-level Multi-path Feature Fusion (M2-FF).

Utilizing the M2-FF method, the proposed M2FANet model achieves a class mIoU of 68.3% on the Cityscapes test set while utilizing only 1.3 million parameters.

2.5.2 Approach

Traditionally, an encoder-decoder design is used for semantic segmentation, as can be seen in Figure 2.5(a). The encoder extracts rich contextual details, and the decoder deconvolves the extracted features through a series of convolution and upsample layers. In addition to this, a multi-branch and feature reuse design is often employed for designing the backbone.

In object detection, a neck section is utilized between the backbone and the detection head to bridge the gap between low-level and high-level features of the encoder. There are several existing neck architectures such as FPN Lin et al. (2017b) and PAN Liu et al. (2018). Motivated by the neck architecture used in object detection, this paper firstly introduces this neck architecture in semantic segmentation and proposes an optimized design neck architecture called M2-FF. The layout of M2-FF can be seen in Figure 2.5. This paper adopt one-branch approach to design the backbone architecture.

2.5.3 Methodology and findings

The complete architecture of the proposed model M2FANet consists of four parts: the encoder, the neck section, multi-feature scaling, and the segmentation head, as depicted in Figure 2.6. The encoder comprises a 3×3 Convolution layer followed by 15 residual blocks (MBConv) from MobileNet Sandler et al. (2018). Since the aim of this paper is to design a real-time segmentation model capable of handling high-resolution input images, the proposed backbone has an output

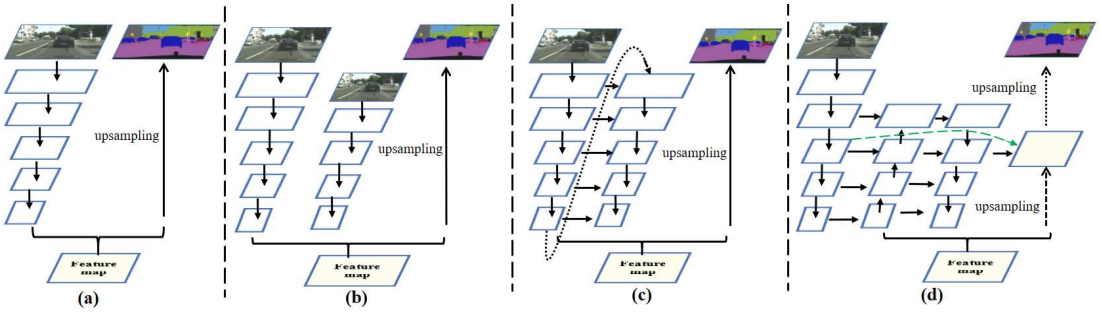

Figure 2.5: Different approaches. From left to right: (a) One-branch encoder, (b) Multi-branch encoder, (c) Feature reuse in sub-encoder (d) M2-FF. (this figure is derived from the literature Singha et al. (2021b).)

stride of 2^7. This means that for an input image size of 1024×2048, the output tensor at the top of the encoder will have a size of $8 \times 16 \times 128$. The maximum number of channels at the end of the encoder is 128.

To address semantic gaps among the feature maps at different stages of the encoder, a neck section is employed. The design of the neck section is inspired by PAN Liu et al. (2018) and can be observed in Figure 2.6. It is referred to as the Multi-level Multi-path Feature Fusion (M2-FF) method as it aggregates or fuses feature maps from different resolutions. It utilizes two top-down and one bottom-up paths to capture both fine-grained details and high-level semantics, which are crucial for precise object localization. To mitigate the issue of gradient vanishing, a lateral connection is established at each level from the encoder to the last path of the M2-FF.

This paper adopts the Atrous Spatial Pyramid Pooling (ASPP) Chen et al. (2018) with several modifications on top of the neck architecture. The following modifications are made:

- Instead of five branches, four branches are used.
- Dilated separable branches (DSConv) are used instead of dilated convolution branches to reduce model parameters.
- The point-wise convolution branch is employed to reduce the number of channels in the input feature map by one-fourth. This reduced number of

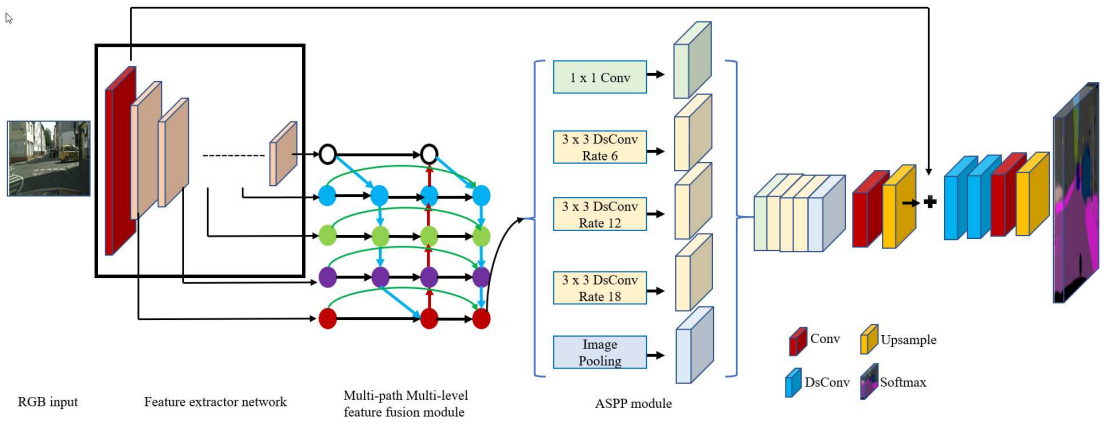

Figure 2.6: Complete architecture of M2FANet (this figure is derived from the literature Singha et al. (2021b)).

channels is utilized across all branches, resulting in a significant reduction in parameters and floating-point operations (FLOPs).

- The projection layer after the concatenation operation is removed.

Thus, with the help of the feature scaling method, the noisy pixels in the rich semantic feature map from the M2-FA module are removed. Generally, the feature scaling technique is applied immediately after the encoder to refine the deep global feature map at low spatial dimensions. However, in this paper, it is deployed after the neck section, where the global and local features are already aggregated, and the final feature map after M2-FF attains a higher spatial resolution. Consequently, a higher dilation rate can be used in the dilated branches of the feature scaling methods. The output of the feature scaling module is then bilinearly upsampled and fused with the shallow feature before being passed to the segmentation head.

The paper follows a simple design for the segmentation head, which consists of two separable convolutions, one standard convolution, one upsampling, and one softmax layer.

The key findings of this study are as follows:

- For a semantic segmentation model, having a maximum of 128 channels at

the end of the encoder is sufficient for capturing semantic details, especially when the model is built using MBConv blocks.

- The utilization of a neck architecture is beneficial for bridging the gap between low-level and high-level features at different scales from the encoder.
- The inclusion of lateral connections at each level of M2-FF effectively addresses the issue of gradient vanishing.
- In the feature scaling technique, deploying a higher dilation rate in branches that operate on feature maps with higher spatial dimensions can be advantageous for achieving better field-of-views.
- When employing multiple parallel branches in the feature scaling technique for filtering the input, reducing the number of channels in each branch can improve efficiency without sacrificing accuracy, as long as the outputs of each branch are concatenated at the end.

2.5.4 Contributions

The key contributions of this paper are as follows:

- Introducing a lightweight backbone design based on a single branch approach.
- Introducing a novel neck architecture called M2-FF, inspired by the neck architecture used in object detection models. M2-FF effectively captures both fine-grained details and high-level semantics from the feature maps.
- Optimizing the design of the ASPP feature scaling technique and applying it on top of the neck architecture to filter out noise.
- Proposing a lightweight segmentation model called M2FANet, capable of efficiently handling high-resolution input images in real-time environments, with a parameter count of 1.3 million.

- Evaluating the proposed M2FANet using two publicly available benchmarks: Cityscapes Cordts et al. (2016) and CamVid Brostow et al. (2009). The results demonstrate competitive performance compared to existing real-time semantic segmentation models with less than 2 million parameters.

2.6 Publication 6: SC-CrackSeg: A Real-time Shared Feature Pyramid Network for Crack Detection and Segmentation

Bibliographic reference:

Singha, T., Bergemann, M., Pham, DS., Krishna, A., **"SC-CrackSeg: A Real-time Shared Feature Pyramid Network for Crack Detection and Segmentation,"** In: Proc. DICTA, Sydney, Australia, 2022, pp. 1-8, doi: 10.1109/DICTA56598.2022.10034629.

2.6.1 Abstract

Regular monitoring of the condition of existing buildings, roads, and pavements is necessary for maintenance processes. Specifically, detecting structural cracks is crucial for avoiding significant damage and providing a safer environment. However, due to a lack of human and other resources, regular monitoring becomes challenging. Therefore, there is a need for an automated monitoring system. With advancements in computer vision, structural crack detection and segmentation can be automated using machine learning techniques. Since such applications run in real-time environments without sophisticated hardware support, designing an efficient and resource-friendly crack segmentation model is a challenging task. Several existing models have been designed for crack detection; however,

their large backbones result in poor efficiency in real-time environments. To achieve both efficiency and improved detection accuracy, this paper introduces a lightweight crack detection and segmentation model called SC-CrackSeg, capable of processing over 200 frames per second at an input resolution of 448×448. The proposed model's performance is evaluated on different crack datasets, and the results demonstrate that SC-CrackSeg achieves better accuracy and efficiency compared to existing models.

2.6.2 Approach

There are several existing deep learning methods for detecting structural cracks. Similar to object detection, these methods can detect the presence of cracks in an input image and identify them using bounding boxes in the output. However, these techniques do not provide information about the thickness of the cracks. Semantic segmentation can be used to address this limitation. Segmentation not only detects cracks but also segments them in the scene, allowing for a more detailed analysis. Therefore, based on the segmented output, an automated application can be developed to raise a flag when the thickness of the segmented crack exceeds a certain threshold value. Building on this idea, this paper aims to develop a crack segmentation model suitable for real-time applications.

Inspired by the concept of knowledge sharing between the deep and shallow branches introduced in Singha et al. (2021a), this paper deploys a shared-branch encoder to extract spatial and contextual details of the cracks from the scene.

2.6.3 Methodology and findings

The complete pipeline of the proposed model is displayed in Figure 2.7. Motivated by the shared two-branch backbone of SCMNet Singha et al. (2021a), this paper introduces a similar shared backbone design with the following modifications:

- Instead of taking inputs of two different resolutions at the encoder, the proposed model's encoder accepts a single-resolution input image. The shallow branch is created after the first stage of the encoder.

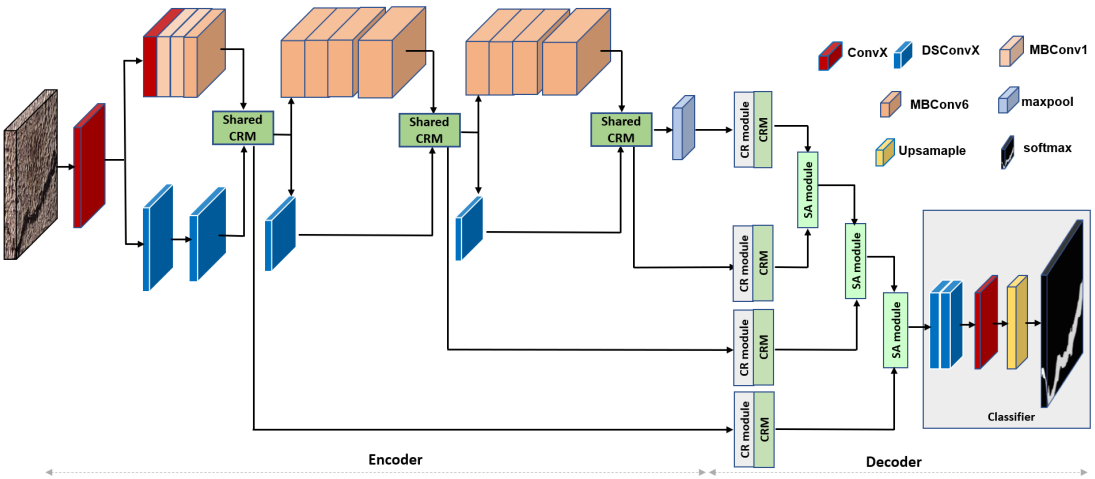

Figure 2.7: Complete architecture of SC-CrackSeg (this figure is derived from the literature Singha et al. (2022).)

- The context mining module after each shared point is replaced by a shared context refinement module (CRM).

The shared CRM first takes the shallow and deep feature maps from their respective branches and fuses them together. It then passes the shared feature map through four parallel branches: one point-wise convolution and three dilated separable convolution branches with higher dilation rates. Since the image pooling layers in CMM Singha et al. (2021a) do not have any learnable parameters, they are replaced by dilated branches. Once the filtering through the four branches is complete, the output of each branch is concatenated together to capture fine semantic details. To avoid gradient vanishing issues, a skip connection is also introduced at the end. As a result, the shared CRM provides a better coarse-to-fine refinement process.

At the decoder end, a semantic aggregation (SA) module is introduced, which aggregates the shared feature maps at various stages of the encoder. Before aggregating the shared feature maps, they pass through a channel reduction (CR) module and a CRM module. The CR module ensures that the feature maps have the same channel dimension, while the SA module ensures they have the same spatial dimensions before being fused together. Finally, the fused feature map

passes through the classifier, generating the segmented crack in the output.

The key findings of this study are:

- Segmenting cracks provides more contextual details compared to simply detecting them.
- Although there are only two classes (crack and background), the pixels near the crack have different class IDs, resulting in noisy pixels that can negatively impact the model's performance.
- Avoiding noisy pixels improves the model's performance.
- The number of pixels belonging to the crack class is much fewer than the background class pixels, leading to a class imbalance problem in the dataset.
- Since the dataset combines 12 different crack datasets, the annotations of cracks are not uniform across all datasets, which can influence the model's performance.
- Models generally struggle to segment very thin cracks, which is a common phenomenon observed across all models.

2.6.4 Contributions

The key contributions of this paper are:

- Introducing a new and efficient structural crack segmentation architecture called SC-CrackSeg.
- Deploying several techniques to improve efficiency during training and inference time, such as channel reduction, supplying a single resolution input image to the encoder, and creating a shallow branch at the intermediate stage.
- Introducing a new module called CRM for context mining and feature scaling, aimed at keeping the design simple.

- Deploying a simple yet efficient semantic aggregation module at the decoder to reduce computer overhead and achieve better semantic representation.

- Introducing a void class to label noisy pixels, in addition to the crack and background classes. The void class assigns a label of 255 to noisy pixels, and it is not used during training.

- Exploring different loss functions, including per-pixel class-weighted loss, Focal Tsverky loss, and categorical cross-entropy.

- Measuring performance using various metrics such as mIoU, F1 score, precision, and recall to demonstrate the model's performance and versatility on publicly available combined datasets.

- Reproducing different existing crack segmentation models and training them under the same system configuration for a fair comparison.

- The proposed model achieved state-of-the-art performance (mIoU: 77.04%, F1 score: 85.34%, Precision: 88.59%, Recall: 82.32%) compared to existing models.

2.7 Publication 7: SDBNet: Lightweight Real-time Semantic Segmentation Using Short-term Dense Bottleneck

Bibliographic reference:

Singha, T., Pham, DS., Krishna, A., **"SDBNet: Lightweight Real-time Semantic Segmentation Using Short-term Dense Bottleneck,"** In: Proc. DICTA, Sydney, Australia, 2022, pp. 1-8, doi: 10.1109/DICTA56598.2022.10034634.

2.7.1 Abstract

Designing an efficient backbone for a semantic segmentation model is crucial to achieve better semantic results. Adopting a pre-trained backbone, such as a deep convolutional neural network (DCNN), designed for image classification may not produce the desired segmented result due to the lack of task-specific design. Furthermore, most existing DCNN models do not provide a sufficiently large receptive field to capture objects of varied geometrical shapes in complex scenes. It has often been observed that tiny or distant objects in a complex scene are overlooked due to the presence of larger objects. To address this issue, this paper proposes a novel architecture called Short-term Dense Bottleneck (SDB) block, which provides two to three times larger field-of-view than existing bottleneck blocks. Building upon this, an efficient real-time semantic segmentation model named SDBNet is designed to better contextualize varied geometrical objects in complex scenes. At the decoder, simple yet effective feature refinement (FR) and semantic aggregation (SA) modules are introduced to fine-tune the semantic information and accurately position objects. The proposed SDBNet is evaluated on three publicly available benchmarks, and it achieves improved results on these datasets.

2.7.2 Approach

It has been observed that existing models struggle to detect and localize tiny or distant objects in complex scenes. Due to the presence of large objects in the scene, existing models often fail to detect them. Popular deep convolutional neural networks (DCNNs) are primarily designed for image classification and may not capture all the necessary contextual details for semantic segmentation. Moreover, the residual blocks of existing DCNNs provide a limited number of field-of-views, leading to missing details from the scene. To address this limitation, this study proposes a dense bottleneck architecture that increases the number of receptive fields in the bottleneck block. This architecture enhances the model's ability to

capture more contextual information and reduces the semantic gaps among the feature maps.

2.7.3 Methodology and findings

The paper proposes two types of SDB blocks: SDB1 with a stride of 1 and SDB2 with a stride of 2. The deep branch consists of 2 Convolution blocks and 6 SDB blocks. The Convolution blocks are part of the down-sampling technique, which down-samples the input image by four and creates the first 2 stages. The remaining three stages contain all the SDB blocks, with 2 blocks in each stage. Each SDB block consists of 3 mobile inverted residual blocks (MBConv) Sandler et al. (2018) with three different expansion ratios 2, 3, 4. Therefore, each SDB block introduces three different channel dimensions to the feature map. In SDB2, the spatial dimensions of the input feature are down-sampled by 2 in the first MBConv block, then by 4 in the second MBConv block, and finally upsampled by 2 and processed by the third MBConv block, while maintaining the same spatial dimensions. This design provides a larger number of field-of-views compared to existing models. In SDB1, the spatial dimensions do not change as the feature map passes through the three MBConv blocks, but excitations and squeezes occur along the channel dimension in each MBConv block. Dense skip connections are added from the input to the output of each SDB1 block, reducing the semantic gap and addressing the gradient vanishing issue.

A shallow branch is also deployed in parallel with the deep branch to provide local spatial details of regions to the deep branch. Three ShallowX blocks are used to design the shallow branch.

At the decoder side, 3 FR modules, 2 SA modules, a series of upsampling layers, and one classifier are deployed. The FR modules refine two deep features and one shallow feature, while the SA modules aggregate feature maps from various stages. The complete pipeline of the proposed SDBNet model can be seen in Figure 2.8.

The key findings of this study are:

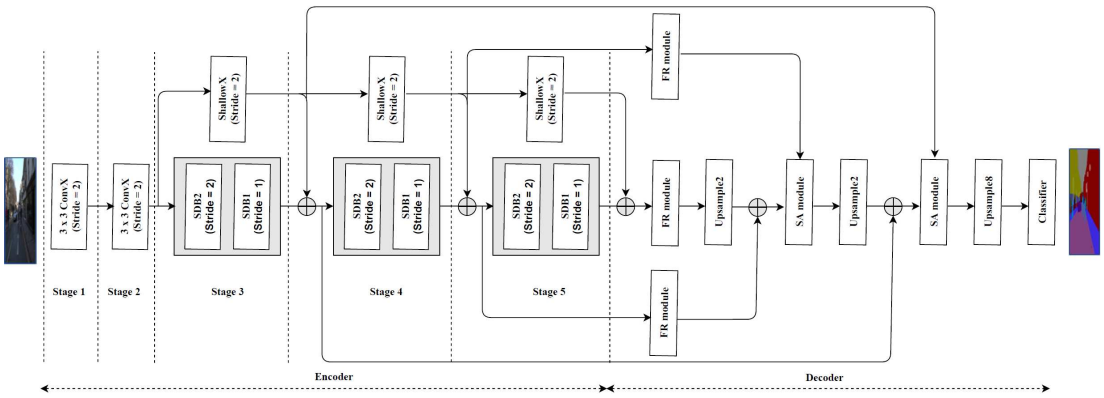

Figure 2.8: Complete architecture of SDBNet (this figure is derived from the literature Singha et al. (2022).)

- Existing models face challenges in detecting tiny or distant objects in complex scenes.
- The bottleneck architecture used in existing models provides a limited number of field-of-views, which leads to the omission of contextual details for tiny objects in complex scenes.
- The proposed dense architecture with an increased number of receptive fields can effectively capture objects with various geometrical shapes.
- Using multiple smaller expansion ratios, rather than a single specific ratio, can enhance the receptive field along the channel dimensions and reduce the model's parameters and computational requirements (FLOPs).
- Dense addition operation is found to be more efficient than dense concatenation operation.

2.7.4 Contributions

The key contributions of this paper are:

- Introduction of a novel dense bottleneck block called SDB, designed to capture objects of different shapes in complex scenes.

- Utilization of multiple expansion ratios 2,3,4 within each SDB module to improve field-of-view along the channel dimensions.
- Development of a carefully designed shallow branch that serves as an attention vector, providing local spatial guidance to the deep branch.
- Introduction of the context refinement module (CRM) for context mining and feature scaling, contributing to the simplicity of the overall design.
- Deployment of an efficient feature refinement module at the decoder to filter out noisy pixels from the semantic feature maps.
- Implementation of an optimized semantic aggregation module at the decoder, which fuses deep and shallow refined feature maps for improved semantic results.
- Evaluation of the proposed SDBNet model on various datasets, achieving performance metrics of 70.8% on Cityscapes, 73.2% on CamVid, and 51.8% on KITTI test sets. The model also achieves high processing speed, handling 98 frames per second at an input resolution of 512×1024.
- Submission and publication of the model's test results on Cityscapes and KITTI datasets on their respective servers.
- Creation of an official public GitHub repository containing the project details, including the design of the proposed model.

2.8 Publication 8: A real-time semantic segmentation model using iteratively shared features in multiple sub-encoders

Bibliographic reference:

Singha, T., Pham, DS., Krishna, A., **"A real-time semantic segmentation**

model using iteratively shared features in multiple sub-encoders", Pattern Recognition, Volume 140, 2023, 109557, ISSN 0031-3203, https://doi.org/10.1016/j.patcog.2023.109557.

2.8.1 Abstract

Gradient vanishing is a common issue in Deep Convolutional Neural Networks (DCNNs) where gradients propagated through the network during training become very small or even vanish as they pass through numerous layers. This can have a significant impact on the learning process and the overall performance of the model. While various techniques have been proposed to address this issue, such as weight initialization, normalization techniques, skip connections, and the use of activation functions with less prone vanishing gradients, the performance of the models is still affected.

One of the main reasons for this issue is the deep single-branch network design. To mitigate gradient vanishing, this paper proposes an efficient Shared Feature Reuse Segmentation (SFRSeg) model specifically designed for resource-constrained applications. The model incorporates a shared-branch design that improves the entire feature hierarchy by sharing spatial and contextual details in both deep and shallow branches. Additionally, multiple sub-encoders with a gradual increase in the number of layers in the top stages are deployed in the encoder. These sub-encoders reprocess the rich semantic feature maps, and lateral connections from the same stage of the previous sub-encoder are introduced to the successive sub-encoders. By reusing features in multiple sub-encoders and incorporating lateral connections, the proposed model mitigates the gradient vanishing issue.

Furthermore, the paper introduces a Hybrid Path Attention Semantic Aggregation (HPA-SA) on top of this novel backbone. Leveraging this design, the proposed model achieves state-of-the-art performance on the Cityscapes and CamVid datasets, surpassing existing models with less than 2 million parameters.

2.8.2 Approach

Existing semantic segmentation models often rely on popular deep convolutional neural networks (DCNNs), which have shown outstanding results in image classification tasks. However, these DCNNs may not produce optimal semantic segmentation results. One of the main reasons is that these image classification models primarily focus on capturing contextual details while often disregarding local information. However, for accurate object localization, local information plays a crucial role as it captures fine-grained details and spatial relationships between neighboring pixels. Therefore, to complement the global feature maps, this paper adopts a shared-branch encoder approach where the shallow branch serves as a source of local details.

In DCNNs, back-propagation is typically used to update the network's weights by computing gradients with respect to the loss function. However, a common issue arises when the gradients diminish significantly as they propagate backwards through a deep network. This problem is often observed in existing models due to the lack of sufficient skip connections and the reusability of feature maps in the entire encoder design. To address this issue, this paper introduces a design with multiple sub-encoders. By deploying multiple sub-encoders, the model can reprocess the feature maps, facilitating better gradient flow and preserving valuable information throughout the network.

By combining the shared-branch approach with the multiple sub-encoders design, this paper presents a novel approach called Shared Feature Reuse (SFR). This approach aims to improve the overall performance of semantic segmentation models by incorporating local information through the shared-branch encoder and mitigating gradient vanishing issues through the use of multiple sub-encoders.

2.8.3 Methodology and findings

The complete pipeline of the proposed SFRSeg model can be observed in Figure 2.9. The diagram illustrates that the first sub-encoder consists of both a deep

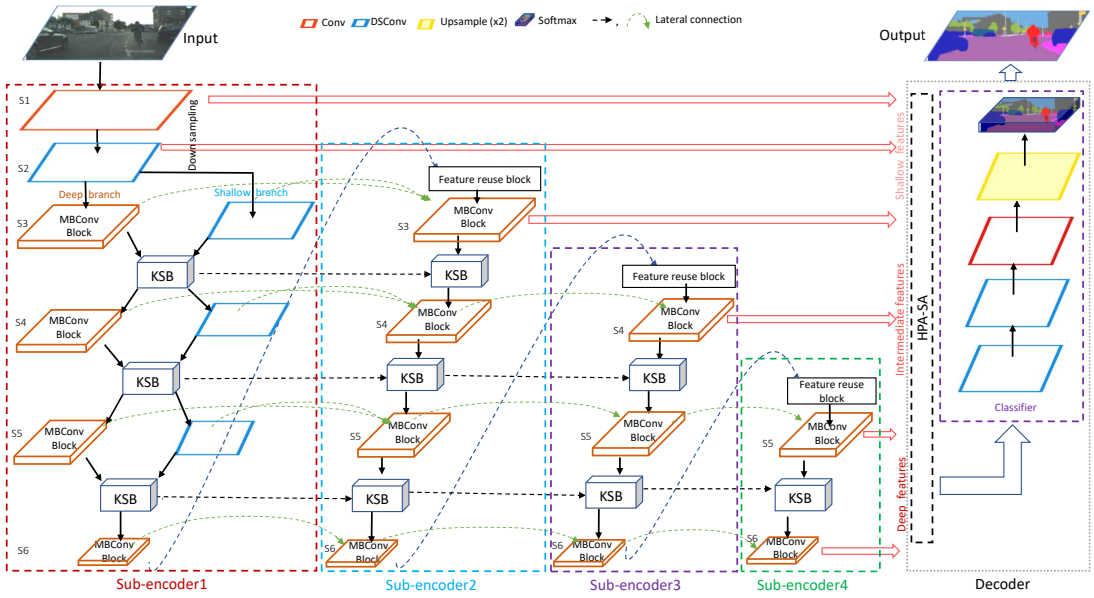

Figure 2.9: Complete architecture of SFRSeg (this figure is derived from the literature Singha et al. (2023a).)

and a shallow branch, which share their knowledge through a Knowledge Sharing Block (KSB). The deep branch in each sub-encoder is designed using staging mobile residual (MBConv) blocks Sandler et al. (2018). Notably, the entire encoder incorporates only one shallow branch, which is deployed in the first sub-encoder and comprises three separable convolution layers.

To enhance the context mining process and filter out noise from the shared feature maps, a new design is introduced within each KSB. This context mining design is inspired by DenseASPP Yang et al. (2018) and is referred to as the Cascading Context Mining (CCM) module, given the cascading architecture. It is worth noting that the proposed CCM module is significantly smaller in size compared to ASPP Chen et al. (2018) and DenseASPP Yang et al. (2018), being 22 and 130 times smaller, respectively.

To overcome the issue of gradient vanishing in the first sub-encoder, the number of layers in the top stages is reduced. The first sub-encoder consists of 6 stages with fewer layers. The output of the first sub-encoder is then passed to the second sub-encoder, which starts at the third stage and processes the input

feature map through subsequent stages. To mitigate the gradient vanishing problem, skip connections are deployed from the same stages of the first sub-encoder to the second sub-encoder.

Similarly, the third and fourth sub-encoders are designed, and the output of the previous sub-encoder is processed by the next sub-encoder. This overall encoder design effectively addresses several existing issues, particularly the gradient vanishing problem.

The key findings of this study are as follows:

- The deep network architecture of existing models leads to the issue of gradient vanishing.
- Reusing rich semantic features in multiple sub-encoders improves the model's accuracy.
- Filtering semantic feature maps through multiple sub-encoders reduces the problem of gradient vanishing.
- Lateral connections from the same stages of the previous sub-encoder to the successive sub-encoder also mitigate the issue of gradient vanishing.
- A sequential and cascaded design for context mining is more effective than a parallel-branch design.
- Boundary degeneration can be observed in the feature map as the output of the previous sub-encoder is scaled up by a large scale, but this can be addressed by a series of layers.

2.8.4 Contributions

The key contributions of this paper are:

- Introducing a novel architecture called Shared Feature Reuse (SFR) to address the gradient vanishing problem in the encoder design.

- Introducing Cascading Context Mining (CCM) as a more efficient method for filtering out noise from shared feature maps compared to existing mining methods.

- Carefully designing each sub-encoder to overcome the gradient vanishing issue without compromising model performance. Deep features are processed more times than shallow feature maps to capture more contextual details.

- Introducing a Feature Reuse Block (FRB) to address the boundary degeneration effect that occurs when the output of each sub-encoder is scaled up.

- Deploying Hybrid Path Attention Semantic Aggregation (HPA-SA) on top of the encoder to guide the propagation of semantic signals in a hybrid direction.

- Employing methods such as separable convolution filters and channel reduction to keep the model's parameters and FLOPs low.

- Achieving a balance between model accuracy and efficiency. The proposed model, SFRSeg, achieves 70.6%, 74.7%, and 49.3% on the Cityscapes, CamVid, and KITTI test sets respectively, while having only 1.6 million parameters.

- SFRSeg can process 194 frames per second at a resolution of 512×1024, outperforming existing models in terms of speed.

- Evaluating the proposed model on the Indoor Object dataset, demonstrating impressive results with greater efficiency.

- Creating an official public GitHub repository and uploading project details, including the design of the proposed model.

2.9 Publication 9: Multi-encoder Context Aggregation Network for Structured and Unstructured Urban Street Scene Analysis

Bibliographic reference:

Singha, T., Pham, DS., Krishna, A., **"Multi-encoder Context Aggregation Network for Structured and Unstructured Urban Street Scene Analysis,"** in IEEE Access, vol. 11, pp. 66227-66244, 2023. doi: 10.1109/ACCESS.2023.3289968.

2.9.1 Abstract

Capturing the long-range dependencies and rich contextual details from a complex scene is a difficult task in semantic segmentation. Most of the existing models struggles to capture that due to their deep single branch encoder design. Moreover, rare class objects are often missed out by the model due to the presence of over occurred classes objects in the scenes. To address these issues, this paper introduces a multiple sub-encoders design for feature extraction where the deep feature maps are re-filtered multiple times by multiple sub-encoders. The novelty of this design lies in reusing the deep feature maps by successively increasing number of layers at the deeper stages in successive sub-encoders. At decoder end, it deploys a comprehensive Local and Global Context Aggregation (LGCA) module which accepts contextually rich feature maps from the different stages of the sub-encoders and significantly enhances the semantic representation by fusing these feature maps. Combining these novel designs, this paper introduced an efficient semantic segmentation model called Multi-encoder Context Aggregation Network (MCANet) for resource constrained mobile devices. Among the existing real-time models with less than 3 million parameters, the proposed model achieved sate-of-the-art performance on various datasets without ImageNet Deng

et al. (2009) pre-trained weights.

2.9.2 Approach

Existing models often suffer from the gradient vanishing issue and overlook tiny, far, and rare class objects in the scene while producing the semantic output. Gradient vanishing occurs due to their deep network architecture, and lack of feature re-usability can be the cause of overlooking tiny and far objects in the scene. In order to resolve these issues, this paper follows the approach of feature re-usability in the entire pipeline by introducing multiple sub-encoders design at the encoder and a multi-paths bi-directional decoder design. Deep features from the first sub-encoder are reprocessed by the successive sub-encoders to achieve the best refined global feature maps. Later on, these feature maps pass through multiple paths of the decoder for feature aggregation and accurate object localization.

2.9.3 Methodology and findings

The complete architecture of the proposed MCANet can be seen in Figure 2.10. To target resource-constrained mobile devices, the study carefully designs the multi-encoder and multi-path decoder. Empirically, it has been proven that MobileNetV2 Sandler et al. (2018) inverted residual blocks (MBConv) are more efficient than any other existing residual blocks. Hence, relying on its optimized layered architecture, this study designed the encoder architecture. It consists of four sub-encoders for feature reuses. The first sub-encoder has six stages, which consist of one standard convolution layer and 11 MBConv blocks. At each stage, the image size is bilinearly down-sampled. The second sub-encoder has three (fourth, fifth, and sixth) stages for reprocessing the last three stages' deep feature maps. Note that the number of layers in the fourth stage has increased. The third sub-encoder has two (fifth and sixth) stages, where the number of layers in the fifth stage has increased. The fourth sub-encoder has only the last (sixth) stage, which has more layers than the previous sub-encoders. In this way, the deep feature maps in the entire feature hierarchy are filtered more times as

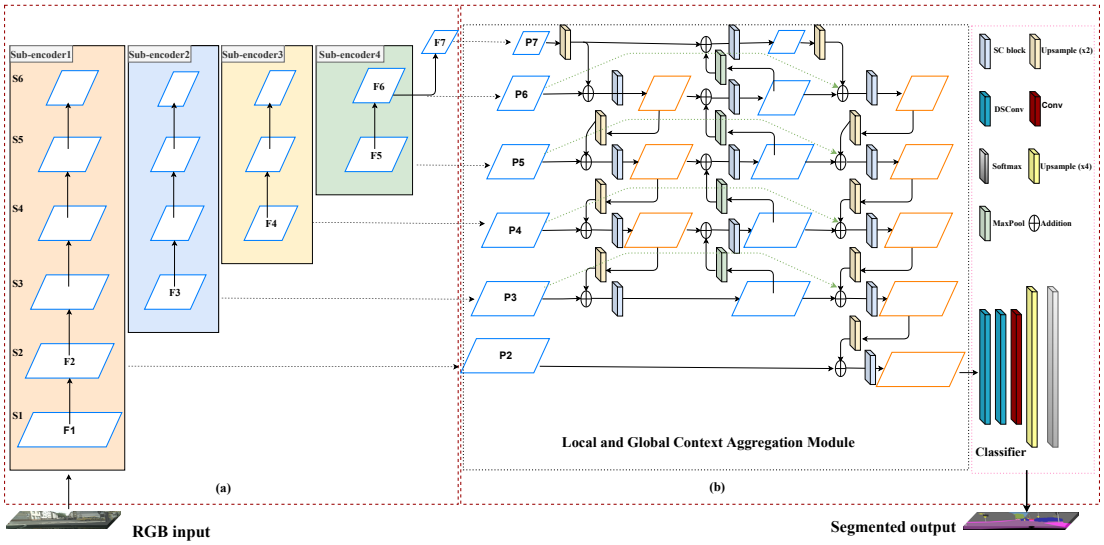

Figure 2.10: Complete architecture of MCANet (this figure is derived from the literature Singha et al. (2023b).)

they contain more contextual details. The paper also introduces several lateral connections among the same stages of all sub-encoders to address the gradient vanishing issue. Thus, the feature re-usability network is designed.

Since local spatial feature maps are important to localize the objects in the scene, this paper proposes a novel design for a feature fusion method called Local and Global Context Aggregation (LGCA). The design of LGCA can be seen in Figure 2.10. It has three paths: two top-down and one bottom-up. It takes features from the second to seventh stages of the encoder for feature fusion. Features from the fourth to seventh stages are contextually rich features; hence, these can be identified as deep features, whereas features at the second and third stages mainly contain local spatial details required to localize the objects in the scene. Thus, these features are aggregated through multiple paths for better semantic representation. The key findings of this study are as follows:

- Tiny, far, and rare classes objects are often overlooked by the existing models due to their deep single branch encoder design.
- Feature re-usability can improve the model's performance on rare classes objects.

- Increasing the number of layers in deep stages of successive sub-encoders enhances the deep feature quality.
- Lateral connections between the sub-encoders at the same stages mitigate the issue of gradient vanishing.
- For better context assimilation and precise object localization, fusing features through multiple paths is necessary.
- Spatial details are required for accurate object localization in the scene.

2.9.4 Contributions

The main contributions of this paper are as follows:

- The paper proposes a novel backbone architecture that incorporates multiple sub-encoders designed specifically for optimal feature scaling. Additionally, we control the number of model parameters by reducing the number of stages in each successive sub-encoder.
- The paper introduces an effective multi-stage module for local and global feature aggregation at the decoder. This module combines feature maps at different levels generated by our proposed backbone network.
- Leveraging the novel backbone architecture and efficient multi-stage feature aggregation module, the paper presents an efficient semantic segmentation model called Multi-encoder Context Aggregation Network (MCANet). This model achieves a sensible balance between accuracy and efficiency, making it well-suited for resource-constrained mobile devices.
- Finally, the study conducts comprehensive experiments in both structured and unstructured environments with varying numbers of classes. Our results demonstrate the superior performance of our model compared to existing real-time semantic segmentation models with fewer than 3 million (M) parameters.

2.10 Contributions of the thesis

This thesis aims to address several issues present in existing methods for complex scene understanding. These issues include:

- Large computational cost due to the large size of the network
- Model scalability issue
- Inability to capture objects of varied geometrical shapes in a complex scene
- Boundary degeneration effect
- Large semantic gaps among the feature maps in the entire pipeline
- Gradient vanishing issue
- Inability to capture rare classes objects in complex scenes
- Poor latency and infeasible for resource constrained applications
- Inability to maintain a balance between model accuracy and model efficiency

To address these issues, the thesis proposes several novel architectures and methods that can be utilized for computer vision tasks to improve scene understanding. These designs and methods are evaluated using various publicly available benchmarks. The thesis includes the publication of nine research articles that tackle these issues.

In summary, Publication 1 demonstrates that different dimensions of the deep convolutional network are not independent, and scaling the network along the depth may not substantially improve model performance. Hence, a compound scaling technique is relied upon to propose a family of scene segmentation models and address the issue of model scalability.

Publication 2 emphasizes the importance of building a lightweight backbone from scratch instead of using pre-trained deep CNNs. This approach optimizes

the end-to-end network architecture and provides control over the architectural design.

Publication 3 illustrates the importance of feature scaling and feature fusion techniques in the entire semantic pipeline. These techniques enhance the capture of objects of various geometric sizes and enable accurate region identification.

Publication 4 introduces the novel concept of knowledge sharing among the deep and shallow branches to improve the feature hierarchy in the entire encoder. A method called the context mining module is also introduced to filter out noise from the shared knowledge.

Based on this shared knowledge backbone, Publication 6 designs techniques for structural crack detection and segmentation, efficiently detecting building and road cracks from high-resolution input images.

Publication 5 demonstrates that the neck section of object detection techniques can be utilized in semantic segmentation to reduce semantic gaps among the feature maps of the encoder. This improves the overall quality of the feature hierarchy.

Publication 7 shows the importance of various sizes of receptive fields in the encoder for capturing long-range dependencies and rich contextual details. A novel short-term dense bottleneck design is introduced, providing three to four times more receptive fields than existing bottleneck blocks. This dense bottleneck can be used for designing CNNs related to various vision tasks.

Publications 8 and 9 highlight the importance of feature re-usability in addressing rare classes objects and gradient vanishing issues. They introduce a novel architecture called multiple sub-encoders for feature re-usability and feature extraction. Publication 8 also incorporates the concept of knowledge sharing using a knowledge shared block for building the encoder and employs a hybrid path attention mechanism to alleviate semantic gaps among the feature maps and improve overall semantic representation.

Publication 9 demonstrates that successive increases in the number of layers in

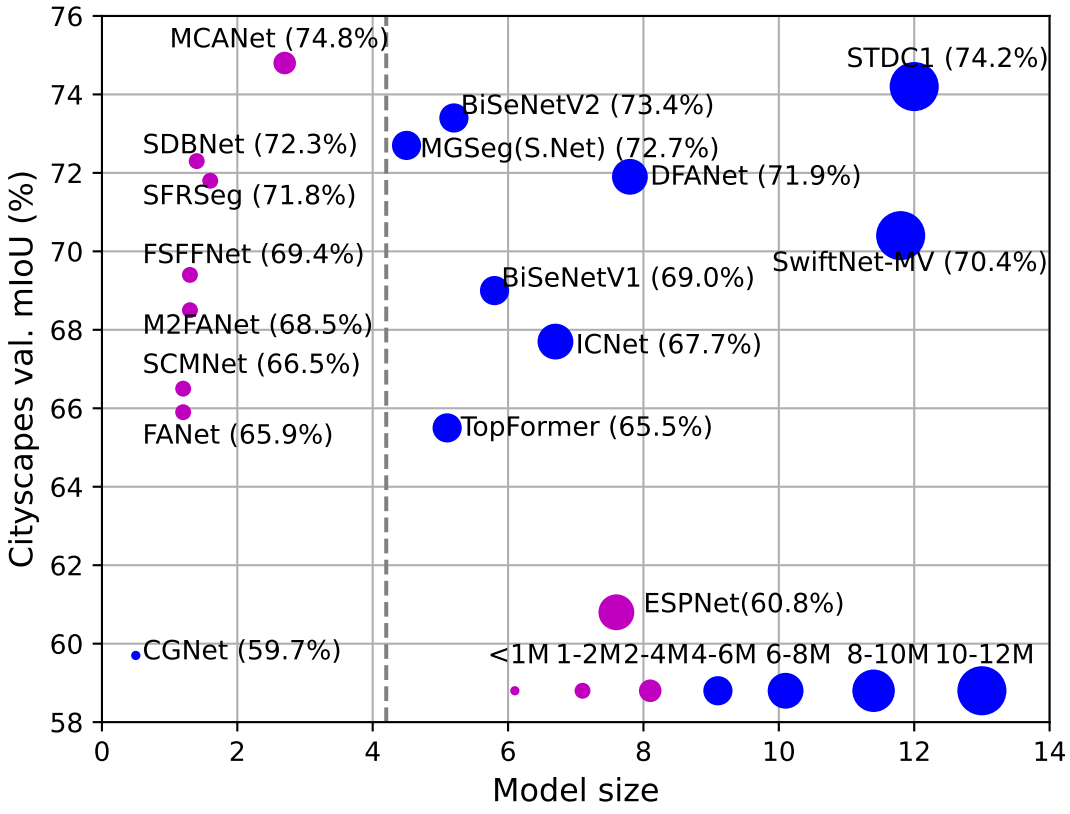

Figure 2.11: Cityscapes validation mIoU (%) Vs Model Size

the deep stages of the successive sub-encoders enhance the quality of deep feature maps. Aggregating spatial and global feature maps through multiple paths in both directions improves signal flow and enables better object representation in the semantic output.

To compare the performance of all the proposed models with existing real-time semantic segmentation models, two scatter plots were generated and displayed in Figures 2.11 and 2.12. The first plot in Figure 2.11 illustrates the model's validation mIoU on the Cityscapes dataset Cordts et al. (2016) against the model size. The proposed models are represented by magenta color dots, while the existing real-time models are represented by blue color dots. The size of each dot corresponds to the size of the model.

It is evident that, in comparison to the existing real-time models, all the proposed models (except ESPNet Singha et al. (2020b)) are significantly smaller

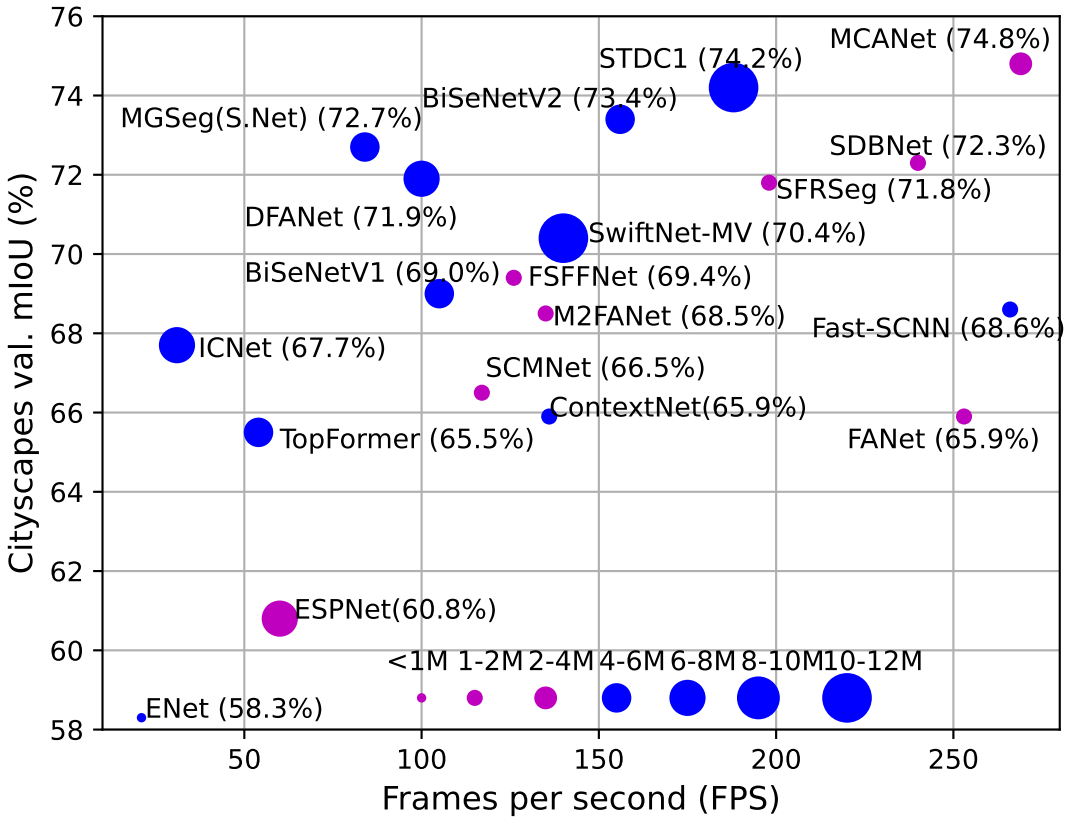

Figure 2.12: Cityscapes validation mIoU (%) Vs Frame per second (FPS)

and still achieve competitive results, despite having fewer parameters. ESPNet Singha et al. (2020b) utilizes the EfficientNet Tan & Le (2019) B0 backbone, which results in a higher number of parameters compared to the other proposed models.

The second plot in Figure 2.12 showcases the model's Cityscapes validation mIoU against FPS. It demonstrates that most of the proposed models exhibit significantly higher efficiency compared to existing real-time semantic segmentation models. Among them, the proposed MCANet Singha et al. (2023b) stands out with its state-of-the-art performance. MCANet achieves a Cityscapes validation mIoU of 74.8% while having only 2.7 million parameters. Additionally, it can process 269 frames per second at an input resolution of 512×1024. It is important to note that the FPS count of some of the proposed models may vary with the literature as different TRT-based engines are used.

When compared to all existing real-time models, MCANet achieves the most sensible trade-off between the model's accuracy and efficiency, making it an outstanding choice for practical applications.

Chapter 3

Conclusions, limitations and future research

3.1 Conclusions

Balancing the accuracy and efficiency of semantic segmentation models is crucial in various real-time applications, where computational resources are limited, and rapid decision-making is essential. Some of these applications are:

- **Video Surveillance:** For security and surveillance systems, real-time semantic segmentation plays a pivotal role in identifying objects and activities. Surveillance cameras continuously capture video data, and to efficiently analyze this continuous stream without overburdening computational resources, it's imperative to employ an efficient semantic segmentation model.

  In some scenarios, such as aerial surveillance using drones, the use of embedded devices is common. These devices typically have limited GPU support and face power supply constraints. Consequently, running a resource-intensive, memory-hungry, and power-demanding large model becomes infeasible on these embedded devices. Thus, the deployment of a lightweight model with minimal computational overhead becomes a necessity in such

contexts. This ensures that the surveillance system can effectively operate within the constraints of these devices.

- **Robotics:** Robots are increasingly integrated into various settings, including manufacturing, warehouses, and even homes, where they heavily rely on semantic segmentation for effective navigation and interaction with their surroundings. To underscore the importance of this technology, consider its application in assisting blind individuals using wheelchairs to navigate indoor environments.

  In this scenario, efficient object detection and segmentation techniques play a crucial role. These techniques enable the robotic wheelchair to identify obstacles, pathways, and objects in real-time, facilitating safe and autonomous navigation for the user. It's a prime example of where the balance between efficiency and accuracy is paramount. While real-time processing ensures timely responses to dynamic environments, the accuracy of the semantic segmentation is vital to ensure the safety and reliability of the robotic system, ultimately enhancing the user's mobility and independence.

- **Augmented Reality:** In augmented reality applications, such as interactive gaming or navigation assistance, real-time object detection is essential. However, achieving optimal performance on mobile devices presents a unique challenge. It's crucial to strike the perfect balance between accuracy and efficiency to provide users with a seamless experience.

  For instance, consider a mobile augmented reality game where users explore a virtual world overlaid on their surroundings through their smartphone camera. To accurately detect and interact with virtual objects in real time, the segmentation model must be efficient. Excessive computational demands can result in lag, detracting from the immersive experience. Therefore, finding the best trade-off between accuracy and efficiency is vital to ensure that the augmented reality game runs smoothly and provides an

engaging user experience on mobile devices.

- **Structural monitoring system:** Continuously employing semantic segmentation on data collected from sensors, drones, or cameras serves as a powerful tool for real-time structural health monitoring. This technology can be exemplified in the context of bridge inspections.

  Consider a scenario where cameras and sensors are mounted on a bridge to monitor its structural integrity. Semantic segmentation algorithms analyze the real-time data to identify any anomalies or structural issues, such as cracks, corrosion, or deformation. If the system detects deviations from the norm, it can promptly trigger alerts for maintenance or inspection. This proactive approach to structural health monitoring helps ensure the safety and longevity of critical infrastructure.

- **Monitoring system for agriculture:** In the realm of precision agriculture, drones equipped with high-resolution cameras and real-time segmentation models have emerged as valuable tools. These technologies can be exemplified in the context of monitoring crop health and identifying pest infestations.

  Imagine a vast agricultural field where drones equipped with advanced cameras fly over the crops. Real-time segmentation models meticulously process the data, distinguishing between healthy and distressed plants and detecting signs of potential pest infestations. The crux of the matter lies in accurately identifying distressed plants within the real-time environment. A precise model is indispensable for making informed decisions regarding the timing and location of treatments. With access to this real-time information, farmers can optimize their interventions, reduce the need for excessive pesticide use, and ultimately boost crop yields while conserving valuable resources.

In these applications, achieving the delicate balance between accuracy and effi-

ciency is paramount to ensure that models provide valuable and timely insights without straining the available computing resources. However, the challenge lies in the sheer number of parameters present in existing semantic segmentation models, making their deployment on resource-constrained mobile devices a formidable task. Take, for example, NVIDIA Jetson devices, recognized for their AI capabilities and compact form factors, which are frequently employed in such scenarios.

Among the Jetson devices, the Jetson Nano stands out as the most resource-constrained. To successfully deploy a semantic segmentation model built using deep learning frameworks like TensorFlow, PyTorch, or ONNX on the Jetson platform, it necessitates the optimization of the model through the TensorRT engine. A prime illustration of this is the DeepLabV3+ model with the Xception backbone, whose size can span from 100 MB to 150 MB contingent upon factors like input image size, precision mode, and structural optimization. This model comprises over 50 million parameters, rendering it impractical for utilization on resource-constrained devices. However, post-optimization with the TensorRT engine, the size of the DeepLabV3+ model can be efficiently reduced to a range of 60 MB to 100 MB.

Such optimization endeavors can yield significant advantages, accelerating model inference and diminishing computational demands for real-time performance. Nonetheless, even with optimization, running on real-time devices like the NVIDIA Jetson Nano remains a challenge. Typically, models with smaller footprints, such as those based on the MobileNet architecture, boasting a TensorRT model size of around 10-15 MB and fewer than 3 million parameters, can efficiently operate on the Jetson Nano.

To accomplish this objective, the thesis introduces a range of pioneering concepts, optimized backbone architectures, and efficient methodologies aimed at crafting real-time semantic segmentation models specifically tailored for resource-constrained mobile applications. Remarkably, all of the proposed architectures boast an impressively low parameter count, with each containing fewer than 3

million parameters. What's more, following TensorRT optimization, these models exhibit a compact size range spanning from 7 MB to 15 MB. This remarkable reduction in model size renders them exceptionally well-suited for deployment on devices like the Jetson Nano.

These architectural innovations serve a dual purpose: they not only bridge the performance divide that often separates real-time and offline semantic segmentation models but also bolster the overall efficiency of these models for real-time computational tasks. This makes the proposed architectures ideal for resource-constrained applications where minimal hardware resources are available to support segmentation tasks. Furthermore, it's worth noting that these innovative architectures and methods extend their applicability to various computer vision tasks, including object detection, instance segmentation, and panoptic segmentation.

The thesis presents various novel backbone designs, including the shared knowledge two-branch design for feature extraction, feature re-usability design by deploying multiple sub-encoders, short-term dense bottleneck design, and encoder with neck design. These innovative CNN architectures effectively address several issues present in existing semantic segmentation models. These issues include the inability to capture long-range dependencies, large semantic gaps among feature maps, gradient vanishing, and the inability to capture tiny, distant, and rare classes of objects. The proposed architectures efficiently address these issues. Additionally, due to the optimized network architectures, all these feature extractors produce fewer parameters and FLOPs, significantly reducing the computational cost. This makes these architectures feasible for resource-constrained mobile devices.

The proposed methods, such as context mining, feature refinement, spatial and contextual feature aggregation, semantic aggregation, and Hybrid path attention modules, address several other issues, including noise filtering, context assimilation, boundary degeneration, and precise region and object localization.

These methods have a smaller memory footprint and can be deployed on any CNN network. However, as the main objective of this thesis is to reduce model architecture without sacrificing performance, we tested the performance of these modules only with the proposed network architectures.

All the proposed segmentation models, except ESPNet Singha et al. (2020b), have their own optimized novel backbones designed from scratch. As a result, these models have a parameter range of 1.2 to 2.7 million and 2.9G to 7.8G FLOPs at a resolution of 512×1024, indicating lower computational costs. Only ESPNet Singha et al. (2020b) incorporates the EfficientNet Tan & Le (2019) B0 backbone, which adds 5.6 million parameters. Thus, the complete ESPNet pipeline has a total of 7.6 million parameters.

The proposed models can process more than 100 frames per second at a resolution of 512×1024. In particular, FANet, SFRSeg, and MCANet demonstrate outstanding efficiency by processing around 200 or more frames per second. The proposed model MCANet achieves state-of-the-art results on Cityscapes Cordts et al. (2016) (73.4%), CamVid Brostow et al. (2009) (80.2%), BDD100K Yu et al. (2020) (58.8%), KITTI Alhaija et al. (2018) (58.5%), and IDD-lite Mishra et al. (2020) (73.8%) among the existing real-time semantic segmentation models with less than 3 million parameters. Thus, the thesis demonstrates achieving a sensible trade-off between the accuracy and efficiency of the model.

This thesis also proposes an efficient method for detecting and segmenting structural cracks, which achieves state-of-the-art performance on various publicly available crack datasets. It can process 220 frames per second at a resolution of 448×448, demonstrating its high efficiency in real-time environments. Thanks to the optimized and efficient network architecture, this crack detection method can be easily implemented in real-time embedded devices for monitoring the structural condition of buildings or roads.

The test results of the proposed models, evaluated using the Cityscapes Cordts et al. (2016) and KITTI Alhaija et al. (2018) datasets, have been published on

the test evaluation server.

An official GitHub repository has been created for each research paper, making it publicly available for reusability.

3.2 Limitations and future work

3.2.1 Limitations

Even though all the proposed architectures and methods improved segmentation model's performance for resource-constrained mobile devices, there are several limitations that hinder the model's performance. These limitations are:

- **Performance gap:** Due to the increasing demand for developing resource-constrained applications in various fields, the thesis has focused on designing real-time semantic segmentation models which can handle high resolution input images with greater efficacy and efficiency. These models are designed to cater to application-specific requirements, where real-time performance with good accuracy is essential. It substantially reduces the performance gap between the offline and real-time segmentation models. However, still certain percentage of performance gap can be observed. For instance, DeepLabV3+ Chen et al. (2018) has 54.6 million parameters and it produces 79.6% mIou on Cityscapes validation set. In comparison, the proposed model MCANet generates 74.8% validation mIoU while having only 2.7 million parameters. the proposed model is 20 times smaller than DeepLabV3+. Hence, it would not be possible to attain the same result with a small model such as MCANet. However, the existing performance gap can be reduced further with the new emerging techniques such as domain adaption Yang et al. (2020), teacher-student and semi-supervised design Xiao et al. (2022); Baldeon Calisto (2022). This thesis did not explore these techniques extensively.

- **Lack of pre-trained knowledge:** Most of the existing semantic segmenta-

tion models are pre-trained on datasets such as ImageNetDeng et al. (2009) or other publicly available datasets or synthetic datasets. This pre-trained knowledge boosts the model's performance. However, all the proposed models in this thesis are designed from scratch and are not pre-trained on any datasets. Pre-training the proposed models with the ImageNet dataset can improve model's accuracy by 1% to 2%.

- **Limited generalization:** Semantic segmentation models may struggle with generalization to unseen or novel classes. If the model has not been trained on examples of specific classes, it may have difficulty accurately segmenting them or distinguishing them from similar-looking classes. This observation has been noticed while using the IDD-lite dataset, where some images taken in rural environments are completely different from urban street scenes, leading the model to misclassify cart-tracks as roads.

- **Limited occlusion handling:** The proposed models relatively perform well than the existing real-time semantic segmentation models. However, it has been observed that the segmentation models may not effectively handle changes in lighting conditions, viewpoint variations, or occlusions. These factors can introduce challenges in accurately segmenting objects under different circumstances, leading to decreased performance and reliability in real-world scenarios. This phenomenon has been observed in the BDD100K dataset, which contains images taken in various lighting conditions and weather conditions. For instance, the number of images taken at night under low lighting conditions is relatively low. As a result, the models have limited exposure to such scenarios during the training process. Consequently, they may struggle to accurately segment scenes captured at night or from different viewpoints. New techniques such as Domain adaptation Yang et al. (2020), semi-supervised Xiao et al. (2022); Baldeon Calisto (2022) can be beneficial in addressing issues like limited occlusion and improving generalization.

- **Slight squarish boundaries:** Semantic segmentation algorithms can encounter difficulties in accurately delineating fine-grained object boundaries. Due to the nature of pixel-level classification, objects with intricate shapes or regions with ambiguous boundaries pose challenges for precise segmentation. In some proposed models, slightly squared boundaries have been observed in the segmented output. This is attributed to the aggressive upsampling (by a factor of 2^2 or 2^3) at the end of the decoder, which significantly improves model efficiency but can lead to squared boundaries. To address this issue, pre-processing or post-processing techniques can be applied during the training process. These techniques include boundary detail analysis, scaling and clipping, and post-processing filtering. It is important to note that these pre/post-processing tasks do not affect the model's efficiency during the inference time. Some existing models utilize these techniques to enhance performance, but this thesis did not explore them extensively.

3.2.2 Future work

Addressing these aforementioned limitations is an active area of research, and several advancements can be made in the future to improve the robustness, efficiency, and generalization capabilities of semantic segmentation models.

The immediate future work, which would be conducted to address the squarish boundary issue, is to generate the boundary ground truth of all objects from the original semantic ground truth and fuse this boundary detail into the main pipeline as additional information to improve the model's overall performance. In this case, along with the main loss function, a boundary loss function needs to be evaluated during the training process. The boundary ground truth can be seen in Figure 3.1. Figure 3.1(b) shows the original colored ground truth provided by the dataset, and Figure 3.1(c) displays the newly generated boundary details, which can be used during training as additional information. Other

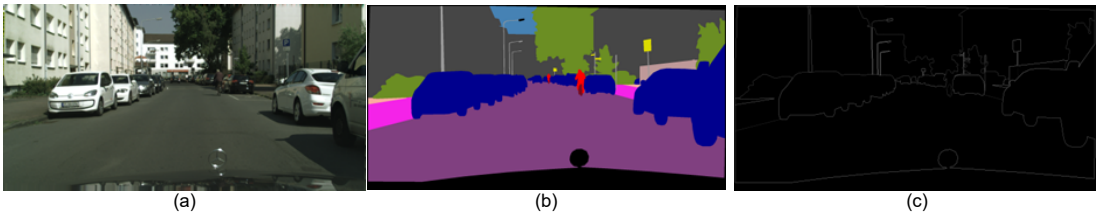

Figure 3.1: Boundary annotation

pre/post-processing techniques will also be explored in the future to assess their effectiveness in computer vision tasks.

The thesis references several pieces of literature He et al. (2019); Liu et al. (2019); Park & Heo (2020) that have implemented a technique known as knowledge distillation for reducing a model's parameters while maintaining its performance. This technique involves training a smaller model, referred to as the student, to emulate the behavior of a larger model, the teacher. Throughout training, the student assimilates the teacher's knowledge and predictions, guided by a distillation loss rooted in the similarity between their outputs. This methodology proves advantageous for crafting lightweight models well-suited for resource-constrained environments, such as edge devices like the NVIDIA Jetson Nano. Nonetheless, it's important to acknowledge that this technique carries several limitations. These include the potential loss of fine-grained information, a heavy reliance on the teacher model, restricted generalization, sensitivity to hyperparameters, the complexity of the training process, and computational resource intensiveness. Furthermore, existing models developed through this technique often maintain a considerable number of parameters. For example, in the work by Park & Heo (2020), the DeepLabV3+ model Chen et al. (2018) with Xception as an encoder serves as the teacher model, while the student model employs MobileNetV2 Sandler et al. (2018), containing 2.3 million parameters. Through knowledge distillation, the student model achieves test accuracy of 64.7% and 60.7% on the Cityscapes Cordts et al. (2016) and CamVid Brostow et al. (2009) datasets, respectively. Despite these results, a significant performance gap between offline and real-time semantic segmentation models persists.

Looking ahead, the thesis will delve deeper into this technique by integrating one of the proposed models as the student model. Given that many existing semantic segmentation models struggle to generalize effectively to unseen or out-of-distribution data, the future research work will concentrate on enhancing the models' generalization capabilities. This will involve the incorporation of domain adaptation techniques, domain randomization, or the utilization of few-shot learning approaches to enable the models to adapt to new domains, even when limited labeled data is available.

Furthermore, in the future, the thesis can explore the deployment of multiple models in a distributed fashion, with their outputs combined using techniques like model averaging or weighted averaging. This ensemble approach often leads to enhanced segmentation performance. However, it's important to note that this approach can be resource-intensive and may increase inference time. Given that the primary objectives of the thesis revolve around striking a balance between model accuracy and efficiency, this particular approach is not explored in the thesis.

While the thesis primarily focuses on pixel-level labeling for outdoor scenes, there is a growing need for fine-grained segmentation. This entails segmenting objects at a more detailed level, capturing object parts and fine-grained structures. Instance-level segmentation, which involves detecting and segmenting individual instances of objects, will also continue to be an important research direction. Additionally, panoptic segmentation provides a holistic understanding of a scene by incorporating both semantic and instance-level information. This enables various applications such as object counting, tracking, and further analysis at the individual object level, contributing to a more detailed scene understanding. Thus, panoptic segmentation will be an important research direction for the future.

For further performance improvement of semantic segmentation models, integrating information from multiple modalities, such as RGB images, depth data, or point clouds, can be beneficial. Therefore, future research will explore techniques

for effectively fusing multi-modal data and leveraging sensor fusion to improve segmentation accuracy and handle challenging scenarios.

Currently, semantic segmentation models heavily rely on pixel-level annotations for training. Future research will explore weakly supervised learning approaches that can learn from coarser annotations like image-level labels or bounding boxes. Additionally, unsupervised or self-supervised learning techniques will be investigated to leverage unlabeled data for training semantic segmentation models.

Semantic segmentation models often lack transparency, making it challenging to understand their decision-making processes. Future research will focus on developing interpretable and explainable models that can provide insights into how the model arrives at its segmentation predictions. This will enable better trust, debugging, and accountability in critical applications.

Bibliography

F. Abdullah & A. Jalal (2023). 'Semantic segmentation based crowd tracking and anomaly detection via neuro-fuzzy classifier in smart surveillance system'. *Arabian Journal for Science and Engineering* **48**(2):2173–2190.

H. Alhaija, et al. (2018). 'Augmented Reality Meets Computer Vision: Efficient Data Generation for Urban Driving Scenes'. *IJCV* .

R. Amhaz, et al. (2016). 'Automatic crack detection on two-dimensional pavement images: An algorithm based on minimal path selection'. *IEEE Transactions on Intelligent Transportation Systems* **17**(10):2718–2729.

V. Badrinarayanan, et al. (2017). 'SegNet: A deep convolutional encoder-decoder architecture for image segmentation'. *IEEE TPAMI* **39**(12):2481–2495.

M. Baldeon Calisto (2022). 'Teacher-Student Semi-supervised Approach for Medical Image Segmentation'. In *MICCAI Challenge on Fast and Low-Resource Semi-supervised Abdominal Organ Segmentation*, pp. 152–162. Springer.

G. J. Brostow, et al. (2009). 'Semantic object classes in video: A high-definition ground truth database'. *Pattern Recognition Letters* **30**(2):88–97.

W. Cai & B. Wang (2023). 'DSE-Net: Deep Semantic Enhanced Network for Mobile Tongue Image Segmentation'. In *Proc. ICONIP*, pp. 138–150. Springer.

L.-C. Chen, et al. (2017). 'DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs'. *IEEE TPAMI* **40**(4):834–848.

L.-C. Chen, et al. (2018). 'Encoder-decoder with atrous separable convolution for semantic image segmentation'. In *Proc. ECCV*, pp. 801–818.

S. Choi, et al. (2020). 'Cars can't fly up in the sky: Improving urban-scene segmentation via height-driven attention networks'. In *Proc. CVPR*, pp. 9373–9383.

W. Choi & Y.-J. Cha (2019). 'SDDNet: Real-time Crack Segmentation'. *IEEE Transactions on Industrial Electronics* **PP**:1–1.

F. Chollet (2017). 'Xception: Deep learning with depthwise separable convolutions'. In *Proc. CVPR*, pp. 1251–1258.

M. Cordts, et al. (2016). 'The Cityscapes Dataset for Semantic Urban Scene Understanding'. In *Proc. CVPR*.

J. Deng, et al. (2009). 'ImageNet: A large-scale hierarchical image database'. In *Proc. CVPR*, pp. 248–255. Ieee.

A. Dosovitskiy, et al. (2020). 'An image is worth 16x16 words: Transformers for image recognition at scale'. *arXiv preprint arXiv:2010.11929* .

M. Eisenbach, et al. (2017). 'How to Get Pavement Distress Detection Ready for Deep Learning? A Systematic Approach.'. In *Proc. IJCNN*, pp. 2039–2047.

M. Everingham, et al. (2015). 'The pascal visual object classes challenge: A retrospective'. *IJCV* **111**(1):98–136.

M. Fan, et al. (2021). 'Rethinking BiSeNet for real-time semantic segmentation'. In *Proc. CVPR*, pp. 9716–9725.

D. Freedman & T. Zhang (2005). 'Interactive graph cut based segmentation with shape priors'. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 755–762 vol. 1.

J. Fu, et al. (2019). 'Dual attention network for scene segmentation'. In *Proc. CVPR*, pp. 3146–3154.

R. Girshick, et al. (2014). 'Rich feature hierarchies for accurate object detection and semantic segmentation'. In *Proc. CVPR*, pp. 580–587.

J.-Y. He, et al. (2021). 'MGSeg: Multiple granularity-based real-time semantic segmentation network'. *IEEE TIP* **30**:7200–7214.

T. He, et al. (2019). 'Knowledge adaptation for efficient semantic segmentation'. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 578–587.

A. Howard, et al. (2019). 'Searching for MobileNetv3'. In *Proc. ICCV*, pp. 1314–1324.

G. Huang, et al. (2017). 'Densely connected convolutional networks'. In *Proc. CVPR*, pp. 4700–4708.

A. Khan, et al. (2022). 'A deep semantic vegetation health monitoring platform for citizen science imaging data'. *Plos one* **17**(7):e0270625.

W. Kim & J. Seok (2018). 'Indoor semantic segmentation for robot navigating on mobile'. In *2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN)*, pp. 22–25. IEEE.

K.-T. Lai, et al. (2021). 'High-efficient semantic segmentation for Internet-of-Things-enabled smart cameras'. In *2021 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW)*, pp. 1–2. IEEE.

Y. LeCun, et al. (1998). 'Gradient-based learning applied to document recognition'. *Proc. of the IEEE* **86**(11):2278–2324.

H. Li, et al. (2019). 'DFANet: Deep feature aggregation for real-time semantic segmentation'. In *Proc. CVPR*, pp. 9522–9531.

G. Lin, et al. (2017a). 'RefineNet: Multi-path refinement networks for high-resolution semantic segmentation'. In *Proc. CVPR*, pp. 1925–1934.

T.-Y. Lin, et al. (2017b). 'Feature pyramid networks for object detection'. In *Proc. CVPR*, pp. 2117–2125.

S. Liu, et al. (2018). 'Path aggregation network for instance segmentation'. In *Proc. CVPR*, pp. 8759–8768.

Y. Liu, et al. (2019). 'Structured knowledge distillation for semantic segmentation'. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2604–2613.

J. Long, et al. (2015). 'Fully convolutional networks for semantic segmentation'. In *Proc. CVPR*, pp. 3431–3440.

A. Mishra, et al. (2020). 'Semantic segmentation datasets for resource constrained training'. In *Proc. NCVPRIPG*, pp. 450–459. Springer.

E. Mohamed, et al. (2022). 'A pixel-wise annotated dataset of small overlooked indoor objects for semantic segmentation applications.'. *Data in Brief* **40**:107791.

F. Nayyeri & J. Zhou (2021). 'Multi-Resolution ResNet for Road and Bridge Crack Detection'. In *Proc. DICTA*, pp. 1–8. IEEE.

M. Orsic, et al. (2019). 'In defense of pre-trained imagenet architectures for real-time semantic segmentation of road-driving images'. In *Proc. CVPR*, pp. 12607–12616.

S. Park & Y. S. Heo (2020). 'Knowledge distillation for semantic segmentation using channel and spatial correlations and adaptive cross entropy'. *Sensors* **20**(16):4616.

A. Paszke, et al. (2016). 'ENet: A deep neural network architecture for real-time semantic segmentation'. *arXiv preprint arXiv:1606.02147* .

N. Plath, et al. (2009). 'Multi-class image segmentation using conditional random fields and global classification'. In *Proc. ICML*, pp. 817–824.

R. P. Poudel, et al. (2018). 'ContextNet: Exploring context and detail for semantic segmentation in real-time'. *arXiv preprint arXiv:1805.04554* .

R. P. Poudel, et al. (2019). 'Fast-SCNN: Fast semantic segmentation network'. *arXiv preprint arXiv:1902.04502* .

P. H. Progga & S. Shatabda (2023). 'iResSENet: An Accurate Convolutional Neural Network for Retinal Blood Vessel Segmentation'. In *Proc. ICONIP*, pp. 567–578. Springer.

J. Redmon, et al. (2016). 'You only look once: Unified, real-time object detection'. In *Proc. CVPR*, pp. 779–788.

S. Ren, et al. (2015). 'Faster R-CNN: Towards real-time object detection with region proposal networks'. *Advances in neural information processing systems* **28**.

B. Revanasiddappa, et al. (2020). 'Real-time early detection of weed plants in pulse crop field using drone with IoT'. *Technology* **16**(5):1227–1242.

O. Ronneberger, et al. (2015). 'U-Net: Convolutional networks for biomedical image segmentation'. In *Proc. MICCAI*, pp. 234–241. Springer.

M. Sandler, et al. (2018). 'MobileNetv2: Inverted residuals and linear bottlenecks'. In *Proc. CVPR*, pp. 4510–4520.

Y. Shi, et al. (2016). 'Automatic road crack detection using random structured forests'. *IEEE Transactions on Intelligent Transportation Systems* **17**(12):3434–3445.

J. Shotton, et al. (2008). 'Semantic texton forests for image categorization and segmentation'. In *2008 IEEE conference on computer vision and pattern recognition*, pp. 1–8. IEEE.

K. Simonyan & A. Zisserman (2014). 'Very deep convolutional networks for large-scale image recognition'. *arXiv preprint arXiv:1409.1556* .

T. Singha, et al. (2021a). 'SCMNet: Shared context mining network for real-time semantic segmentation'. In *Proc. DICTA*, pp. 1–8. IEEE.

T. Singha, et al. (2022). 'SC-CrackSeg: A Real-time Shared Feature Pyramid Network for Crack Detection and Segmentation'. In *Proc. DICTA*, pp. 1–8.

T. Singha, et al. (2020a). 'FANet: Feature aggregation network for semantic segmentation'. In *Proc. DICTA*, pp. 1–8. IEEE.

T. Singha, et al. (2021b). 'Urban street scene analysis using lightweight multi-level multi-path feature aggregation network'. *Multiagent and Grid Systems* **17**(3):249–271.

T. Singha, et al. (2023a). 'A real-time semantic segmentation model using iteratively shared features in multiple sub-encoders'. *Pattern Recognition* **140**:109557.

T. Singha, et al. (2023b). 'Multi-encoder Context Aggregation Network for Structured and Unstructured Urban Street Scene Analysis'. *IEEE Access* **11**:66227–66244.

T. Singha, et al. (2020b). 'Efficient Segmentation Pyramid Network'. In *Proc. ICONIP*, pp. 386–393. Springer.

T. Singha, et al. (2021c). 'A lightweight multi-scale feature fusion network for real-time semantic segmentation'. In *Proc. ICONIP*, pp. 193–205. Springer.

F. Stache, et al. (2023). 'Adaptive path planning for UAVs for multi-resolution semantic segmentation'. *Robotics and Autonomous Systems* **159**:104288.

R. Strudel, et al. (2021). 'Segmenter: Transformer for semantic segmentation'. In *Proc. CVPR*, pp. 7262–7272.

K. Sun, et al. (2019). 'High-resolution representations for labeling pixels and regions'. *arXiv preprint arXiv:1904.04514* .

C. Szegedy, et al. (2015). 'Going deeper with convolutions'. In *Proc. CVPR*, pp. 1–9.

M. Tan & Q. Le (2019). 'Efficientnet: Rethinking model scaling for convolutional neural networks'. In *Proc. ICML*, pp. 6105–6114. PMLR.

M. Tan, et al. (2020). 'EfficientDet: Scalable and efficient object detection'. In *Proc. CVPR*, pp. 10781–10790.

L. Tanzi, et al. (2021). 'Real-time deep learning semantic segmentation during intra-operative surgery for 3D augmented reality assistance'. *International Journal of Computer Assisted Radiology and Surgery* **16**(9):1435–1445.

S. Targ, et al. (2016). 'ResNet in ResNet: Generalizing residual architectures'. *arXiv preprint arXiv:1603.08029* .

S. Woo, et al. (2018). 'Cbam: Convolutional block attention module'. In *Proc. ECCV*, pp. 3–19.

M. Wu, et al. (2019a). 'Towards accurate high resolution satellite image semantic segmentation'. *Ieee Access* **7**:55609–55619.

T. Wu, et al. (2020). 'CGNet: A light-weight context guided network for semantic segmentation'. *IEEE TIP* **30**:1169–1179.

Z. Wu, et al. (2019b). 'Wider or deeper: Revisiting the ResNet model for visual recognition'. *Pattern Recognition* **90**:119–133.

H. Xiao, et al. (2022). 'Semi-supervised semantic segmentation with cross teacher training'. *Neurocomputing* **508**:36–46.

F. Yang, et al. (2019). 'Feature pyramid and hierarchical boosting network for pavement crack detection'. *IEEE Transactions on Intelligent Transportation Systems* **21**(4):1525–1535.

J. Yang, et al. (2020). 'Label-driven reconstruction for domain adaptation in semantic segmentation'. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVII 16*, pp. 480–498. Springer.

M. Yang, et al. (2018). 'DenseASPP for Semantic Segmentation in Street Scenes'. In *Proc. CVPR*, pp. 3684–3692. IEEE.

N. Yang & H. Tang (2021). 'Semantic segmentation of satellite images: A deep learning approach integrated with geospatial hash codes'. *Remote Sensing* **13**(14):2723.

C. Yu, et al. (2021). 'BiSeNetV2: Bilateral network with guided aggregation for real-time semantic segmentation'. *International Journal of Computer Vision* **129**(11):3051–3068.

C. Yu, et al. (2018). 'BiseNet: Bilateral segmentation network for real-time semantic segmentation'. In *Proc. ECCV*, pp. 325–341.

F. Yu, et al. (2020). 'BDD100k: A diverse driving dataset for heterogeneous multitask learning'. In *Proc. CVPR*, pp. 2636–2645.

Y. Yuan, et al. (2020). 'Object-contextual representations for semantic segmentation'. In *Proc. ECCV*, pp. 173–190. Springer.

H. Zhang, et al. (2020). 'Slimmer: Accelerating 3D semantic segmentation for mobile augmented reality'. In *2020 IEEE 17th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pp. 603–612. IEEE.

L. Zhang, et al. (2016). 'Road crack detection using deep convolutional neural network'. In *Proc. ICIP*, pp. 3708–3712. IEEE.

W. Zhang, et al. (2022). 'TopFormer: Token pyramid transformer for mobile semantic segmentation'. In *Proc. CVPR*, pp. 12083–12093.

H. Zhao, et al. (2018). 'ICNet for real-time semantic segmentation on high-resolution images'. In *Proc. ECCV*, pp. 405–420.

H. Zhao, et al. (2017). 'Pyramid scene parsing network'. In *Proc. CVPR*, pp. 2881–2890.

P. Zheng, et al. (2022). 'Brain tumour segmentation based on an improved U-Net'. *BMC Medical Imaging* **22**(1):1–9.

S. Zheng, et al. (2021). 'Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers'. In *Proc. CVPR*, pp. 6881–6890.

Q. Zou, et al. (2012). 'CrackTree: Automatic crack detection from pavement images'. *Pattern Recognition Letters* **33**(3):227–238.

Every reasonable effort has been made to acknowledge the owners of copyright material. I would be pleased to hear from any copyright owner who has been omitted or incorrectly acknowledged.

Appendices

Publications

.1 Publication 1

Efficient Segmentation Pyramid Network

Tanmay Singha( ), Duc-Son Pham, Aneesh Krishna, and Joel Dunstan

School of Electrical Engineering, Computing, and Mathematical Sciences,
Curtin University, Bentley, WA 6102, Australia
tanmay.singha@postgrad.curtin.edu.au

Abstract. Extensive growth in the field of robotics and autonomous industries, the demand for efficient image segmentation is increasing rapidly. Whilst existing methods have been shown to achieve outstanding results on challenging data sets, they cannot scale the model properly for real-world computational constraints applications due to a fixed large backbone network. We propose a novel architecture for semantic scene segmentation suitable for resource-constrained applications. Specifically, we make use of the global contextual prior by using a pyramid pooling technique on top of the backbone network. We also employ the recently proposed EfficientNet network to make our model efficiently scalable for computational constraints. We show that our newly proposed model - Efficient Segmentation Pyramid Network (ESPNet) - outperforms many existing scene segmentation models and produces 88.5% pixel accuracy on validation and 80.9% on training set of the Cityscapes benchmark.

Keywords: Scene segmentation · Pyramid pooling · EfficientNet

1 Introduction

Semantic segmentation is a process to label each pixel of an input image [1,2] with a class. To improve the performance, a common approach is to increase the size of the backbone network [7,18,19] such as from ResNet-18 to ResNet-200. However, all these deep convolutional neural networks (DCNNs) have balanced only one of these dimensions- depth, width and image resolution. Recently, a family of models, called EfficientNet has been proposed in [18] by exploiting an effective compound scaling technique to balance all the dimensions and optimise the architecture for a given computational constraint. Inspired by this, we use EfficientNet's B0 network as a feature extractor to develop an efficient scalable segmentation model for real-time computation. We also employ the pyramid pooling module (PPM) [5,22] to extract region-based global information from the feature map. Finally, we use a classifier module at the output stage.

We propose two models for semantic segmentation targetting real-time embedded devices, namely Base ESPNet and Final ESPNet, the latter includes an additional shallow branch to preserve the local context of the input. They both have comparably less parameters than many offline (e.g. DeepLab [2] and PSPNet [22]) and real-time segmentation models (e.g. SegNet [1]) and also produce better results on the Cityscapes data set [4].

H. Yang et al. (Eds.): ICONIP 2020, CCIS 1332, pp. 386–393, 2020.
https://doi.org/10.1007/978-3-030-63820-7_44

2 Related Work

Semantic segmentation typically follows an encoder and decoder design. Existing DCNNs, such as VGG [17], ResNet [6,19], or MobileNet [14] are used as the backbone network. To perform semantic segmentation, the fully connected layer of DCNN is replaced by a convolution layer, for example in Fully Convolution Network (FCN) [16], UNet [13], SegNet [1], Bayesian Segnet [9], and Deeplapv3+ [3]. Further improvement can also be made with techniques to encode and manage global context, such as the pooling module (PPM) [22], atrous spatial pyramid pooling (ASPP) [16] and Xiphoid Spatial Pyramid Pooling [15]. Recently, semantic segmentation research has started to address real-time low-resource constraints, such as ENet [10], ICNet [21] and ContextNet [11]. Many methods trades real-time inference speed for lower segmentation quality. To address this, DSMRSeg [20] introduced a dual-stage feature pyramid network with multi-range context aggregation module to achieve high speed with high accuracy, although this model does not accept high-resolution input images. FAST-SCNN [12] was proposed to handle high resolution. Whilst it is promising for resource-constrained applications, its accuracy is lower than state-of-the-arts.

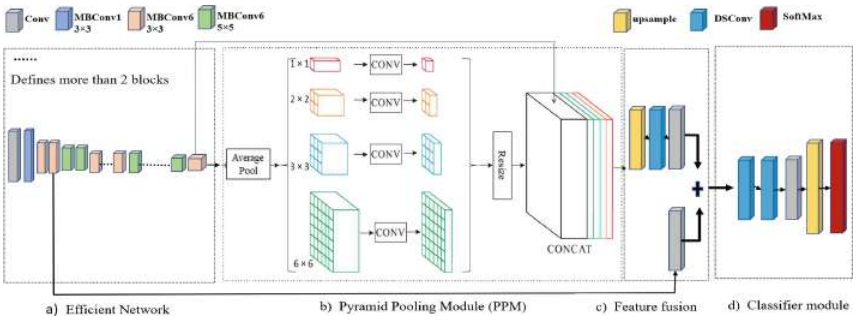

Fig. 1. Complete architecture of ESPNet

3 Proposed Methods

We propose the following two models- Base ESPNet and Final ESPNet. Our Base ESPNet is inspired by the encoder-decoder with skip connections [13,16]. In Final ESPNet, we introduce a new feature fusion module (FFM) after PPM to fuse local and global context. The detail is described next.

3.1 Network Architecture

The overall architecture of ESPNet is shown in Fig. 1 and consists of:

Backbone Network. We focus on MobileNet [7,14] to address resource constrained applications. We also adopt EfficientNet B0 [18] as our backbone network as it optimizes both accuracy and FLOPS. Its optimization objective is of the form $ACC(m) \times [FLOPS(m)/T]^w$ where $w = -0.07$ is a hyper-parameter to control the trade-off between accuracy ACC and FLOPS for model m. Figure 2 and Table 1 show the layer architecture of different MBConv blocks in EfficientNet B0.

The operations and connections of each block are determined by a per-block sub search space which involves the following: 1) Convolution operation: regular conv (Conv), depth-wise conv (DwConv), and mobile inverted bottleneck conv [14]; 2) Convolution kernel size: 3×3, 5×5; 3) Squeeze-and-excitation [8] ratio SERatio: 0, 0.25; 4) Skip operation: pooling, identity residual, or no skip; 5) Output filter size F_i; and 6) Number of layers per block N_i. Convolution operation, kernel size, SERatio, skip operation and F_i determine the architecture of a layer whereas N_i controls the repetition of a layer inside the block. For example, Fig. 2 shows that each layer of block 4 has an inverted bottleneck 3×3 convolution and an identity residual skip path. The same layer is repeated N_4 times inside block 4. In the proposed models, two types of MBConv blocks of different layer architecture (MBConv1 and MBconv6) are used. We also introduce squeeze-and-excitation optimization each MBConv block to improve channel inter-dependencies at almost no computational cost. The filter size of each block is defined as {0.75, 1.0, 1.20, 1.25} of the size of filter in each block of MobileNetV2.

Table 1. Stages of ESPNet

Stage (i)	Operators	Resolution	Channels	Layers
1	Conv, 3×3	256×256	32	1
2	MBConv1, k3 $\times$ 3	256×256	16	1
3	MBConv6, k3 $\times$ 3	128×128	24	2
4	MBConv6, k5 $\times$ 5	64×64	40	2
5	MBConv6, k3 $\times$ 3	32×32	80	3
6	MBConv6, k5 $\times$ 5	32×32	112	3
7	MBConv6, k5 $\times$ 5	16×16	192	4
8	MBConv6, k3 $\times$ 3	16×16	320	1
9	PPM	16×16	1600	1
10	FFM*	128×128	128	1
11	Classifier	512×512	20	1

*FFM is absent in base ESPNet.

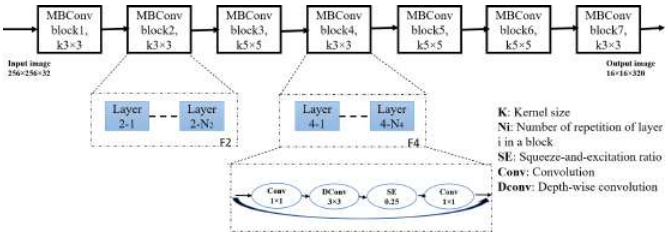

Fig. 2. Different Layers architecture of MBConv blocks

Shallow Branch. Inspired by down-to-sample technique in [12], we have introduced one shallow branch to our final ESPNet. The main motivation is to preserve the low-level features at high resolution. To ensure this, we establish this shallow connection after the stage 3 (Table 1) where image resolution becomes just a quarter of the original input size. As shown subsequently, this branch helps to improve the performance over the baseline model. Figure 1 shows the presence of this branch in final ESPNet.

Pyramid Pooling Module (PPM). The global contextual information is a key to successful semantic segmentation [5,11,22]. Therefore, a global scene-level reception is required by a deep network before sending the final feature map to the classifier module. We introduce PPM, which empirically proves to be an effective global contextual receptor. To reduce the loss of contextual information between different regions, our model separates the feature map into different subregions and forms pooled representation for different locations. After pooling, the feature maps of different sizes are fed into 1×1 convolution layer to reduce the dimension of context representation to $1/N$ of the original one where N represents the level size of pyramid. Then by re-sizing all low-dimension feature maps, we concatenate all to get the final global feature map (Fig. 1).

Feature Fusion Module. This section fuses the global and local feature maps to produce the final feature map. But due to lower size of global feature map, we use upsample method to scale up the resolution and then convolve both the features before final fusion. We adopt a simple addition technique like ICNet [21] and ContextNet [11] for fusion. Figure 1 shows the layer architecture of FFM.

Classifier. To assign a class to each pixel of an image, we design a classifier with two depth-wise separable convolutions (DSConv), one convolution (Conv), one upsampling and one softmax layer. DSConv not only convolves the input along all dimensions, but also reduces computational cost by reducing number of operations in convolution process. Here, we use 20 classes (including background) of the Cityscapes data set. Therefore, to generate a final output of 20 channels, we use one Conv layer followed a batch normalization and dropout layer. At last, we use softmax activation to generate final output of size $512 \times 512 \times 20$.

4 Experiment

Cityscapes Data Set. It is a large dataset for semantic understanding of urban street scenes. It consists of about 5000 fine and 20000 coarse annotated images. We used only the fine annotated images. Only the annotations for the training and validation sets are provided. We followed the standard split of the dataset.

Implementation. We conducted our experiments using a desktop computer with two Nvidia GeForce RTX 2080Ti GPU cards, each with 11GB GPU RAM. For parallel computing platform, we used `CUDA` 10.2. We developed our models based on `tensorflow` version 2.1 and `keras` 2.3.1. We also use the `horovod` framework to implement data-parallel distributed training. We set a batch size

of 4 and use stochastic gradient decent (SGD) as the model optimizer with a momentum of 0.9. Inspired by [2,7,22], we use the 'poly' learning rate scheme which computes the current learning rate (LR_{current}) as $LR_{\text{current}} = LR_{\text{base}} \times (1 - \text{iter}/\text{maxiter})^{\text{power}}$ where `iter` defines current iteration and `maxiter` defines maximum number of iterations in each epoch. We set LR_{base} to 0.045 and `power` to 0.9. This allow us to determine the optimal learning rate.

To overcome the limited training data, we apply various data augmentation techniques, such as random horizontal/vertical flip, random crop, re-sizing of image and many more. We use cross-entropy to calculate the model loss. Due to the limited size of physical memory of GPU cards, we use an input image size of 512×512 for training. Other existing models are also trained under the same configuration.

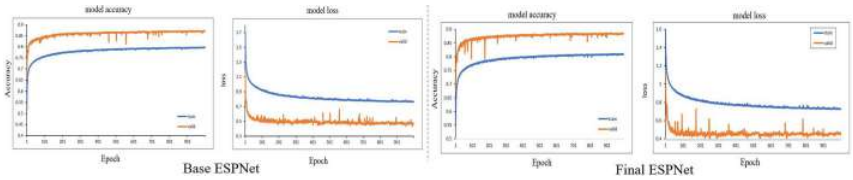

Fig. 3. Plot of model accuracy and loss

Model Evaluation. In this section, we study the performance of ESPNets on the Cityscapes data set. We consider both pixel-wise accuracy (Pi. Acc.) and mean of class-wise intersection over union (mIoU) on validation set. Figure 3 shows the performance of base and final ESPNet.

Table 2. Segmentation performance evaluation

Model	Input Size	Pi. Acc.	mIoU	Number of parameters (in Million)	Number of FLOPS (in Billion)	Train time per epoch (in Sec.)
Separable UNet	512×512	83%	29.5%	0.35	3.4	197
Bayesian SegNet	512×512	86.2%	49.4%	29.5	170.6	170
DeepLabV3+	512×512	72.9%	29.6%	37.7	33.4	53
FAST-SCNN	512×512	83.3%	43.3%	1.78	1.2	50
Base ESPNet S0	512×512	86.4%	55.1%	7.56	6.0	108
Base ESPNet S1	640×640	87.3%	58.3%	10.1	10.9	165
Base ESPNet S2	768×768	88.4%	59.5%	11.6	16.9	227
Final ESPNet	512×512	88.5%	60.8%	7.59	6.5	109

*All models are trained for 500 epochs under same system configuration.

Table 2 shows the quantitative performance of ESPNets after training all models for 500 epochs (no further improvement was observed for 1000 epochs). Here, we achieve 87.2% Pi. Acc. using base model and 88.5% Pi. Acc. using final model on the validation set. It clearly reflects that shallow branch in final model improves model performance by 1.3%. Likewise, the final model also achieves 60.8% val. mIoU which is 5.3% more than the base model mIoU. Table 2 also suggests that both proposed models provide the best overall balance between prediction time and segmentation accuracy. Whilst they are approximately as fast as FAST-SCNN, the performance in terms of mIoU is considerably superior. Note that offline training time is less critical in real-time applications.

For qualitative assessment, we generate the models' prediction on the validation and test set and show the prediction in Fig. 4. It can be seen that the edges of different objects such as car, bicycle, person, sidewalk are more accurately segmented by final ESPNet compared to baseline ESPNet. Even small tiny objects such as traffic signals are rightly detected by the final ESPNet whereas, base ESPNet misses few traffic signals.

Model Scaling. Our proposed model can be scaled up efficiently to adapt different resource constraints due to the scaling properties of EfficientNet. Doing so, we generate base ESPNet S0, S1 and S2 models which can tackle various input resolutions. Table 2 shows that S2 achieved 4.4% improvement on val. mIoU over S0 at the cost of 35% more FLOPS. Further scaling requires additional hardware resources and computational cost.

Performance Comparison. We compare the performance of some existing off-line and real-time segmentation models with ESPNets. To have a meaningful comparison among all, we trained FAST-SCNN, DeepLabV3+, Separable Unet, Bayesian SegNet under the same system configuration using 512×512px size of input. Note that the results in Table 2 are obtained after 500 epochs. Due to large size of parameters and FLOPS, DeepLabV3+ and Bayesian SegNet are used as off-line model whereas other models in Table 2 can be used for real-time computation. We replaced all standard Conv of UNet by DSConv layers to reduce computational cost, but performance of the model is still low. Comparably, FAST-SCNN performs better at full resolution whilst having less parameters. But if we compare FAST-SCNN's performance with our model under same configuration and input resolution, then both ESPNets perform better. For qualitative assessment, all models' prediction is shown in Fig. 5. The last two columns show the prediction by the proposed models. It can be seen that large objects such as road, vegetation, sky are segmented by both Separable UNet and DeepLabV3+. However, they fail to identify tiny objects such as pole, traffic lights whereas these tiny objects are not overlooked by FAST-SCNN and proposed ESPNets. Bayesian SegNet Identifies few tiny objects but fails to detect all and also unable to reconstruct few classes (e.g. sky). All these objects are correctly segmented by the ESPNets and FAST-SCNN. A closer inspection reveals that the segmentation quality of final ESPNet is better than that of FAST-SCNN: the edges of the objects are nicely segmented by final ESPNet and the appearance of tiny objects are generally sharper in the image.

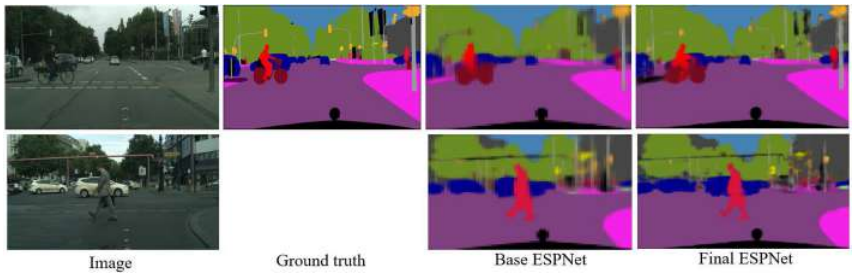

Fig. 4. Prediction by base and final ESPNet on validation (1st row) and test set (2nd row-no ground truth for test set)

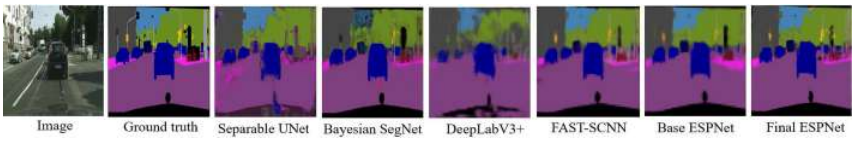

Fig. 5. Prediction by all models on validation set

5 Conclusion

We have proposed efficient semantic segmentation models that provide the best overall balance for resource-constrained applications. Addition of a global pyramid prior provides rich contextual information whereas a shallow branch enriches the model performance by providing a local context. We have shown that the performance of ESPNets outperforms other competitive segmentation models. In future, we are planning to evaluate our models on other public benchmarks. Our implementation is available at https://github.com/tanmaysingha/ESPNet.

Acknowledgement. The authors would like to acknowledge Pawsey supercomputing centre for providing J. Dunstan the internship during which part of the work was done.

References

1. Badrinarayanan, V., Kendall, A., Cipolla, R.: Segnet: a deep convolutional encoder-decoder architecture for image segmentation. TPAMI **39**(12), 2481–2495 (2017)
2. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. TPAMI **40**(4), 834–848 (2017)
3. Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation. In: Proceedings of ICCV, September 2018
4. Cordts, M., et al.: The cityscapes dataset for semantic urban scene understanding. In: Proceedings of CVPR, June 2016

5. He, K., Zhang, X., Ren, S., Sun, J.: Spatial pyramid pooling in deep convolutional networks for visual recognition. TPAMI **37**(9), 1904–1916 (2015)
6. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of CVPR, June 2016
7. Howard, A.G., et al.: Mobilenets: efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)
8. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: Proceedings of CVPR, June 2018
9. Kendall, A., Badrinarayanan, V., Cipolla, R.: Bayesian segnet: model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. arXiv preprint arXiv:1511.02680 (2015)
10. Paszke, A., Chaurasia, A., Kim, S., Culurciello, E.: Enet: a deep neural network architecture for real-time semantic segmentation. arXiv preprint arXiv:1606.02147 (2016)
11. Poudel, R.P., Bonde, U., Liwicki, S., Zach, C.: Contextnet: exploring context and detail for semantic segmentation in real-time. arXiv preprint arXiv:1805.04554 (2018)
12. Poudel, R.P., Liwicki, S., Cipolla, R.: Fast-scnn: fast semantic segmentation network. arXiv preprint arXiv:1902.04502 (2019)
13. Ronneberger, O., Fischer, P., Brox, T.: U-Net: convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) MICCAI 2015. LNCS, vol. 9351, pp. 234–241. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24574-4_28
14. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv 2: inverted residuals and linear bottlenecks. In: Proceedings of CVPR (June 2018)
15. Shang, Y., Zhong, S., Gong, S., Zhou, L., Ying, W.: DXNet: an encoder-decoder architecture with XSPP for semantic image segmentation in street scenes. In: Gedeon, T., Wong, K.W., Lee, M. (eds.) ICONIP 2019. CCIS, vol. 1143, pp. 550–557. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-36802-9_59
16. Shelhamer, E., Long, J., Darrell, T.: Fully convolutional networks for semantic segmentation. TPAMI **39**(4), 640–651 (2017)
17. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
18. Tan, M., Le, Q.V.: Efficientnet: Rethinking model scaling for convolutional neural networks. arXiv preprint arXiv:1905.11946 (2019)
19. Targ, S., Almeida, D., Lyman, K.: Resnet in resnet: generalizing residual architectures. arXiv preprint arXiv:1603.08029 (2016)
20. Yang, M., Shi, Y.: DSMRSeg: dual-stage feature pyramid and multi-range context aggregation for real-time semantic Segmentation. In: Gedeon, T., Wong, K.W., Lee, M. (eds.) ICONIP 2019. CCIS, vol. 1142, pp. 265–273. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-36808-1_29
21. Zhao, H., Qi, X., Shen, X., Shi, J., Jia, J.: ICNet for real-time semantic segmentation on high-resolution images. In: Proceedings of ECCV, September 2018
22. Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. In: Proceedings of CVPR, July 2017

.2 Publication 2

FANet: Feature Aggregation Network for Semantic Segmentation

Tanmay Singha
School of EECMS
Curtin University
Perth, Australia
tanmay.singha@postgrad.curtin.edu.au

Duc-Son Pham
School of EECMS
Curtin University
Perth, Australia
dspham@ieee.org

Aneesh Krishna
School of EECMS
Curtin University
Perth, Australia
a.krishna@curtin.edu.au

2020 Digital Image Computing: Techniques and Applications (DICTA) | 978-1-7281-9108-9/20/$31.00 ©2020 IEEE | DOI: 10.1109/DICTA51227.2020.9363370

***Abstract*—Due to the rapid development in robotics and autonomous industries, optimization and accuracy have become an important factor in the field of computer vision. It becomes a challenging task for the researchers to design an efficient, optimized model with high accuracy in the field of object detection and semantic segmentation. Some existing off-line scene segmentation methods have shown an outstanding result on different datasets at the cost of a large number of parameters and operations, whereas some well-known real-time semantic segmentation techniques have reduced the number of parameters and operations in demand for resource-constrained applications, but model accuracy is compromised. We propose a novel approach for scene segmentation suitable for resource-constrained embedded devices by keeping a right balance between model architecture and model performance. Exploiting the multi-scale feature fusion technique with accurate localization augmentation, we introduce a fast feature aggregation network, a real-time scene segmentation model capable of handling high-resolution input image (1024 $\times$ 2048 px). Relying on an efficient embedded vision backbone network, our feature pyramid network outperforms many existing off-line and real-time pixel-wise deep convolution neural networks (CNNs) and produces 89.7% pixel accuracy and 65.9% mean intersection over union (mIoU) on the Cityscapes benchmark validation dataset whilst having only 1.1M parameters and 5.8B FLOPS.**

***Index Terms*—Semantic segmentation, BiFPN, MobileNet.**

I. Introduction

Semantic segmentation is one of the most challenging tasks in computer vision. It aims to assign a class/label to each pixel of an input image. These classes/labels are defined by the training set and could be anything such as car, building, people, bicycle, train, and many more. By clustering parts of the image together based on same object of interest, it identifies regions of different objects in the scene. Thus, it opens the door for developing numerous real-time applications specially in the field of autonomous driving, robotics, virtual reality and video surveillance.

Over the decade, innumerable CNN models [1], [2], [3] have been proposed to generate a segmentation map for an entire image in a single forward pass. Due to the high robustness to variance in scale and handling capability of rich semantics, CNNs are widely used for object detection, instance and scene segmentation. To meet the growing demand of multi-scale testing, it becomes essential to improve prediction accuracy of these models [4], [5]. To address this, feature pyramid network (FPN) was introduced in [6]. It utilizes an in-network feature hierarchy architecture and introduces a top-down pathway to achieve multi-scale feature fusion. Many object detection and instance segmentation models adopt FPN to achieve high accuracy. Later on, PANet [7] was introduced to enhance the entire feature hierarchy of FPN. It resolves the issue of localizing the signals in the lower layers of FPN by introducing a bottom-up path augmentation. Inspired by these two approaches, recently Google Brain has come up a new technique, called Bi-directional Feature Pyramid Network (Bi-FPN) for object detection [8]. It optimizes PANet feature fusion approach and introduces few skip connections to enhance feature maps' quality at different levels. These multi-scale techniques are vastly used for object detection, but not for semantic segmentation. Inspired by these multi-scale fusion approaches, we introduce a modified Bi-FPN technique in our proposed scene segmentation model. This modified design also eliminates the need for global contextual prior, thus optimizing the overall model architecture.

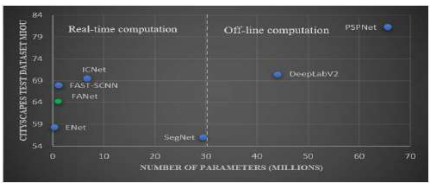

Fig. 1. **Number of parameters vs cityscapes mIoU**- Our model generates less parameters compare to others and produces better accuracy than ENet and SegNet. It also generates lesser FLOPs than all models (see Table VI)

To generate rich semantic features for multi-scale feature fu-

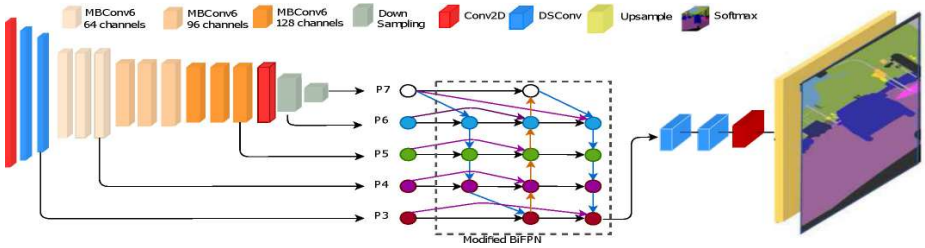

Fig. 2. Complete architecture of FANet

sion in scene segmentation, we need an efficient and optimized feature extractor. Many segmentation models use ResNet as a backbone due to its high scalable nature. It can be scaled up from ResNet-18 to ResNet-200 [9] to improve model performance, but the number of parameters and FLOPS also increase with the model size. Various segmentation models such as DeepLabV3+ [2], EfficientNets [10], Fast-SCNN [11] use MobileNetV2 [12] as feature extractor due to its high scalabilty and optimized features. Inspired by MobileNetV2 and FAST-SCNN, we use mobile inverted residual bottleneck blocks to design the backbone network of our model. We also use depth-wise convolution (DsConv) to optimize the number of operations and parameters of convolution layers, thus we strike a right balance between the model architecture and model performance. Hence, we design and propose an optimized scene segmentation model for resource-constrained computer vision devices.

II. Related Work

Semantic segmentation models typically follow an encoder-decoder architecture. In the encoder stage, a deep CNN typically computes a feature hierarchy layer by layer and develops an inherent multi-scale pyramid shape, whereas at the decoder end, high-semantic feature map is up-sampled and fused with the previous layer feature map through lateral connections to recover higher spatial dimensions. After extracting spatial details, model predicts the class for each pixel to complete the segmentation process. Different networks such as VGG [13], ResNet [9], Xception [14], or MobileNet [15] are often used as the encoder.

The Fully Convolutional Network (FCN) [16] has shown a revolutionary approach for all modern CNNs by adopting an encoder-decoder architecture. It replaces the fully connected layers from the top of the encoder by convolution one to generate a spatial map instead of classification scores. Based on this foundation, several models such as UNet [17], RefineNet [18], DeepLabV3+ [19] are then developed by exploiting the lateral connection between the low-level feature maps across resolutions and semantic levels. Inheriting the benefits of multi-scale features fusion, several other approaches (PSP-Net [20], DeepLab [2], ParseNet [21]) are also developed by utilizing a pyramid pooling module (PPM) [20] or an atrous spatial pyramid module [22] as a global contextual prior. All these techniques have shown that multi-scale feature fusion plays an important role in improving the prediction. However, fusing different feature maps of different spatial dimensions often introduces a large semantic gap caused by different depths of layers. To address this issue, FPN [6] and PANet [7] were introduced for fusing semantic features of all semantic labels. In contrast to these top-down and bottom-up approaches, the Google Brain team introduced another multi-scale feature fusion technique, called NAS-FPN [23] by leveraging the neural architecture search for automatic design of feature network topology. But due to long search time and unpredictable network architecture, this technique is not suitable for embedded devices. Recently, another efficient, scalable object detection model, named EfficientDet [8] has been introduced by the Google Brain team, based on a new feature-fusion technique, called Bi-directional Feature Pyramid Network (Bi-FPN). Bi-FPN is an optimized version of PANet and it produces better results compared to existing multi-scale feature fusion techniques.

Since, the demand for high-performing real-time methods is increasing rapidly, so several research works have been conducted in the field of semantic segmentation to target resource-constrained embedded devices. SegNet [1] is one of the pioneering models targeting real-time computation. Later on, ENet [24], ICNet [25], ContextNet [26], BiSeNet [27] were introduced to improve the performance in a real-time environment. All these models generate moderate results but still could not process high-resolution images quickly enough due to their large number of FLOPS and parameters. By keeping a balance between model size and model performance, FAST-SCNN [11] attempted to address this issue. It can process high-resolution input images quickly in real time and produces better segmentation results. Later on, another model, called DFANet [28] is introduced for real-time scene segmentation. Though the model claims to achieve 70.3% mIoU on Cityscapes test set, but it has more than 6 times parameters and FLOPS compared to FAST-SCNN and cannot process full resolution of Cityscapes images.

Therefore, inspired by the simple design of FAST-SCNN,

we develop a new model by exploiting optimized architecture of the residual block of MobileNetV2. To reduce the large semantic gap between the spatial dimensions of low-feature map and global feature map, we also incorporate a new multi-scale feature fusion approach. We detail our approach in the next section.

III. Proposed Methods

In this section, we propose an optimized and efficient network for semantic segmentation, call FANet, capable of handling high-resolution images and producing a quality output at a lower computational cost than many existing alternatives .

A. Network Architecture

The overall architecture of the proposed model is shown in Fig.2. In the following subsections, we discuss the backbone network, the modified Bi-FPN and classifier module in detail.

1) Backbone Network: Since our main focus is to design an optimized model capable of handling full-resolution images with higher accuracy in real time, we decide to employ the MobileNetV2 architecture [12]. Specifically, we use three bottleneck blocks, each repeating three times. Similar to the residual block, each bottleneck block contains an input followed by several bottlenecks, then followed by expansion. The expansion ratio, ratio between the size of the input bottleneck and the inner size, is 6. The basic implementation structure of MBConv6 is demonstrated in Table I. Here, h, w, c and c' denote the spatial dimensions of tensors, t defines the expansion ratio and s defines the stride. Note that we use ReLU non-linearity in the first two layers because of its robustness. However, we do not use it after the last layer to prevent non-linearities from destroying meaningful information. We provide the similar layered architecture to each MBConv6 block to make our design uniform and simple. Based on the size of the filter in each block of MobileNetV2, we set the filter size of each MBConv6 block. Table II shows that we use 64 channels for first bottleneck block, 96 for second and 128 for third block.

TABLE I
Bottleneck residual block

Input	Operator	Output
$h \times w \times c$	1×1 Conv,1/1, Relu	$h \times w \times tc$
$h \times w \times tc$	3×3 DwConv, 3/s, Relu	$h/s \times w/s \times tc$
$h/s \times w/s \times tc$	1×1 Conv,1/1, -	$h/s \times w/s \times c'$

Inspired by FAST-SCNN, we introduce three layers at the beginning of the residual blocks to down sample the input. The first layer is a standard convolution layer (Conv) and remaining two layers are depth-wise separable convolution layers (DSConv). Although, DSConv optimizes number of operations and parameters, we employ Conv at the first stage due to less channels (only three) of the input image which makes the use of DSConv insignificant. As the low-dimensional input subspace produces rich semantic features, we introduce two max-pooling layers on top of residual blocks to reduce the spatial dimensions of feature map and create two additional stages for multi-scale feature fusion. Note that adding these down-sampling layers does not increase computational cost.

TABLE II
Layer architecture of backbone network

Stage (i)	Input	Operators	Layers (n)	Output
1	1024×2048×3	Conv, k3×3	1	512×1024×32
2	512×1024×32	DSConv, k3×3	1	256×512×48
3	256×512×48	DSConv, k3×3	1	128×256×64
4	128×256×64	MBConv6, k3×3	3	64×128×64
-	64×128×64	MBConv6, k3×3	3	32×64×96
-	32×64×96	MBConv6, k3×3	3	32×64×128
5	32×64×128	Conv, k1×1	1	32×64×64
6	32×64×64	MaxPooling	1	16×32×64
7	16×32×64	MaxPooling	1	8×16×64

The feature maps of the same spatial sizes produced by different layers fall under the same stage. Thus, we generate seven stages in backbone network and it reflects in Fig.2. Note that the original input size should be divisible by 2^7. We use semantic features of P3, P4, P5, P6, P7 levels for feature fusion. A complete description of our multi-scale feature fusion technique is given in the next sub-section.

2) Modified Bi-FPN: Fig. 3 shows the design of different features scaling techniques. FPN (a) introduces the concept of multi-scale feature fusion by setting a top-down pathway, whereas PANet (b) suggests an additional bottom-up path augmentation for preserving the local context. By exploiting neural architecture search, recently a new model, called NAS-FPN (c), is introduced for object detection. Although it optimizes a large number of operations, it is difficult to interpret the feature network due to long neural network search time. More recently, EfficientDet introduces Bi-FPN (d) technique for object detection which optimizes several cross-connections of PANet and introduces lateral connections between the input to output node if they are at the same level. Thus, it improves network performance. Inspired by this approach, we design a modified version of Bi-FPN technique (e). We introduce a new top-down path augmentation for feature aggregation and produce final semantic features at the finest level. We also employ few lateral connections between the input to output node at same label to enhance the model performance.

Formally, given a list of semantic features F^{in} = ($F^{in}_{l_1}$, $F^{in}_{l_2}$,..., $F^{in}_{l_i}$) generated by different layer stages(P3 to P7), our aim is to find a transformation f that can effectively map different features at different semantic levels and produce a rich multi-scale semantic features F^{out} = ($F^{out}_{l_1}$, $F^{out}_{l_2}$,..., $F^{out}_{l_i}$) at all labels. Finally, all contextual feature maps at different labels will be aggregated to generate final global feature map at the finest level: $FF_{out} = F^{in}_{l_1} + \sum f(F^{out})$. Fig. 2 shows a graphical representation of our multi-scale feature fusion technique. Features from P3 to P7 stages are taken for feature fusion. First, a top-down approach is employed to fuse global features with local features. Secondly, a bottom-up path augmentation technique is implemented to localize

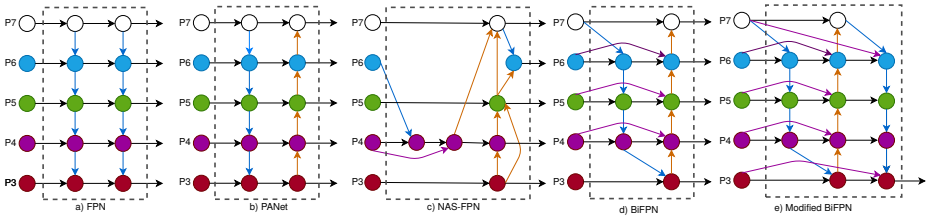

Fig. 3. Design of different feature networks. a) Feature Pyramid network - introduces a top-down pathway for multiscale feature fusion; b)Path Aggregation Network introduces a bottom-up pathway for better localisation; c) Neural Architecture Search FPN uses neural search to find out irregular feature network topology; d) Bidirectional FPN optimises PANet and introduces skip connection; d) Modified BFPN introduces an additional top-down pathway and two lateral connections for feature aggregation and better prediction

each objects of the scene and generate a list of contextual feature maps at different labels($F_{l_3}^{out}$, $F_{l_4}^{out}$, $F_{l_5}^{out}$, $F_{l_6}^{out}$ and $F_{l_7}^{out}$). The size of each feature map in each successive stage is reduced to 1/2 of its previous stage size. For example, at P3, the feature size is 128 $\times$ 256px whereas, at stage P4, the size is 64 $\times$ 128px. Once the rich contextual features are generated, we generate the final aggregated feature map at the finest level through a top-down path augmentation.

$$F_{l_7}^{out} = Conv(F_{l_7}^{out}) \tag{1}$$

$$F_{l_6}^{out} = Conv(F_{l_6}^{out} + Upsample(F_{l_7}^{out}) + Upsample(F_{l_7}^{in})) \tag{2}$$

$$F_{l_5}^{out} = Conv(F_{l_5}^{out} + Upsample(F_{l_6}^{out})) \tag{3}$$

$$F_{l_4}^{out} = Conv(F_{l_4}^{out} + Upsample(F_{l_5}^{out})) \tag{4}$$

$$F_{l_3}^{out} = Conv(F_{l_3}^{in} + F_{l_3}^{out} + Upsample(F_{l_4}^{out})) \tag{5}$$

3) Classifier module: The purpose of adding this module on top of the model is to assign a class to each pixel of feature map. To keep the model design simple, we use only two depth-wise separable convolutions, one standard convolution, one Upsample and one softmax layer. Depth-wise separable convolution layer convolves the feature map along all dimensions by optimizing the number of parameters and FLOPs, whereas Upsample layer recovers the spatial dimensions of feature map without increasing computational cost. Finally, by using softmax activation, a multinomial logistic regression function, our model assigns a class to individual pixel. Thus, segmented output is predicted by the model. In this research, we use 19 classes (excluding background) out of 30 classes of the Cityscapes dataset.

IV. Experiment

We trained the proposed FAST-FANet model with the Cityscapes training set and evaluated its performance on the validation and test sets. We also compared model performance with few off-line (DeepLabV3+, Bayesian SegNet) and real-time (FAST-SCNN) segmentation models under same configuration. Details are given in the following.

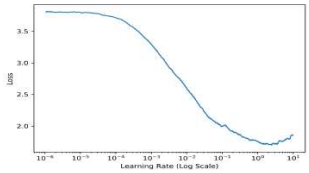

Fig. 4. Finding optimal learning rate

A. Implementation Details

To conduct this experiment, we use a dual Nvidia GeForce RTX 2080Ti GPUs system, each GPU has 11GB of memory. To exploit the parallel processing power of GPUs, we use CUDA 10.2. Our proposed model is developed using `tensorflow` 2.1.0 and `keras` 2.3.1. To utilize both GPUs in data-parallel distributed training environment, we employ the `horovod` framework [29]. It takes a single-GPU `tensorflow` program and trains it on multiple GPUs. For instance, in this research, `horovod` divides the whole training set into two sets and runs the training script on individual set in each GPU. Thus, it boosts the run-time performance by effectively utilizing all resources. We use stochastic gradient decent (SGD) as the model optimizer with 0.9 momentum.

Inspired by [2], [15], [20], we use 'poly' learning rate by setting 0.045 as base value and 0.9 as power. To find out the optimal learning rate in each epoch during training the model, we train our model for 5 epochs using polynomial scheduler and plot model loss against learning rate. Thus, we set upper and lower bound of learning rate for training. Fig. 3 illustrates the plot of learning rate vs model loss as an example. To calculate model loss, we use categorical cross-entropy.

Following the suggestion by MobileNetV2, we use 0.00004 as l_2 regularization for top layers of the model except depth-wise convolution. To avoid overfitting due to limited data, we apply various data augmentation techniques such as random horizontal flip, vertical flips, random crop, resizing and adjusting brightness, saturation and contrast of images. We also use a dropout layer just before the softmax layer. It also can

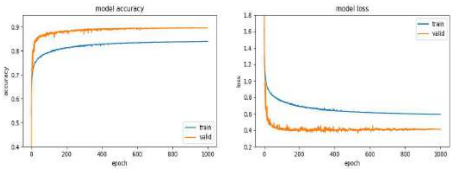

Fig. 5. Plot of model accuracy and loss against epoch

regulate model over-fitting issue and can improve validation accuracy. After training the model several times with different dropout rates, we noticed that a dropout rate less than 0.2 cannot address the overfitting issue whilst a value more than 0.6 value causes underfitting. Therefore, we set the average of these two values as dropout rate.

B. Dataset

We used Cityscapes [30] for evaluation. It is a large-scale dataset, mainly used for object detection, instance and semantic segmentation. It provides annotated data for 30 classes grouped into 8 categories. The dataset consists of around 5000 fine annotated images and 20000 coarse annotated ones.

In the experiments, we used only fine annotated images and considered only 19 classes for pixel annotations. The whole dataset is divided into three parts- training set (2,975 images), validation set (500 images) and test set 1,525 images). The labels for the training set and validation set are given by Cityscapes whereas the test labels are not provided by the benchmark. However, the prediction on the test set can be submitted to the Cityscapes server for evaluation.

We also used the CamVid dataset [31] for performance evaluation. This dataset is mainly designed for object detection in automated driving vehicle. The images of this dataset were generated from a recorded video, then annotated frames were created by assigning a class colour to each object of the frame by human operators. This dataset contains 267 images for training, 101 for validation and 233 for testing. Out of 32 classes, we used 12 classes (including void) for performance evaluation. Due to the small size of the dataset, models were under-learned. We used data augmentation to address this issue.

C. Model Evaluation

In this section, we illustrate the performance of FANet on the Cityscapes dataset for semantic segmentation. Pixel accuracy (Pi. Acc.), Class mean Intersection Over Union (mIoU) and category mIoU on validation and test sets are measured. We train our proposed model with different input resolutions for 1000 epochs. Fig. 5 shows the model accuracy and loss against the number of epochs on training and validation sets. It demonstrates that the model reached its saturation point after 800-900 epochs and obtained 89.7% pixel accuracy on the validation dataset which is much better than many existing models. Model also achieves 65.9% mIoU on validation set. We also performed an ablation study to verify our model's performance. We used FPN, BiFPN and modified BiFPN with our light-weighted backbone network and documented model performance in Table V. It can be observed that additional feature aggregation path and few lateral connections of the modified BiFPN technique enhance model's performance by 0.9%. Proposed FANet with modified BiFPN has a total of 1.1M parameters and 5.8B FLOPs which are comparably lesser than many existing real-time scene segmentation models. We also evaluated our model performance in terms of class-wise prediction accuracy at different input resolutions and presented the results in Table III as ready reference. From Tables III and IV, it is observed that the model prediction accuracy is almost 90% in most of the object categories such as flat, construction, nature, sky and vehicle, whereas in object and human categories, the model performance is lower due to the tiny size of the objects in these categories. For motorcycle and truck, performance is less than 50% due to the lack of training data available in these two classes. The same phenomenon is also observed with other models. The results for category-wise mIoU on the validation set are exhibited in Table IV: our model achieves 83.6% accuracy which is better than many real-time scene segmentation models such as SegNet, ENet and ContextNet.

D. Performance Comparison

To compare FANet performance with other existing models, we train few off-line and real-time segmentation models under same configuration with full input resolution and 4 batch size. Due to the hardware limitation, we could not set higher batch size for the experiment. Results of comparison on validation set are displayed in Table V. Due to large size of FLOPS, separable UNet, deepLab and Bayes-segnet are used as off-line segmentation model, where as FAST-SCNN is used as real-time computational model. As per the literature, FAST-SCNN produces 68.6% validation accuracy, but our experiment has achieved at most 63.3% accuracy on cityscaapes validation set and 63% on test set. We could not verify our FAST-SCNN implementation due to the unavailability of official GitHub implementation. But to our understanding, we have achieved best performance of FAST-SCNN compared to all available unofficial GitHub implementations. Therefore, our experiment shows that proposed model has achieved better accuracy (65.9%) than the FAST-SCNN. Based on other parameters such as FLOPS, number of parameters, model size, our proposed model out performs other models. We have also calculated inference time for all the models for 500 validation images and Table V shows that FANet takes less time to predict all the images. Hence, we can conclude that overall FANet performs well in real-time environment.

We have also compared the prediction on the test set as evaluated by the Cityscapes server. The result of comparison is displayed in Table VI. To compare overall performance of all models, we calculated FLOPS and number of parameters of each model listed in the Table VI. From this table, it can be observed that DeepLab and PSPNet produce best

TABLE III
FANET PERFORMANCE ON VALIDATION SET AT DIFFERENT INPUT RESOLUTIONS

Input Size	Road	S.walk	Build.	Wall	Fence	Pole	T.light	T.sign	Veg.	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	M.cycle	Bicycle	mIoU
1024×2048	96.2	75.1	89.3	52.7	47.3	47.0	53.5	64.6	89.8	55.2	92.5	70.2	45.6	90.9	47.0	70.4	61.5	38.8	65.2	65.9
768×1536	95.5	73.6	88.2	42.5	41.1	45.6	49.8	61.3	89.7	53.3	90.8	68.5	41.2	89.6	43.5	63.9	48.2	35.5	62.8	62.3
512×1024	95.4	71.8	87.1	34.9	36.8	44.2	43.6	58.0	89.4	56.5	90.8	66.7	38.3	88.1	41.5	55.8	41.4	32.1	61.9	59.7

TABLE IV
CATEGORY-WISE FANET PERFORMANCE ON CITYSCAPES VALIDATION DATASET

Input size	Flat	Construction	Object	Nature	Sky	Human	Vehicle	mIoU
1024×2048	96.7	89.4	55.6	90.1	92.5	71.5	89.2	83.6
768×1536	96.1	88.5	52.8	89.7	90.8	70.4	87.6	82.3
512×1024	96.1	87.7	50.6	89.3	90.8	68.9	86.5	81.4

TABLE V
PERFORMANCE EVALUATION OF DIFFERENT ON CITYSCAPES VALIDATION SET

Model	Class mIoU	Category mIoU	Number of parameters (in Million)	Number of FLOPS (in Billion)	Run-time per epoch	Inference time for validation set (sec.)	Model Size (MB)
Separable UNet	29.6%	62.3%	0.35	27	754	129	3.1
Bayes-SegNet	56.8%	77.2%	29.5	2720	672	214	225.4
DeepLab	58.2%	79.3%	37.9	1575	728	123	90.2
FAST-SCNN	63.3%	82.2%	1.2	7.7	258	120	9.4
FANet (with FPN)	**62.6%**	**81.3%**	**1.2**	**5.9**	**247**	**117**	**9.5**
FANet (with BiFPN)	**65.0%**	**82.5%**	**1.1**	**5.8**	**241**	**113**	**9.4**
FANet (with modified BiFPN)	**65.9%**	**83.6%**	**1.1**	**5.8**	**241**	**113**	**9.4**

*All models are trained with input of size 1024×2048×3 and batch size 4.

TABLE VI
PERFORMANCE EVALUATION OF DIFFERENT MODELS ON CITYSCAPES TEST SET

Model	Class mIoU	Category mIoU	Number of parameters (in Million)	Number of FLOPS (in Billion)
DeepLabV3+ [19]	82%	91.6%	37.7	267
DeepLabV2 [2]	70.4%	86.4%	37.9	1575
PSPNet [20]	81.2%	90.6%	65.5	516
SegNet extended [1]	56.1%	79.8%	29.5	1365
ENet [24]	58.3%	80.4%	0.4	19
ICNet [25]	69.5%	–	6.7	30
FAST-SCNN [11]	68%	84.7%	1.2	7.7
FANet	**64.1%**	**83.1%**	**1.1**	**5.8**

*empty cell means that data is missing on evaluation server.

accuracy on test set, but at the cost of large FLOPS and parameters. Due to the large size of backbone network, these models are suitable for off-line segmentation. For real-time scene segmentation, we need an optimized and efficient model which can work fast in real-time environment and requires less memory footprint. Among all the models, ENet has much less parameters. However, it has 19B FLOPS due to its multi-branch approach which slows down model performance and it also produces low prediction accuracy. On the other hand, ICNet generates a better accuracy (69.5%) among all the real-time segmentation models, but it has 30B FLOPS and 6.7M parameters. Compared to all other models, our proposed FANet has less parameters (1.1M) and FLOPS (5.8B) and also produces 64.1% class mIoU on the test set.

We also evaluated our proposed model on the CamVid dataset. For a comprehensive analysis, we compared our model performance with some existing real-time segmentation models. The results are readily shown in Table VII. Models with

TABLE VII
EVALUATION RESULTS ON VALIDATION SET OF CAMVID DATASET

Model	input size	Class mean IoU	Pi. Acc.
SegNet* [1]	360×480	55.6%	–
Enet* [24]	360×480	51.3%	–
FAST-SCNN [11]	512×1024	57.5%	86.9%
FANet	**512×1024**	**57.8%**	**87.1%**

*empty cell means missing value in literature.

* sign are not trained by us. The results for these models are extracted from the literature. It is shown that FANet produces better class mIoU and Pi. accuracy on the CamVid validation set. It also demonstrates that FAST-SCNN performance is almost similar to FANet on CamVid dataset. A same observation is also made made after analyzing the predicted images by these two models. For qualitative assessment, we compare the prediction on the Cityscapes validation set and present it in Fig. 6. The second column shows the ground-truth of the

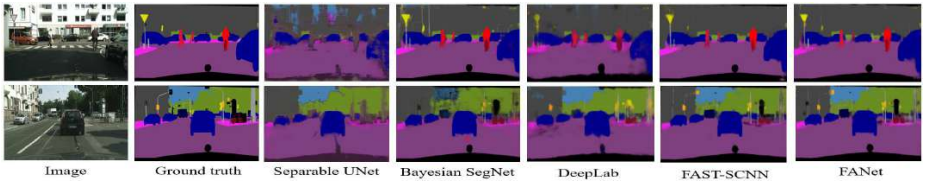

Fig. 6. All models (mentioned in table III) prediction on Cityscapes validation set

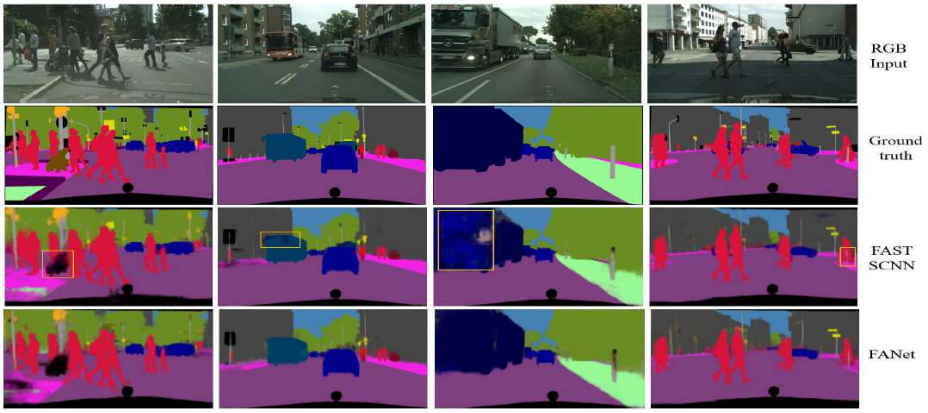

Fig. 7. Prediction samples by FAST-SCNN and FANet on Cityscapes validation set

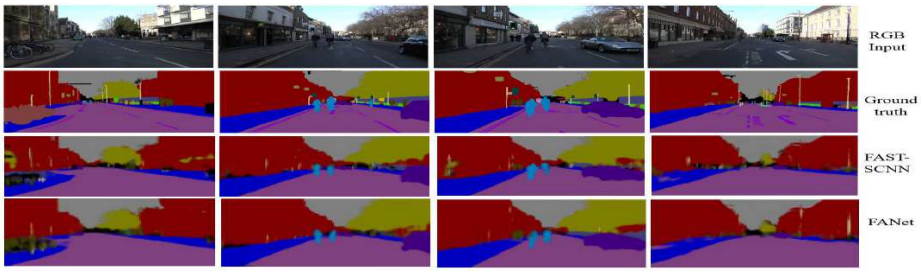

Fig. 8. FANet predictions on CamVid Validation Set

original images. It can be clearly identified that quality of the predicted images by FANet is better than other models. Especially, the edges of objects are more properly segmented and most of the tinny objects (traffic sign, traffic lights) are identified more clearly. We also observed that the quality of predicted images of FAST-SCNN is better compared to other models. We separately compared FANet's performance with FAST-SCNN and exhibit their prediction in Fig. 7. Though the quality of images by both models are almost similar but it can be observed that FAST-SCNN occasionally assigned incorrect labels to some pixels. This could be due to a large semantic gap between the local feature and global feature maps. In FANet, this gap is reduced by the multi-scale feature fusion technique which produces better predicted images overall.

Fig. 8 also shows predicted images by FAST-SCNN and FANet on CamVid dataset which appear almost same. However, the prediction quality by both models are not satisfactory due to the lack of training images which is an inherent issue with this dataset.

V. Conclusion

We have proposed an efficient and optimized semantic segmentation model which can handle high-resolution input images and can quickly produce output in real time with low computational cost. Due to its optimized structure and the ability to capture contextual information by a new feature scaling technique, it outperforms many existing real-time semantic segmentation models. We also demonstrate that our multi-scale feature fusion technique reduces the semantic gap between global feature and local feature and also substitutes the need for global contextual prior. In the future, we plan to evaluate FANet performance on COCO dataset. Our implementation of FANet is available at https://github.com/tanmaysingha/FANet.

References

[1] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *TPAMI*, vol. 39, no. 12, pp. 2481–2495, 2017.

[2] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. CVPR*, June 2015.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, 2016, pp. 770–778.

[5] A. Shrivastava, A. Gupta, and R. Girshick, "Training region-based object detectors with online hard example mining," in *Proc. CVPR*, 2016, pp. 761–769.

[6] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. CVPR*, 2017, pp. 2117–2125.

[7] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *Proc. CVPR*, 2018, pp. 8759–8768.

[8] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," *ArXiv*, vol. abs/1911.09070, 2019.

[9] S. Targ, D. Almeida, and K. Lyman, "Resnet in resnet: Generalizing residual architectures," *arXiv preprint arXiv:1603.08029*, 2016.

[10] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *arXiv preprint arXiv:1905.11946*, 2019.

[11] R. P. Poudel, S. Liwicki, and R. Cipolla, "Fast-scnn: fast semantic segmentation network," *arXiv preprint arXiv:1902.04502*, 2019.

[12] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proc. CVPR*, 2018, pp. 4510–4520.

[13] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[14] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. CVPR*, 2017, pp. 1251–1258.

[15] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[16] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Annals of the History of Computing*, no. 04, pp. 640–651, 2017.

[17] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Proc. MICCAI*. Springer, 2015, pp. 234–241.

[18] G. Lin, A. Milan, C. Shen, and I. Reid, "Refinenet: Multi-path refinement networks for high-resolution semantic segmentation," in *Proc. CVPR*, 2017, pp. 1925–1934.

[19] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proc. ICCV*, September 2018.

[20] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proc. CVPR*, 2017, pp. 2881–2890.

[21] W. Liu, A. Rabinovich, and A. C. Berg, "Parsenet: Looking wider to see better," *arXiv preprint arXiv:1506.04579*, 2015.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[23] G. Ghiasi, T.-Y. Lin, and Q. V. Le, "Nas-fpn: Learning scalable feature pyramid architecture for object detection," in *Proc. CVPR*, 2019, pp. 7036–7045.

[24] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "Enet: A deep neural network architecture for real-time semantic segmentation," *arXiv preprint arXiv:1606.02147*, 2016.

[25] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, "Icnet for real-time semantic segmentation on high-resolution images," in *Proc. ECCV*, 2018, pp. 405–420.

[26] R. P. Poudel, U. Bonde, S. Liwicki, and C. Zach, "Contextnet: Exploring context and detail for semantic segmentation in real-time," *arXiv preprint arXiv:1805.04554*, 2018.

[27] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "Bisenet: Bilateral segmentation network for real-time semantic segmentation," in *Proc. ECCV*, 2018, pp. 325–341.

[28] H. Li, P. Xiong, H. Fan, and J. Sun, "Dfanet: Deep feature aggregation for real-time semantic segmentation," in *Proc. CVPR*, 2019, pp. 9522–9531.

[29] A. Sergeev and M. Del Balso, "Horovod: fast and easy distributed deep learning in tensorflow," *arXiv preprint arXiv:1802.05799*, 2018.

[30] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. CVPR*, June 2016.

[31] G. J. Brostow, J. Fauqueur, and R. Cipolla, "Semantic object classes in video: A high-definition ground truth database," *Pattern Recognition Letters*, vol. 30, no. 2, pp. 88–97, 2009.

.3 Publication 3

A Lightweight Multi-scale Feature Fusion Network for Real-Time Semantic Segmentation

Tanmay Singha[1( )], Duc-Son Pham[1], Aneesh Krishna[1], and Tom Gedeon[2]

[1] School of Electrical Engineering, Computing, and Mathematical Sciences, Curtin University, Bentley, WA 6102, Australia
tanmay.singha@postgrad.curtin.edu.au

[2] Research School of Computer Science, The Australian National University, Canberra, Australia

Abstract. Recently, semantic segmentation has become an emerging research area in computer vision due to a strong demand for autonomous vehicles, robotics, video surveillance, and medical image processing. To address this demand, several real-time semantic segmentation models have been introduced. Relying on existing Deep Convolution Neural networks (DCNNs), these models extract contextual features from the input image and construct the output at the decoder end by simply fusing deep features with shallow features which causes a large semantic gap. However, this large gap causes boundary degeneration and noisy feature effects in the output. To address this issue, we propose a novel architecture, called Feature Scaling Feature Fusion Network (FSFFNet) which alleviates the gap by successively fusing features at consecutive levels in multiple directions. For better dense pixel-level representation, we also employ a feature scaling technique which helps the model assimilate more contextual information from the global features and improves model performance. Our proposed model achieves 71.8% validation accuracy (mIoU) on the Cityscapes dataset whilst having only 1.3M parameters.

Keywords: Semantic segmentation · Feature scaling · Feature fusion · Deep learning · Deep neural networks · Real-time applications

1 Introduction

Semantic scene segmentation is an important task in many applications such as medical image processing, autonomous vehicles, and damage detection. Previous studies [5,7–9,17] have shown the wide application of Deep Convolutional Neural Networks (DCNNs) in different computer vision tasks. However, semantic segmentation is still a challenging task, partly due to objects of various scales in a complex scene. Whilst filters of varying sizes can be used to create multiple receptive fields to process these objects, they can lead to an exponential growth of parameters and computational cost. To address this issue, PSPNet

T. Mantoro et al. (Eds.): ICONIP 2021, LNCS 13109, pp. 193–205, 2021.
https://doi.org/10.1007/978-3-030-92270-2_17

[22] introduced a Pyramid Pooling Method (PPM) in which multiple parallel pooling branches with different pool sizes and strides are deployed in order to capture multi-scale contextual information from the scene. Although, multiple pooling branches create receptive fields of different sizes, the pooling operation causes a loss of neighboring information of each object in the scene. To address this problem, the authors of [2] presented a new approach, called Atrous Spatial Pyramid Pooling (ASPP). It uses multiple dilated convolution branches with different dilation rates for multiple receptive fields. A higher dilation rate enlarges the field of view without contributing any extra parameters and GFLOPs. However, it uses dilated convolutions which are sensitive to input image resolution. Moreover, getting a trade-off between dilation rates and input size is a challenging task in ASPP. Similar to feature scaling, object positioning is also an important factor for better scene representation. The literature has shown that global features, produced by the encoder, are highly sensitive to the entire objects whereas local features mainly focus on the boundaries and edges of the object [2,22]. Therefore, fusing local feature with rich global feature is essential for accurate object localization. Existing off-line [2,3,22] and real-time [9,11–13,16,21] semantic segmentation models mainly focus on feature extraction and contextual representation. For object localization, deep features at low resolution are simply fused with shallow features at higher resolution, creating a large semantic gap between the feature maps. This gap produces semantic inconsistency due to the background noisy features. The study [10] also shows that the fusing of global and local features directly is less effective. Therefore, our work introduces an optimised multi-stage feature fusion module at the decoder side for better object localization. Our key contributions are as follows:

- We design a lightweight backbone using MobileNetV2 residual blocks, capable of handling high-resolution input images in real-time environments;
- We introduce a multi-scale Feature Scaling Module (FSM), inspired by [2], and obtain the best trade-off between input size and dilation rates;
- We introduce a multi-stage Feature Fusion Module (FFM) to bridge the semantic gap and improve semantic performance; and
- The proposed model produces state-of-the-art results on Cityscapes among all the real-time models having less than 5 million parameters.

2 Related Work

Traditionally, a semantic segmentation design typically follows an image pyramid structure, which is inefficient for real-time applications as it increases training and inference time. Later on, [5,9,20] introduce an encoder-decoder architecture which involves both a pyramid structure to create semantic features and upsampling layers to produce segmentation. However, many of these models, for example DeepLab [1,2], PSPNet [22], HANet [3], are not suitable for real-time applications as they use a large encoder, such as ResNet [18].

To address real-time requirements, several approaches (Bayesian SegNet [5], ENet [11], ICNet [21], BiSeNet [20], DFANet [6]) have been proposed using a simpler variant of ResNet, but their parameters and GFLOP counts of these models

are still high. More recent models such as ContextNet [12], FAST-SCNN [13], FANet [15], ESPNet [16] have achieved further reduction by using MobileNet bottleneck residual block (MBConv) whilst still maintaining good segmentation performance.

Feature Scaling. Using multiple scales is necessary for better contextual representation in semantic segmentation [2,4,22]. There are three notable approaches to feature scaling: 1)PPM [22] achieves better contextualization but lacks fine details and is still expensive; 2) DMNet [4] provides a dynamic scaling whilst being expensive at high resolutions; and 3) ASPP [2] provides a robust scaling and more controllability through varying dilation rates.

Feature Fusion. Traditionally, high-level features are upsampled and then fused with lower-level features in the deconvolution process [7]. Many offline [2,3] and some real-time [6,9,12,13,21] semantic segmentation models skipped the intermediate stages and upsampled semantic features directly by between 2^3 to 2^5 times, which causes a large semantic gap while fusing features and loses object localization. To address this issue, PAN [8] has introduced a new bottom-up path for accurate signal propagation from lower layers to higher layers for instance segmentation, and this bi-directional propagation has been utilised in FANet [15] and DSMRSeg [19].

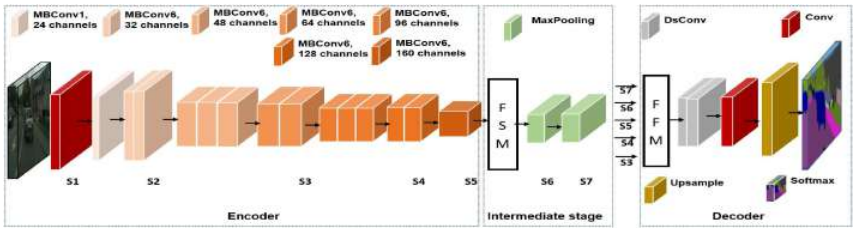

Fig. 1. Complete architecture of the proposed model

3 Proposed Method

Figure 1 displays the complete pipeline of our proposed model. Our work addresses the above challenges by appropriately using multi-scale feature representation and multi-stage feature fusion together with a slim backbone.

3.1 Network Architecture

Encoder. In our design, we exploit MobileNetV2 bottleneck residual blocks (MBConv) to design the backbone of our proposed model as they are much more efficient than other residual blocks. The layer architecture of the encoder network is shown in Table 1. Two types of MBConv blocks are used- MBConv1 with expansion ratio 1 and MBConv6 with expansion ratio 6.

While passing through a typical bottleneck architecture, the channel `c` of the feature gets expanded based on the expansion ratio `t` and becomes `tc`. To reduce the complexity, we employ a Depth-wise Convolution (DwConv) layer at the expansion stage. The layout of the bottleneck residual block is exhibited in Table 2. Variables h, w denote the spatial dimensions of input feature map and s represents stride. ReLU non-linearity is deployed in the first two layers, however it is skipped at the last stage of each MBConv block to preserve meaningful information of the input feature map.

Table 1. Layer architecture of encoder

Stage (i)	Input	Operators	Stride	Layers (n)	Output
1	$1024 \times 2048 \times 3$	Conv, k3 $\times$ 3	2	1	$512 \times 1024 \times 32$
2	$512 \times 1024 \times 32$	MBConv1, k3 $\times$ 3	2	1	$256 \times 512 \times 24$
	$256 \times 512 \times 24$	MBConv6, k3 $\times$ 3	1	2	$256 \times 512 \times 32$
3	$256 \times 512 \times 32$	MBConv6, k3 $\times$ 3	2	3	$128 \times 256 \times 48$
	$128 \times 256 \times 48$	MBConv6, k3 $\times$ 3	1	2	$128 \times 256 \times 64$
4	$128 \times 256 \times 64$	MBConv6, k3 $\times$ 3	2	3	$64 \times 128 \times 96$
	$64 \times 128 \times 96$	MBConv6, k3 $\times$ 3	1	2	$64 \times 128 \times 128$
5	$64 \times 128 \times 128$	MBConv6, k3 $\times$ 3	2	1	$32 \times 64 \times 160$

Table 2. Bottleneck residual block

Input	Operator	Output
$h \times w \times c$	1×1 Conv,1/1, Relu	$h \times w \times tc$
$h \times w \times tc$	3×3 DwConv, 3/s, Relu	$h/s \times w/s \times tc$
$h/s \times w/s \times tc$	1×1 Conv,1/1, -	$h/s \times w/s \times c'$

We employ 14 MBConv blocks to design the backbone. The filter size of each block is controlled by a tunable hyper-parameter, called width multiplier. Following the suggestion in [14], the width multiplier is set between 0.35 and 0.5 to obtain a better trade-off between the model's accuracy and performance. The encoder processes the input through 5 stages, each stage reduces the input feature by half. It is suggested in [2] that an output stride of 2^3 or 2^4 is optimal for an input of 512×512px. As we target a higher resolution (1204×2048), we have found that an output stride of 2^5 provides a better trade-off between model accuracy and performance whilst not overlooking small objects. The complete layout of each stage is displayed in Table 1.

Intermediate Stage. This stage addresses multi-scale representation, which is crucial for complex scene analysis [2,4,22]. Motivated by ASPP [2], we develop a Feature Scaling Module (FSM) targeting real-time applications. Our FSM employs three scaling branches with different dilation rates and one feature

pooling branch. In contrast to ASPP, we use Depth-wise Separable Convolution (DsConv) in each branch. The dilation rate in each branch is sensitive to the input size. At smaller dilation rates, the size of the receptive field is small and it takes more number of operations to filter the input, whereas a higher dilation rate enlarges model's field-of-view whilst potentially causing artifacts. To obtain the best trade-off among the dilation rates and input sizes, we conducted an ablation study which shows that a dilation rate of {8,16,24} gives the best result for input of size 1024×2048. The layout of each branch is shown in Table 3. At the end of FSM, we concatenate all four branches with the input feature. After FSM, we deploy two successive MaxPooling layers to create two additional stages for feature fusion. The MaxPooling layer does not contribute any parameters, hence the model's real-time performance is not hampered.

Table 3. Layer architecture of FSM

Branch	Input	Operator	Filter	Dilation rate	Output
Dilated branch 1	$h \times w \times c$	DsConv, Bn, f	3×3	r_1	$h \times w \times c'$
Dilated branch 2	$h \times w \times c$	DsConv, BN, f	3×3	r_2	$h \times w \times c'$
Dilated branch 3	$h \times w \times c$	DsConv, BN, f	3×3	r_3	$h \times w \times c'$
Feature pooling	$h \times w \times c$	AveragePooling2D Conv, BN, f UpSampling2D	1×1	-	$h \times w \times c'$

Decoder. The proposed decoder has two modules: multi-stage feature fusion (FFM) and classifier. FFM is required for identifying the region and localizing the objects in the scene. In a pyramid encoder design, neurons at top levels strongly respond to entire objects while neurons at lower level more likely capture local texture and patterns. Motivated by this idea, we introduce an effective multi-stage feature fusion module at the decoder side. It takes five rich semantic features from five different levels and fuses it through three different paths. The operation of FFM is illustrated in Fig. 2.

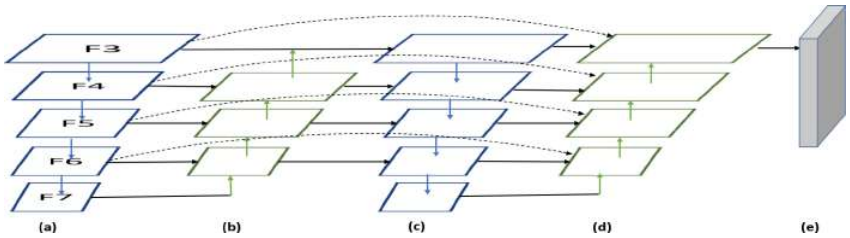

Fig. 2. Multi-stage Feature Fusion Module: (a) Features F_3-F_7 generated by encoder, (b) Top-down path for feature fusion, (c) Bottom-up path for object localization, (d) Top-down path for better contextual assimilation. Dotted lines mean skip connections from the encoder.

Traditionally, a high-level rich semantic feature map F_i ($C_i \times H_i \times W_i$) is upsampled and fused with the former low-level features F_{i-1} ($C_{i-1} \times H_{i-1} \times W_{i-1}$) [5,7] to regenerate the scene. This one direction (top-down) FFM may lead to localization issues in the scene. Moreover, each upsampling method contributes to the loss of neighboring information. This phenomenon clearly manifests the necessity of deploying another bottom-up path for accurate object localization in the entire feature hierarchy and also the need for lateral connections in the encoder to prevent the loss of neighboring details. For that reason, we introduce a new bottom-up path where local feature maps are fused with global features in order to achieve better localization. Finally, we introduce another top-down path for better contextual assimilation using some skip connections from the different stages of the encoder (dotted lines in Fig. 2).

Finally, we deploy a simple, yet effective classifier consisting of two DsConv, one point-wise Conv, one Upsampling and one softmax layer. The activation function softmax is used to assign a class to each pixel of the image. We also use one Dropout layer to avoid overfitting.

4 Experiment

4.1 Datasets

Cityscapes. It is a large-scale data set for semantic understanding of urban street scenes. It provides 5,000 fine-tune and 20,000 coarse annotated images. The fine-tune images are divided into three parts: a training set with 2,975 images, a validation set with 500 images, and a test set with 1,525 images, all at 1024×2048. This data set has 33 classes, 19 of which are used for training.

BDD100K. It is a recent data set developed to meet the growing demand in the field of autonomous car industry. It is the largest driving video dataset with 100K videos. Compared to Cityscapes, it is more challenging due to its diverse nature. It provides 8,000 fine-grained, pixel-level annotations, 7,000 of which are used for training and 1,000 for validation. The class labelling of this benchmark is compatible with Cityscapes (see our Github for further detail). Each image in this dataset has 720×1280 pixels.

4.2 Implementation Details

All our experiments are conducted in a dual Nvidia TITAN RTX GPUs system, each GPU having 24 GB of memory. Our environment includes `CUDA` 10.2, `tensorflow` 2.1.0, `keras` 2.3.1., and `Horovod`. For training, we set a batch size of 2 for full input resolution and 4 for low input resolution. For model optimizer, we employ stochastic gradient decent (SGD) with a momentum of 0.9. Following [13,22], we use the 'poly' learning rate policy which computes the current learning rate (LR_{current}) in each epoch as $LR_{\text{current}} = LR_{\text{base}} \times (1 - \text{iter}/\text{maxiter})^{\text{power}}$, where `iter` defines current iteration and `maxiter` defines maximum number of iterations in each epoch. We set LR_{base} to 0.045 and `power` to 0.9.

To overcome the limited samples of the data sets, we implement several data augmentation techniques, such as random horizontal flip, random crop, resizing of image, adjust the brightness of images. We also employ few regularization techniques such as ℓ_2 regularization and Dropout in the classifier module. We set the ℓ_2 regularization hyper-parameter to 0.00004 and dropout rate to 0.35. We utilize the categorical cross-entropy function for the model loss.

4.3 Ablation Study

At the initial stage, without using any FSM and FFM modules, we evaluated model performance on the Cityscapes data set. In the next stage, we deployed the FSM module on top of the backbone and reported the results. Table 4 clearly displays that the use of FSM module enhances model performance. We exploit different feature-scaling techniques. In the final stage, we introduce one multi-stage feature fusion module on top of FSM to exploit the benefits of FFM. We also report the model's performance by utilizing other existing feature fusion module and compare the results with our designs. Table 4 demonstrates that with the use of our FSM and FFM modules, the proposed model produces better results on the Cityscapes validation set. To get a best trade-off between dilation rate and input size, we also trained our model with different rates. It was shown in [2] that ASPP performed better at dilation rates {6,12,18,24}, whereas this study shows that our model attains best performance at dilation rates {8,16,24} for

Table 4. Segmentation performance evaluation

Backbone	FSM	FFM	mIoU (%)	Number of parameters (Million)	GFLOPs
14MBConv	–	–	60.3	1.11	50.9
14MBConv	PPM	–	63.2	1.32	53.4
14MBConv	ASPP	–	64.1	1.73	53.9
14MBConv	Ours	–	64.6	1.21	51.3
14MBConv	Ours	FPN	66.4	1.25	50.3
14MBConv	Ours	PAN	67.2	1.36	49.4
14MBConv	Ours	Bi-FPN	67.9	1.27	48.7
14MBConv	Ours	Ours	68.3	1.29	50.8

Table 5. Segmentation performance evaluation

Input size	output stride	Global feature size	Dilation rates	mIoU (%)
1024×2048	32	32×64	4,8,12	67.4
1024×2048	32	32×64	6,12,18	68
1024×2048	32	32×64	**8,16,24**	**68.3**
1024×2048	32	32×64	12,24,36	67.9

an input at 1024×2048. The difference is because the proposed model handles higher resolution input than [2]. As the stride is fixed, the size of the global feature map is large for higher input resolution. Therefore, larger dilation rates are required for better receptive fields. However, it is noted that the model performance drops when the dilation rate is beyond 8,16,24. Table 5 displays the performance at different rates. To reduce the training time in the ablation study, we only used the fine-tune set and stopped at epoch 500, which is suitable for its purpose. In the main experiments, we use all relevant sets and extend the number of epochs to 1000.

4.4 Model Evaluation

Model performance on Cityscapes. This section demonstrates model performance on Cityscapes dataset and compares its performance with other existing off-line and real-time semantic segmentation models. Table 6 reports its performance over 19 classes of Cityscapes validation and test sets. All classes of Cityscapes dataset are divided into 7 categories. Table 7 displays model performance on each category of Cityscapes dataset. The class-based result demonstrates that our model attains an accuracy of above 90% for 5 classes. Similarly, its accuracy is more than 90% in 5 categories.

Table 6. Class-wise FSFFNetperformance on Cityscapes validation and test sets

Dataset	Road	S. walk	Build.	Wall	Fence	Pole	T.light	T.sign	Veg.	Terrain
Validation set	**96.4**	77.7	**90.6**	57.0	52.1	58.3	63.5	72.7	**91.0**	62.2
	Sky	Person	Rider	Car	Truck	Bus	Train	M.cycle	Bicycle	**mIoU**
	93.2	75.9	51.2	**93.3**	67.8	79.1	64.0	47.5	70.8	**71.8**
Dataset	Road	S.walk	Build.	Wall	Fence	Pole	T.light	T.sign	Veg.	Terrain
Test set	**97.4**	78.5	**90.7**	41.8	46.1	57.8	65.3	68.5	**92.0**	63.9
	Sky	Person	Rider	Car	Truck	Bus	Train	M.cycle	Bicycle	**mIoU**
	94.4	79.2	56.9	**93.9**	55.4	65.7	54.4	50.4	65.8	**69.4**

Table 7. Category-wise FSFFNetperformance on Cityscapes validation and test set

Dataset	Flat	Construction	Object	Nature	Sky	Human	Vehicle	mIoU
Validation set	**96.0**	**90.3**	65.0	**91.4**	**93.2**	77.9	**91.2**	**86.4**
Test set	**96.8**	**91.5**	64.1	**94.4**	**90.2**	79.8	**92.7**	**87.1**

Table 8. Performance evaluation of different models on Cityscapes validation set

Type	Model	Input Size	Class mIoU(%)	Category mIoU(%)	Parameters (Million)	GFLOPs
Off-line	DeepLabV3+ [2]	1024 × 2048	64.5	82.6	54.8	344.9
	PSPNet [22]	713 × 713	81.2	91.2	250.8	516
	HANet [3]	768 × 768	80.9	-	65.4	2138.02
Real-time	Bayesian SegNet* [5]	1024 × 2048	63	82.1	30	2729.2
	BiseNet [20]	360 × 640	69	-	5.8	2.9
	ContextNet* [12]	1024 × 2048	60.4	81.5	1.0	37.5
	DFANet-B [6]	360 × 640	68.4	-	4.9	2.1
	FAST-SCNN* [13]	1024 × 2048	63.3	82.2	1.2	14.9
	FANet* [15]	1024 × 2024	65.9	83.6	1.1	11.4
	ICNet [21]	1024 × 2048	67.7	-	6.68	58.5
Real-time	**FSFFNet***	**1024 × 2048**	**71.8**	**86.4**	**1.3**	**50.8**

We also trained some existing semantic segmentation models under the same system configuration and presented results in Table 8. Models trained by us are marked with '*' sign in Table 8. Results of other models in Table 8 are obtained from either the literature or Cityscapes leaderboard. Note that the authors of DeeplabV3+ [2] deployed a new Xception (X-65) model on top of DeepLab previous version [1] to make a deeper encoder for semantic segmentation. However, we only used X-65 as feature extractor. We also replaced standard Conv layers of DeepLab and Bayesian SegNet [5] by DsConv layers to make the models computationally efficient in our system. We also incorporated ASPP on top of X-65 as a dense feature extractor.

Without utilizing image-level features, we attained 64.5% validation mean Intersection over Union (mIoU) on Cityscapes validation set. Among other trained models, we achieved 63.3% and 60.5% mIoU for FAST-SCNN [13] and ContextNet [12] respectively. They are different from the original claim of 68.6% and 65.9% mIoU on the validation set. We conjecture that the existing models might have been pre-trained with other datasets or some post-processing techniques might have used to boost their performance. In this study, we trained the model with the fine-tune set first, then followed by the coarse set. After that, we again trained the model with the fine-tune set. Table 8 clearly demonstrates that among all the real-time scene segmentation models, our proposed model produces the best validation accuracy (71.8%) on Cityscapes while having only 1.3 million parameters. Similar observation can be drawn from Table 9. Our proposed model achieves 69.3% test accuracy, setting a new state-of-the-art result among the existing real-time semantic segmentation models having less than 5 million parameters. While comparing model parameter and GFLOPs, we noticed that existing models reported their GFLOPs count at lower input resolution. With the increase of input resolution, GFLOPs increases exponentially. Therefore, comparing GFLOPs at different input resolutions is not an appropriate approach. We can compare model parameters as it does not depend on

Table 9. Performance evaluation of different models on Cityscapes test set

Model	Input size	Class mIoU (%)	Category mIoU (%)	parameters (Million)	GFLOPs
ENet [11]	360 × 640	58.3	80.4	0.4	3.8
ICNet [21]	1024 × 2048	69.5	-	6.68	-
FCN 8S [9]	512 × 1024	65.3	85.7	57	-
BiseNet [20]	360 × 640	68.4	-	5.8	2.9
ContextNet [12]	1024 × 2048	66.1	82.8	1.0	37.5
DFANet-B [6]	360 × 640	67.1	-	4.9	2.1
FAST-SCNN [13]	1024 × 2048	68.0	84.7	1.2	14.9
FANet [15]	1024 × 2048	64.1	83.1	1.1	11.4
FSFFNet	**1024 × 2048**	**69.4**	**87.1**	**1.3**	**50.8**

Table 10. Performance evaluation on validation set of BDD100K dataset

Type	Model	Input size	Parameters (Million) (%)	GFLOPs	Class mIoU (%)
Off-line	HANet (MobileNetV2) [3]	608 × 608	14.8	142.7	58.9
	HANet (ResNet-101)	608 × 608	64.2	2137.8	64.8
Real-time	ContextNet*	768 × 1280	1.0	37.5	44.5
	FAST-SCNN*	768 × 1280	1.2	14.9	47.9
	FANet*	768 × 1280	1.1	11.4	50.0
	FSFFNet*	**768 × 1280**	**1.3**	**50.8**	**55.2**

input size. Compared to previous state-of-the-art performance by ICNet [21], FSFFNetis 5 times smaller, however it produces similar test accuracy and more than 3% higher validation accuracy.

Model Performance on BDD100K. We also trained our model with the BDD100K dataset and presented the results in Table 10. From the literature, we found only one off-line model (HANet [3]) evaluated on BDD100K. The remaining models presented in Table 10 were trained by us under the same settings. As HANet [3] is an off-line model, it was expected to have better accuracy than our proposed real-time model.

To better see the effect of large models, we implemented a shallow variant of HANet which uses MobileNetV2 [14] as the backbone instead, and it produced 58.9% mIoU on BDD100K validation set. Our method (55.2%) is only few percent behind this variant whilst having 11 times smaller number of parameters. Compared to other models reported in Table 10, our proposed model FSFFNet-consistently achieves a better segmentation accuracy.

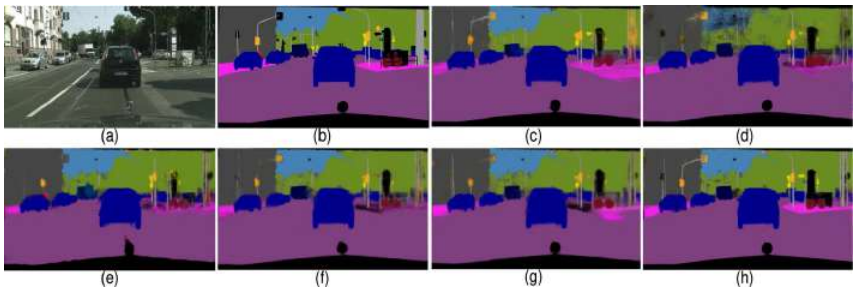

Fig. 3. Cityscapes val. set: (a) RGB input, (b) Color annotation, (c) DeepLabV3, (d) Bayesian SegNet, (e) ContextNet, (f) FAST-SCNN, (g) FANet, (h) FSFFNet

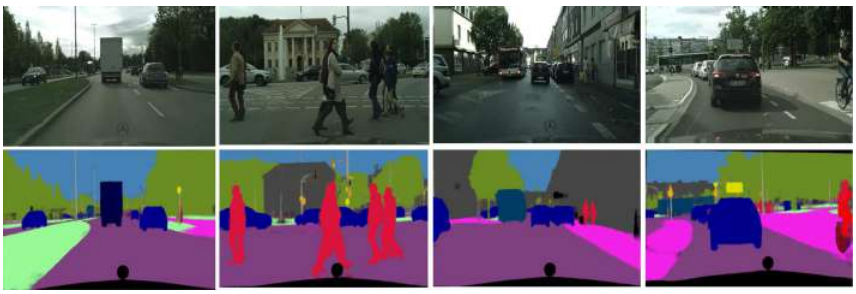

Fig. 4. Output by FSFFNeton Cityscapes test set

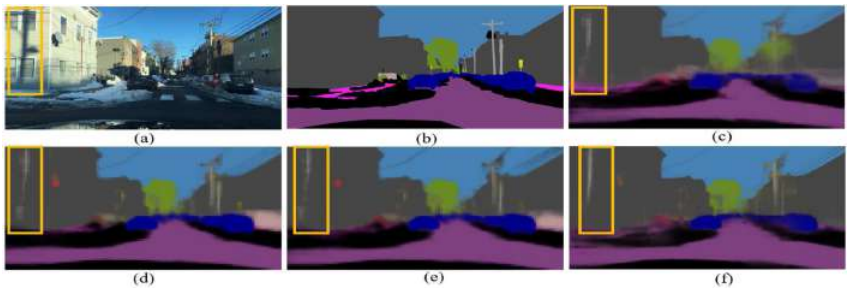

Fig. 5. Output by different models on BDD100K validation set: (a) RGB, (b) Colored Annotation, (c) ContextNet, (d) FAST-SCNN, (e) FANet, (f) FSFFNet

Qualitative Performance Analysis. We present samples of the output generated by all trained models in Figs. 3, 4, 5, 6. It can be clearly seen that the quality of the output generated by FSFFNetis better than the other models: boundary degeneration, overlapping classes, noisy feature effects can be observed in the output generated by other models whereas in FSFFNet, sharp boundaries of

Fig. 6. Output by FSFFNeton BDD100K test set

each object and the presence of tiny objects clearly demonstrate performance superiority in semantic segmentation. Due to the complex nature of BDD100K, noisy feature effects, wrong classification can be observed in the output generated by all models. However, inline with the quantitative results, the segmented images by FSFFNetare much better than others.

5 Conclusion

This study presents a computationally efficient real-time semantic segmentation model based on a light-weighted backbone, capable of handling high-resolution input images. The performance of the model is evaluated by two publicly available benchmarks and the results clearly demonstrate that our proposed model sets a new state-of-the-art results on Cityscapes benchmark in real-time semantic segmentation. Our feature fusion module helps reduce the semantic gap between the features, whereas our feature scaling module assimilates more contextual information for better scene representation. In the future, we plan to extend the model for indoor scene analysis. The implementation of our proposed model is publicly available at https://github.com/tanmaysingha/FSFFNet.

References

1. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: DeepLab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. IEEE TPAMI **40**(4), 834–848 (2017)
2. Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation. In: Proceedings ICCV, September 2018
3. Choi, S., Kim, J.T., Choo, J.: Cars can't fly up in the sky: improving urban-scene segmentation via height-driven attention networks. In: Proceedings CVPR, pp. 9373–9383 (2020)
4. He, J., Deng, Z., Qiao, Y.: Dynamic multi-scale filters for semantic segmentation. In: Proceedings ICCV, pp. 3562–3572 (2019)

5. Kendall, A., Badrinarayanan, V., Cipolla, R.: Bayesian SegNet: model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. arXiv preprint arXiv:1511.02680 (2015)
6. Li, H., Xiong, P., Fan, H., Sun, J.: DfaNet: deep feature aggregation for real-time semantic segmentation. In: Proceedings CVPR, pp. 9522–9531 (2019)
7. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: Proceedings CVPR, pp. 2117–2125 (2017)
8. Liu, S., Qi, L., Qin, H., Shi, J., Jia, J.: Path aggregation network for instance segmentation. In: Proceedings CVPR, pp. 8759–8768 (2018)
9. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings CVPR, pp. 3431–3440 (2015)
10. Pang, Y., Li, Y., Shen, J., Shao, L.: Towards bridging semantic gap to improve semantic segmentation. In: Proceedings ICVV, pp. 4230–4239 (2019)
11. Paszke, A., Chaurasia, A., Kim, S., Culurciello, E.: ENet: a deep neural network architecture for real-time semantic segmentation. arXiv preprint arXiv:1606.02147 (2016)
12. Poudel, R.P., Bonde, U., Liwicki, S., Zach, C.: ContextNet: exploring context and detail for semantic segmentation in real-time. arXiv preprint arXiv:1805.04554 (2018)
13. Poudel, R.P., Liwicki, S., Cipolla, R.: Fast-SCNN: fast semantic segmentation network. arXiv preprint arXiv:1902.04502 (2019)
14. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: MobileNetv 2: inverted residuals and linear bottlenecks. In: Proceedings CVPR, pp. 4510–4520 (2018)
15. Singha, T., Pham, D.S., Krishna, A.: FaNet: feature aggregation network for semantic segmentation. In: Proceedings DICTA, pp. 1–8. IEEE (2020)
16. Singha, T., Pham, D.-S., Krishna, A., Dunstan, J.: Efficient segmentation pyramid network. In: Yang, H., Pasupa, K., Leung, A.C.-S., Kwok, J.T., Chan, J.H., King, I. (eds.) ICONIP 2020. CCIS, vol. 1332, pp. 386–393. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-63820-7_44
17. Tan, M., Le, Q.V.: EfficientNet: rethinking model scaling for convolutional neural networks. arXiv preprint arXiv:1905.11946 (2019)
18. Targ, S., Almeida, D., Lyman, K.: ResNet in ResNet: generalizing residual architectures. arXiv preprint arXiv:1603.08029 (2016)
19. Yang, M., Shi, Y.: DSMRSeg: dual-stage feature pyramid and multi-range context aggregation for real-time semantic segmentation. In: Gedeon, T., Wong, K.W., Lee, M. (eds.) ICONIP 2019. CCIS, vol. 1142, pp. 265–273. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-36808-1_29
20. Yu, C., Wang, J., Peng, C., Gao, C., Yu, G., Sang, N.: BiSeNet: bilateral segmentation network for real-time semantic segmentation. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11217, pp. 334–349. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01261-8_20
21. Zhao, H., Qi, X., Shen, X., Shi, J., Jia, J.: ICNet for real-time semantic segmentation on high-resolution images. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11207, pp. 418–434. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01219-9_25
22. Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. In: Proceedings CVPR, pp. 2881–2890 (2017)

.4 Publication 4

SCMNet: Shared Context Mining Network for Real-time Semantic Segmentation

Tanmay Singha, Moritz Bergemann, Duc-Son Pham, and Aneesh Krishna
School of Electrical Engineering, Computing and Mathematical Sciences
Curtin University, Perth, Western Australia
tanmay.singha@postgrad.curtin.edu.au, moritz.bergemann@student.curtin.edu.au,
dspham@ieee.org, a.krisna@curtin.edu.au

2021 Digital Image Computing: Techniques and Applications (DICTA) | 978-1-6654-1709-9/21/$31.00 ©2021 IEEE | DOI: 10.1109/DICTA52665.2021.9647401

***Abstract*—Different architectures have been adopted for real-time scene segmentation. A popular design is the multi-branch approach in which multiple independent branches are deployed at the encoder side to filter input images at different resolutions. The main purpose is to reduce the computational cost and handle high resolution. However, independent branches do not contribute in the learning process. To address this issue, we introduce a novel approach in which two branches at the encoder share their knowledge whilst generating the global feature map. At each sharing point, the shared features will go through a new effective feature scaling module, called the Context Mining Module (CMM), which will refine the shared knowledge before passing it to the next stage. Finally, we introduce a new multi-directional feature fusion module which fuses deep contextual features with shallow features successively with accurate object localization. Our novel scene parsing model, termed SCMNet, produces 66.5% validation mIoU on the Cityscapes dataset and 78.6% on the Camvid dataset whilst having only 1.2 million parameters. Furthermore, the proposed model can efficiently handle higher resolution input images whilst having low computational cost. Our proposed model produces state-of-the-art results on Camvid.**

***Index Terms*—semantic segmentation, multi-branch, feature fusion, real-time models, DCNNs**

I. Introduction

Semantic Segmentation is a key task in computer vision. It involves assigning pixel-wise labels to identify objects and their boundaries in an image. It is a preliminary step in many modern machine learning applications, such as computational photography, autonomous vehicles, robotics and video surveillance. Notably, many of these practical applications requires real-time scene segmentation which is a very challenging.

For the past decade, Deep Convolutional Neural Networks (DCNNs) have consistently achieved state-of-the-art performance in computer vision tasks [1]–[5]. Different neural architectures such as image cascading, single branch pyramid and multi-branch pyramid layout have been deployed for better scene parsing. Multi-branch structures which extract features from input of different resolutions have been especially successful [2], [6]. Whilst these models achieve top accuracy, they do not meet the real-time performance requirement - annotation times are often upwards of one second even on costly accelerated hardware [2], [7], [8]. Therefore, various models have been developed in recent years focusing on real-time segmentation [9]–[11]. These models attempt to minimise memory usage by reducing the number of layers in the deep branch. The efficiency of handling high-resolution input images is also improved by introducing multiple shallow branches at the encoder.

Several models have effectively utilized multiple branches at the backbone [12]–[15]. Instead of accepting inputs of various sizes from multiple branches, they take a single input from one branch and create an intermediate shallow branch parallel to deep branch via down-sampling. Whilst this approach is more computationally efficient, it often leads to a drop in accuracy.

Recently, it has been shown that feature scaling at the intermediate stage of encoder enhances model performance [1], [7]. PSPNet [8] introduced Pyramid Pooling Module (PPM) and DeepLab [7] proposed Atrous Spatial Pyramid Pooling (ASPP), both scale deep features at different rates. Additionally, ASPP provides a large receptive field which enhances the performance. Building upon these techniques, few real-time scene parsing models [16], [17] have been presented with promising results.

To achieve real-time performance, all existing models directly upsample the global feature map by 2^3 to 2^4 times without having proper feature fusion at the intermediate stage [9]–[11], [14]. However, it was shown that intermediate feature fusion is essential when upsampling global features [18]. It also reduces boundary degeneration effects.

To address all the issues above, we propose a novel shared-branch approach for real-time scene segmentation. Our key contributions are as follows:

- A novel shared-branch backbone in which both deep and shallow branches share their knowledge at three intermediate stages;
- A new yet effective Context Mining Module (CMM) which extracts more contextual information deep feature maps and better feature refinement;
- A new multi-directional Deep Shallow Feature Fusion Module (DS-FFM) for better context assimilation and accurate object localization;

We evaluate our proposed model over three publicly available benchmarks. It produces competitive results (66.5% on Cityscapes, 78.6% on Camvid and 51.2% on BDD100K) whilst having only 1.2 million parameters.

978-1-6654-1709-9/21/$31.00 ©2021 IEEE

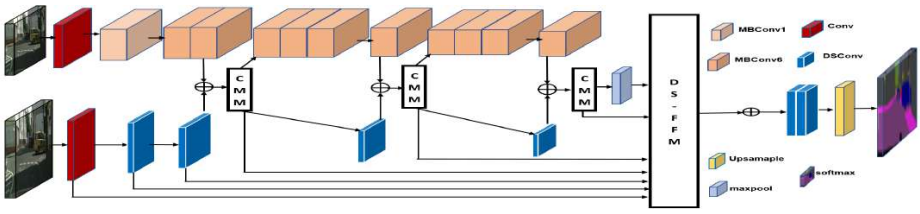

Fig. 1. Complete architecture of SCMNet

II. Related Work

A. Multi-branch architectures

The multi-branch encoder architecture was first proposed for semantic segmentation by RefineNet [2] and BiSeNet [6]. These structures split the encoder into multiple paths, each with different resolutions. Lower-resolution paths are made deeper to extract rich contextual information, while higher-resolution, shallower paths extract fine boundary and spatial details. The branches are then merged at the decoder end to create a final feature map with both deep global feature map at low resolution and shallow local feature map at high resolution. Motivated by this approach, later on, few models such as ICNet [10] and ContextNet [9] were introduced. By keeping less number of layers in all branches, these models improved model efficiency. However, independent parallel branches at encoder do not contribute to knowledge sharing.

To address this, FAST-SCNN [14], FANet [15], DDRNet [12] introduced a dual-branch approach in which an intermediate shallow branch is created after few initial layers of the main branch. However, this approach introduces a large semantic gap between shallow and deep features due to lack of proper feature fusion technique at decoder end.

B. Feature scaling and feature fusion modules

Several additional modules have been proposed in recent years. For instance, PPM [8], and ASPP [7] are feature scaling modules which provide different field of views for a pixel. Thus, they enhance the probability of assigning the right class to each pixel. Inspired by this approach, several real-time semantic segmentation models [9], [14], [16], [17] adopted these scaling techniques and have shown the improvements in model performance. PPM comprises of four image-pooling branches whereas ASPP employs four dilated and one point-wise convolution branches. Due to higher dilation rates, ASPP provides larger receptive field compare to PPM and also consumes less memory.

Like feature scaling, feature fusion module also plays an important for context engrossment and region localization. Literature [15], [18] have shown that by introducing top-down and bottom-up augmented paths in decoder architecture, accurate object localization can be achieved. Top-down path ensures contextual engrossment by fusing deep features with shallow features whereas bottom-up path enhances object localization by fusing shallow features with deep features.

III. Proposed Method

In this section, we demonstrate the complete pipeline of our proposed model. In contrast to existing independent multi-branch encoder designs, we introduce a shared multi-branch encoder architecture for better refinement of local and global information. The overall architecture of the proposed SCM-Netis shown in Figure 1.

A. Shared multi-branch encoder

Our proposed model has two branches at the encoder side. The low input resolution deep branch is employed for extracting rich contextual information, whereas the high resolution shallow branch is used to assimilate local features such as textures, patterns, and object boundaries. Traditionally, these two branches work independently at the encoder side. Then, at the decoder end, rich contextual feature maps from the top of the encoder are fused with high resolution local feature maps from the shallow branches to produce the segmented output. However, this independent approach at the encoder side does not help the model gain spatial details of the scene. Moreover, a lack of synchronization between the branches at the encoder side increases the semantic gap between local and global feature maps. To reduce the semantic gap and improve the model's spatial knowledge, information sharing between the branches at the encoder side is required. At three stages, our deep and shallow branch share information and propagate it to the next stage (see Figure 1).

The complete layout of the shallow and deep branches are displayed in Table I and II respectively. Our shallow branch accepts a higher resolution (1024×2048) input images, whereas the deep branch handles a lower size (512×1024) input. Table I shows that the shallow branch consists of one standard convolution and four successive depth-wise separable convolution layers. While passing through each layer, the input image is filtered by a 3×3 kernel and reduced to half of the previous layer's size. At the end of the shallow branch, the model produces a feature map which is 2^5 times smaller.

We design our deep branch using 11 MBConv blocks of MobileNetV2 [19]. Due to its deep nature, we make the branch's initial input size half the original input's resolution. It

has four stages and each stage reduces the image dimensions by half. Traditionally, semantic segmentation models employ few initial convolution layers for down-sampling the input image before it passes through the deep residual blocks. However, in our proposed deep branch, input size is already reduced by half. Therefore, instead of multiple convolution layers, we employ only one with a stride of 2. After passing through the proposed deep branch, the contextual feature map will have a size of 32×64×160, similar to the final feature map generated by the shallow branch.

Figure 1 displays that both branches share the details at 3 stages. We utilize simple addition of tensors for efficiency. Shallow feature $F_{S_i}^{h*w*c}$ will be mapped with deep feature $F_{D_i}^{h*w*c}$ at the i^{th} stage

$$S_i^{h\times w\times c} = F_{S_i}^{h\times w\times c} \bigoplus F_{D_i}^{h\times w\times c} \tag{1}$$

where $\bigoplus$ denotes the weighted-addition operation. To preserve non-linearity, the ReLU activation function is deployed after each addition. Thus, both branches enhance the knowledge by sharing it and propagating it to themselves. In our ablation study, we will show that the performance is improved with this shared design compared to a non-sharing approach. Our proposed encoder produces 3 shared feature maps which will be exploited in the feature fusion stage.

We place one max pooling layer on top of the encoder to create an additional stage for our feature fusion module and this will be discussed later.

TABLE I
LAYER ARCHITECTURE OF SHALLOW BRANCH OF ENCODER

Stage	Input	Operators	Layers (n)	stride	Output
1	1024×2048×3	Conv, k3×3	1	2	512×1024×24
2	512×1024×24	DSConv, k3×3	1	2	256×512×32
3	256×512×32	DSConv, k3×3	1	2	128×256×48
4	128×256×48	DSConv, k3×3	1	2	64×128×96
5	64×128×96	DSConv, k3×3	1	2	32×64×160

TABLE II
LAYER ARCHITECTURE OF DEEP BRANCH OF ENCODER

Stage	Input	Operators	Layers (n)	stride	Output
1	512×1024×3	Conv, k3×3	1	2	256×512×32
2	256×512×32	MBConv1, k3×3	1	2	128×256×32
	128×256×32	MBConv6, k3×3	2	1	128×256×48
2	128×256×48	MBConv6, k3×3	3	2	64×128×64
	64×128×64	MBConv6, k3×3	1	1	64×128×96
3	64×128×96	MBConv6, k3×3	3	2	32×64×128
	32×64×128	MBConv6, k3×3	1	1	32×64×160

1) MBConv block: The literature has shown that for mobile devices, MobileNet's [19] bottleneck residual block (MBConv) is more efficient than other residual blocks such as ResNet [5] and Xception [20]. Motivated by this, we design our deep branch by utilizing 11 MBConv blocks of MobileNetV2. Two types of MBConv blocks are used. A block with an expansion ratio 1 is denoted by MBConv1, whereas MBConv6 expands the width of the input by 6 times when input passes through the bottleneck section of the block. The basic implementation structure of MBConv is demonstrated in Table III. Here, h, w, c and c' denote the spatial dimensions of tensors, t defines the expansion ratio and s defines the stride. ReLU activation functions are used after every layer except the final one to prevent non-linearities from destroying meaningful information.

The layered architecture to each MBConv block is the same. The filter size of each block is controlled by a tunable hyper-parameter, called width multiplier. Inspired by MobileNetV2 [19], we set {32, 48, 64, 96, 160} as the channel size of respective MBConv blocks in the deep and shallow branches.

TABLE III
BOTTLENECK RESIDUAL BLOCK

Input	Operator	Output
$h\times w\times c$	1×1 Conv,1/1, ReLU	$h\times w\times tc$
$h\times w\times tc$	3×3 DwConv, 3/s, ReLU	$h/s\times w/s\times tc$
$h/s\times w/s\times tc$	1×1 Conv,1/1, -	$h/s\times w/s\times c'$

2) Context mining module: In a pyramid encoder design, global features at lower resolutions strongly respond to entire objects while shallow features at higher resolutions more likely assimilate local texture and patterns. Traditionally, a feature scaling Module (FSM) is utilized on top of the encoder to extract rich contextual information from the last stage global feature. After filtering features at different scales, it passes through feature fusion module (FFM) to combine global context with the shallow details. In contrast, we introduce a new module, the Context Mining Module (CMM), which mines contextual details from the shared feature maps of the encoder network. After every fusion of shallow and deep branch features, we deploy CMM to filter features at different scale and mine more contextual details.

Figure 2 illustrates CMM. It has four parallel branches - one point-wise convolution branch, one dilated depth-wise feature scaling branch, and two feature pooling branches. Among feature pooling branches, one is the coarsest label which pools the feature map by its size and generates the coarsest feature of 1×1×c for a given feature map (x) of $h \times w \times c$ size where h, w define spatial dimensions of input tensor and c denotes the channel dimension respectively. We employ a point-wise convolution after pooling to maintain the weight of the global features and reduce the input channel from c to $c\prime$. We then directly up-sample the coarsest features to get the same spatial output dimensions as the original input feature. Other pooling branch has similar layer architecture, but with a pool size of (2, 2) which generates a fine feature map. Thus, the global features are scaled at coarsest and finest labels via image pooling. Similarly, a point-wise convolution branch is employed to filter each pixel by a 1×1 kernel and produce finest label feature of the input. On the other hand, dilated branch scales input feature along the depth of the feature with higher dilation rate (r). It enlarges the field of view of filters to incorporate a larger context. In all four branches, the depth of the feature map is reduced to one-fourth of input feature map so that after concatenating all four outputs, the final feature map will have same dimensions of the input feature map ($h \times w \times c$). Thus,

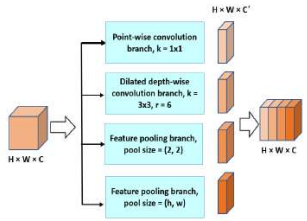

Fig. 2. Proposed context mining module

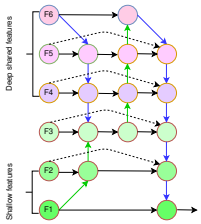

Fig. 3. Proposed deep shallow feature fusion module

mining features at last three stages of encoder design presents a better context of the scene.

3) Feature fusion module: The final contextual feature map generated by the encoder network is 2^6 times smaller than the original input. Therefore, the final feature map must be scaled up to the original input size. In existing real-time semantic segmentation models, researchers usually upsample the feature map by 8 times using a bilinear upsampling layer to reduce inference time. However, this approach compromises the model's performance. To overcome this, we introduce an effective Deep Shallow Feature Fusion Module (DS-FFM) which provides better localization of each object and also reduces the large semantic gap among the feature maps.

The layout of DS-FFM is displayed in Figure 3. There are 3 deep shared features, 2 shallow features and 1 intermediate feature in this module. Deep shared features contain contextual details of entire objects whereas shallow features apprehend boundary and texture details of each object in the scene. The intermediate feature ($F3$) is used to fuse deep and shallow features together at the intermediate stage. The proposed DS-FFM fuses features in two directions. In first path, feature $F6$ is upsampled top-down and feature $F1$ is down-sampled bottom-up to fuse with intermediate feature $F3$. To improve the entire feature hierarchy with accurate propagation of local features, bottom-up path augmentation is introduced. It provides better object localization in the entire scene. Therefore, after fusing features at stage 3, the feature is still down-sampled until stage 6 through the bottom-up path. Finally, we introduce the final top-down path with few lateral connections to aggregate features and produce the final feature output.

For fusing features, we use standard weighted-addition operations. For instance, feature $F_4^{h\times w\times c}$ will be upsampled and feature $F_2^{h\times w\times c}$ will be down-sampled before fusing with feature $F_3^{h\times w\times c}$.

$$F_3^{out} = F_3^{h\times w\times c} \bigoplus Upsampling2D(F_4^{h\times w\times c}) \bigoplus maxPooling2D(F_2^{h\times w\times c}). \quad (2)$$

After every addition operation, a feature passes through a separable convolution block for refinement. Each separable block contains one depth-wise separable convolution, batch-normalization and ReLU layer.

4) Classifier module: This module is employed to assign a class to each pixel of an input image. It contains two depth-wise separable convolutions, one standard convolution, and one upsampling layer. We also use a dropout layer before the final prediction to reduce overfitting. The softmax activation function is used to predict each pixel's class, and thus the final output.

IV. Experiment

A. Datasets

Cityscapes Cityscapes [21] is the most popular benchmark which provides around 5,000 fine annotated images, with an additional 20,000 coarse annotations. Objects are classified into 33 classes and 8 categories. However, the proposed study only considers 19 classes for pixel annotations. The whole fine-tune dataset is divided into three parts - training set (2,975 images), validation set (500 images) and test set (1,525 images). Model performance is evaluated on validation set using fine-tune training set.

BDD100K This is the largest driving video dataset with 100K videos. It provides fine-grained, pixel-level annotations for 7K images which can be used for semantic or instance segmentation. Class labelling of this benchmark is compatible with Cityscapes labelling. Therefore, the same object class annotations (19 classes) are used to evaluate the proposed model's performance on this dataset.

CamVid is a smaller dataset [22]. The images were generated from a recorded video, then annotated frame-by-frame by human operators. This dataset contains only 267 images for training, 101 for validation and 233 for testing. Out of 32 classes, current studies uses 11 classes (excluding void class) for performance evaluation. To measure test accuracy, we combine training and validation set together.

B. Implementation details

To conduct this experiment, we used a dual Nvidia TITAN RTX GPU system where each GPU has 11GB of memory. Our training environment includes `Python` 3.7, `tensorflow` 2.1.0, `keras` 2.3.1, `CUDA` 10.2, and `horovod`. `CUDA` is used for exploiting the parallel processing power of GPUs and for data-parallelism in a distributed environment, we

employ the `horovod` framework [23]. It takes a single-GPU `tensorflow` program and trains model on multiple GPUs. Thus, it boosts the run-time performance by effectively utilizing both the GPUs. For model optimization, we used stochastic gradient decent (SGD) with a momentum of 0.9.

TABLE IV
RESULT OF ABLATION STUDY

Two-branch encoder	CMM	DS-FFM	Val. mIoU (%)
Independent branches	-	-	54.0
Shared branches	-	-	58.7
Shared branches	✓	-	63.5
Shared branches	✓	✓	66.5

We utilized the 'poly' learning rate function with the base value of 0.045 and the power of 0.9. The literature has shown the effectiveness of using a polynomial learning rate for finding the optimal learning rate at each epoch of training [1], [8], [24]. To calculate the model loss, we used the categorical cross-entropy. We employed on-the-fly data augmentation techniques such as random horizontal flipping, vertical flipping, resizing, cropping, adjusting brightness, saturation, and contrast of the input to overcome the limited data. We also exploited ℓ_2 and dropout regularizers to overcome model overfitting. We applied ℓ_2 regularizer with the hyper-parameter value being 0.00004 at top layers of the model except depth-wise convolution layers. A dropout layer is positioned before the final softmax layer. We achieved the best model performance with dropout rate between 0.3 and 0.4.

C. Ablation study

To best design our model, we conducted a series of ablation studies. Initially, we deployed two independent branches at the encoder side and measured model performance without CMM and DS-FFM. In the next stage, we shared the knowledge of both branches in the backbone network. It was observed that sharing the details across the branches boosted model performance by 4.7%. This motivates us to design a shared multi-branch encoder for semantic segmentation. In later stages, we deployed CMM and DS-FFM one by one and have shown model performance in Table IV.

D. Model evaluation

To evaluate the proposed model's performance, we measured validation set mean Intersection over Union (mIoU) of the model on each dataset and present the results here. We also present model parameters, GFLOPs, pixel accuracy, and each class' and category's IoU. The proposed model's performance is compared with some existing offline and real-time semantic segmentation models. Some existing semantic segmentation models were implemented based on the information provided in the literature and they were trained under the same system configuration. For other existing models, their results are obtained directly from the literature and/or official dataset leader-boards. We note that most of the existing models were pre-trained on the ImageNet dataset whereas our proposed model was not. The number of epochs is 1000.

Table V displays class-wise model performance on the Cityscapes validation set at two different input resolutions. Out of the 19 classes, the model performs exceptionally well in 5 categories (Road, Sidewalk, Vegetation, Sky, Car) where it produces almost 90% and above class accuracy at full and half input resolutions. At full input resolution, our proposed model achieves 66.5% mIoU, much better than many existing semantic segmentation models. Table V also demonstrates that its performance is enhanced with increased input resolution. The 19 classes of Cityscapes are divided into 7 categories and the corresponding category-based accuracy is exhibited in Table VI. Out of all categories, its performance is below 80% in 2 categories (object and human). The classes in the object and human categories make up less than 1% of the whole dataset, causing poor model performance on those categories and classes. Overall, its performance would be improved if the model could be trained more with less frequent classes such as motorcycle, rider, traffic light, train, bus, and truck.

E. Performance comparison on Cityscapes

To compare our proposed model's performance with existing models, we present the results of various offline and real-time scene parsing models in Table VII. Models marked with the * sign were trained by us under the same system configuration, whereas the remaining models' performance was extracted from the literature. In DeepLab [7], authors deployed a new Xception (X-65) model on top of DeepLab's previous version [1] to produce a large backbone for feature extraction. However, due to hardware constraints, we could employ only X-65 as an encoder. We also replaced the standard Conv layer of Deeplab and Bayesian SegNet models with DsConv layers to make the models computationally efficient in our hardware system. We present the results obtained from our experiment. Among all offline models, HANet [25] produces the state-of-the-art results and ICNet produces a very competitive performance (67.7%). However, we note importantly ICNet is almost 5 times larger than SCMNetand it also requires more GFLOPs. In terms of model size, ContextNet [9], FAST-SCNN [14], FANet [15] and the proposed SCMNethave a similar number of parameters (1-1.2 Million). Among these, the proposed model produces the best results in our experiments.

F. Performance comparison on BDD100K

Compared to Cityscapes, BDD100K [29] is more due to the diverse nature of input images. We were unable to find any real-time scene segmentation models trained on this benchmark. We could identified one *offline* model (HANet [25]) trained on BDD100K. We trained existing real-time scene parsing models along with the proposed model on BDD100K and present the results in Table VIII. It clearly illustrates that the proposed SCMNetproduces better accuracy (51.2%) than other real-time models. Compared to HANet, SCMNetperforms less competitively, mainly because HANet

TABLE V
CLASS-WISE SCMNETPERFORMANCE ON CITYSCAPES VALIDATION SETS AT DIFFERENT INPUT RESOLUTIONS

Input size	Road	S.walk	Build.	Wall	Fence	Pole	T.light	T.sign	Veg.	Terrain
1024×2048	**96.7**	75.9	**90.2**	53.1	50.3	47.5	54.2	65.1	**90.4**	54.5
	Sky	Person	Rider	Car	Truck	Bus	Train	M.cycle	Bicycle	**mIoU**
	91.9	70.7	46.6	**90.7**	48.7	70.5	60.8	41.2	64.7	**66.5**
Input Size	Road	S.walk	Build.	Wall	Fence	Pole	T.light	T.sign	Veg.	Terrain
512×1024	**95.8**	71.9	87.5	36.0	38.4	46.2	45.8	59.7	**90.0**	53.7
	Sky	Person	Rider	Car	Truck	Bus	Train	M.cycle	Bicycle	**mIoU**
	91.5	67.2	42.5	**90.1**	44.2	56.9	42.7	34.5	60.3	**60.8**

TABLE VI
CATEGORY-WISE SCMNETPERFORMANCE ON CITYSCAPES VALIDATION SET

Input size	Flat	Construction	Object	Nature	Sky	Human	Vehicle	mIoU
1024×2048	**96.5**	**90.0**	56.7	**90.6**	**91.9**	73.2	**90.4**	**84.2**
512×1024	**95.2**	88.3	53.9	**90.4**	**91.4**	70.1	**87.3**	**82.4**

TABLE VII
PERFORMANCE EVALUATION OF DIFFERENT MODELS ON CITYSCAPES VALIDATION SET

Type	Model	Input Size	Class mIoU(%)	Category mIoU(%)	Parameters (Million)	GFLOPs
Off-line	DeepLabV3+ (X-65)* [7]	1024 × 2048	64.5	82.6	54.8	344.9
	DeepLab+LargeFOV [1]	1024 × 2048	63.1	81.2	20.5	-
	HANet [25]	768 × 768	80.3	-	65.4	2138.02
Real-time	Bayesian SegNet* [26]	1024 × 2048	63	82.1	30	2729.2
	ContextNet* [9]	1024 × 2048	60.4	81.5	1.0	37.5
	ESPNet [16]	512 × 512	60.8	-	7.59	13.2
	FAST-SCNN* [14]	1024 × 2048	63.3	82.2	1.2	14.9
	FANet* [15]	1024 × 2024	65.9	83.6	1.1	11.4
	ICNet [10]	1024 × 2048	67.7	-	6.68	58.5
	TwoColumn [27]	512 × 1024	65.3	-	-	57.2
Real-time	**SCMNet***	**1024×2048**	**66.5**	**84.2**	**1.2**	**38.3**

TABLE VIII
PERFORMANCE EVALUATION ON VALIDATION SET OF BDD100K DATASET

Type	Model	Input size	Parameters (Million) (%)	GFLOPs	Class mIoU (%)
Off-line	HANet (MobileNetV2) [25]	608×608	14.8	142.7	58.9
Real-time	ContextNet* [9]	768×1280	1.0	37.5	44.5
	FAST-SCNN* [14]	768×1280	1.2	14.9	47.9
	SCMNet*	**768×1280**	**1.2**	**38.3**	**51.2**

TABLE IX
PERFORMANCE EVALUATION ON CAMVID VALIDATION AND TEST SETS

Model	Input size	Parameters (Million) (%)	Val. class mIoU (%)	Test class mIoU (%)
DeepLab [1]	-	262.1	-	61.6
PSPNet [8]	-	250.8	-	69.1
SegNet [28]	360×480	29.5	-	60.1
ENet [11]	360×480	0.4	-	60.3
ICNet [10]	720×960	6.68	-	67.5
BiseNet [6]	720×960	49.0	-	68.7
FAST-SCNN* [14]	640×896	1.2	73.3	65.8
ContextNet* [9]	640×896	1.0	69.6	61.9
SCMNet*	**640×896**	**1.2**	**78.6**	**71.3**

is an offline model being 12 times larger than our proposed model.

G. Performance comparison on Camvid

Table IX clearly displays that our proposed model achieves the state-of-the-art result on Camvid [22] validation and test sets. Out-of 32 classes, we trained our model with 11 classes. We also trained FAST-SCNN [14] and ContextNet [9] under same system configuration and presented validation and test mIoU in the table. The test results of other models are obtained from the literature. After 1000 epochs, our proposed model generates 78.6% validation and 71.3% test mIoU, which sets a new record. Furthermore, our model has less parameters and GFLOPs compare to the other models, and this makes our model more efficient than others.

H. Qualitative performance analysis

For qualitative assessment, this section presents selected output images from several models in Figures 4, 5, 6, 7. It can be clearly seen that the quality of the proposed SCMNetoutput is much better than the output generated by other models.

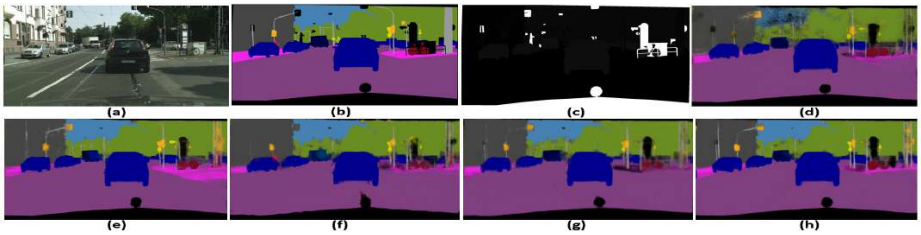

Fig. 4. All models prediction on Cityscapes validation set. (a) RGB input, (b) Colored annotation, (c) Single channel annotation with 19 classes, (d) bayesian SegNet, (d) DeepLab, (e) ContextNet, (f) FAST-SCNN, (g) FANet, (h) SCMNet

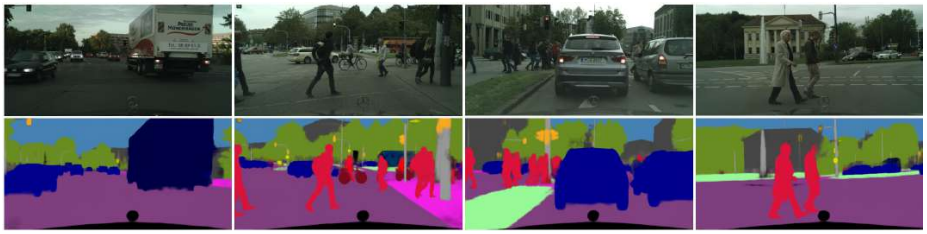

Fig. 5. Output by SCMNetusing Cityscapes test set samples

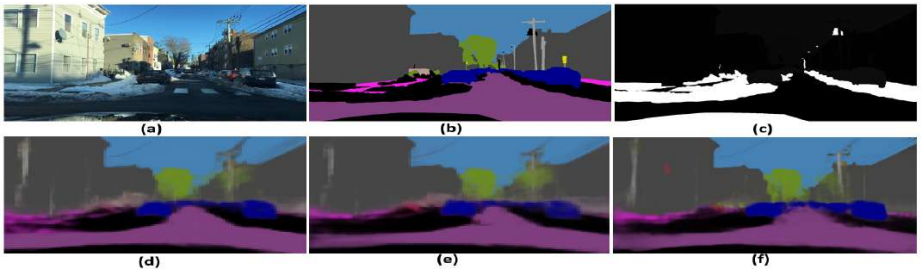

Fig. 6. All models prediction on BDD100K validation set. (a) RGB input, (b) Colored annotation, (c) Single channel annotation with 11 classes, (d) ContextNet, (e) FAST-SCNN, (f) SCMNet

More boundary degeneration, noisy feature effects and overlapping classes can be observed in other models' output. Due to the diverse nature of the BDD100K dataset, the quality of the output by all models is not as good as Cityscapes, however the boundary degeneration problem is better addressed by the proposed model compared to ContextNet and FAST-SCNN. Similar observations can be made on the Camvid outputs (Figure 7). Some classes, such as traffic sign and wall, are misclassified by ContextNet and FAST-SCNN, while the proposed SCMNetassigned the classes correctly. Hence, in line with the quantitative results, our qualitative analysis also indicates performance superiority of SCMNetin the field of real-time scene parsing.

V. Conclusion

We proposed an efficient and optimized shared-branch semantic segmentation model capable of handling high resolution input images. Positioning our proposed CMM at three different locations in encoder refines more the contextual details of feature map before it propagates to next level and the proposed multi-directional DS-FFM provide accurate object localization in the entire scene. Due to the shared branches at encoder side, our proposed backbone enhances the entire feature hierarchy by propagating rich contextual information from bottom to top. The proposed model is tested over three publicly available benchmarks and the outcomes of our experiments clearly demonstrate the advantages of

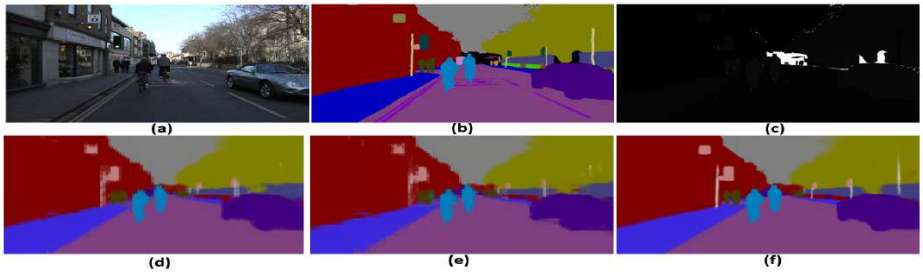

Fig. 7. All models prediction on Camvid validation set. (a) RGB input, (b) Colored annotation, (c) Single channel annotation with 11 classes, (d) FAST-SCNN, (e) ContextNet, (f) SCMNet

SCMNetover several existing real-time scene parsing models. Our proposed model produced the state-of-the-art result on Camvid dataset among all existing real-time semantic segmentation models. Presently, we evaluated our model with outdoor scenes, especially urban street scenes. However, in future we plan to test the proposed model for indoor scene analysis. The implementation of our proposed model is publicly available at https://github.com/tanmaysingha/SCMNet.

ACKNOWLEDGEMENT

The authors acknowledge Pawsey supercomputing centre and the School of EECMS for valuable support of this research work.

REFERENCES

[1] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE TPAMI*, vol. 40, no. 4, pp. 834–848, 2017.

[2] G. Lin, A. Milan, C. Shen, and I. Reid, "Refinenet: Multi-path refinement networks for high-resolution semantic segmentation," in *Proc. CVPR*, 2017, pp. 1925–1934.

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[4] A. Tao, K. Sapra, and B. Catanzaro, "Hierarchical multi-scale attention for semantic segmentation," *arXiv preprint arXiv:2005.10821*, 2020.

[5] S. Targ, D. Almeida, and K. Lyman, "Resnet in resnet: Generalizing residual architectures," *arXiv preprint arXiv:1603.08029*, 2016.

[6] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "Bisenet: Bilateral segmentation network for real-time semantic segmentation," in *Proc. ECCV*, 2018, pp. 325–341.

[7] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proc. ECCV*, 2018, pp. 801–818.

[8] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proc. CVPR*, 2017, pp. 2881–2890.

[9] R. P. Poudel, U. Bonde, S. Liwicki, and C. Zach, "ContextNet: Exploring context and detail for semantic segmentation in real-time," *arXiv preprint arXiv:1805.04554*, 2018.

[10] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, "ICNet for real-time semantic segmentation on high-resolution images," in *Proc. ECCV*, 2018, pp. 405–420.

[11] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "Enet: A deep neural network architecture for real-time semantic segmentation," *arXiv preprint arXiv:1606.02147*, 2016.

[12] Y. Hong, H. Pan, W. Sun, Y. Jia *et al.*, "Deep dual-resolution networks for real-time and accurate semantic segmentation of road scenes," *arXiv preprint arXiv:2101.06085*, 2021.

[13] R. Luo, Y. Cao, Y. Jin, and Y. Li, "A lightweight network for fast semantic segmentation," in *Proc. BESC*. IEEE, 2020, pp. 1–6.

[14] R. P. Poudel, S. Liwicki, and R. Cipolla, "Fast-SCNN: fast semantic segmentation network," *arXiv preprint arXiv:1902.04502*, 2019.

[15] T. Singha, D.-S. Pham, and A. Krishna, "FANet: Feature Aggregation Network for Semantic Segmentation," in *Proc. DICTA*. IEEE, 2020, pp. 1–8.

[16] T. Singha, D.-S. Pham, A. Krishna, and J. Dunstan, "Efficient segmentation pyramid network," in *Proc. ICONIP*. Springer, 2020, pp. 386–393.

[17] J. He, Z. Deng, and Y. Qiao, "Dynamic multi-scale filters for semantic segmentation," in *Proc. ICCV*, 2019, pp. 3562–3572.

[18] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *Proc. CVPR*, 2018, pp. 8759–8768.

[19] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proc. CVPR*, 2018, pp. 4510–4520.

[20] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. CVPR*, 2017, pp. 1251–1258.

[21] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. CVPR*, June 2016.

[22] G. J. Brostow, J. Fauqueur, and R. Cipolla, "Semantic object classes in video: A high-definition ground truth database," *Pattern Recognition Letters*, vol. 30, no. 2, pp. 88–97, 2009.

[23] A. Sergeev and M. Del Balso, "Horovod: fast and easy distributed deep learning in tensorflow," *arXiv preprint arXiv:1802.05799*, 2018.

[24] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[25] S. Choi, J. T. Kim, and J. Choo, "Cars can't fly up in the sky: Improving urban-scene segmentation via height-driven attention networks," in *Proc. CVPR*, 2020, pp. 9373–9383.

[26] A. Kendall, V. Badrinarayanan, and R. Cipolla, "Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding," *arXiv preprint arXiv:1511.02680*, 2015.

[27] Z. Wu, C. Shen, and A. v. d. Hengel, "Real-time semantic image segmentation via spatial sparsity," *arXiv preprint arXiv:1712.00213*, 2017.

[28] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE TPAMI*, vol. 39, no. 12, pp. 2481–2495, 2017.

[29] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, "BDD100k: A diverse driving dataset for heterogeneous multitask learning," in *Proc. CVPR*, 2020, pp. 2636–2645.

.5 Publication 5

Multiagent and Grid Systems 1 (2020) 1–5
IOS Press

Urban Street Scene Analysis Using Lightweight Multi-level Multi-path Feature Aggregation Network

Tanmay Singha [a,*], Duc-Son Pham [b] and Aneesh Krishna [b]
[a] *School of Electrical Engineering, Computing and Mathematical Sciences, Curtin University, Perth, Western Australia, AUstralia*
E-mail: tanmay.singha@postgrad.curtin.edu.au
[b] *School of Electrical Engineering, Computing and Mathematical Sciences, Curtin University, Perth, Western Australia, Australia*
E-mails: dspham@ieee.org, A.Krishna@curtin.edu.au

Abstract. Urban street scene analysis is an important problem in computer vision with many off-line models achieving outstanding semantic segmentation results. However, it is an ongoing challenge for the research community to develop and optimize the deep neural architecture with real-time low computing requirements whilst maintaining good performance. Balancing between model complexity and performance has been a major hurdle with many models dropping too much accuracy for a slight reduction in model size and unable to handle high-resolution input images. The study aims to address this issue with a novel model, named M2FANet, that provides a much better balance between model's efficiency and accuracy for scene segmentation than other alternatives. The proposed optimised backbone helps to increase model's efficiency whereas, suggested Multi-level Multi-path (M2) feature aggregation approach enhances model's performance in the real-time environment. By exploiting multi-feature scaling technique, M2FANet produces state-of-the-art results in resource-constrained situations by handling full input resolution. On the Cityscapes benchmark data set, the proposed model produces 68.5% and 68.3% class accuracy on validation and test sets respectively, whilst having only 1.3 million parameters. Compared with all real-time models of less than 5 million parameters, the proposed model is the most competitive in both performance and real-time capability.

Keywords: DCNN, Semantic Segmentation, encoder-decoder, feature map, dilated convolution

1. Introduction

Semantic scene segmentation is an important task in scene analysis wherein each pixel of an input image is assigned a class label. These labels or classes could include a wide range of objects, such as people, car, table, building, train, etc. depending on the data sets. This study focuses on urban street images which contain objects relevant to outdoor street scenes, such as traffic light, traffic sign, car, train, rider, etc.

Over the last decade, extensive research has been conducted in this field and it has been shown that in robotics, video surveillance and autonomous car driving industries, semantic segmentation plays a key role. It helps the machine segment an image automatically, but how efficiently and accurately machine can perform this task, is always a critical question.

*Corresponding author. E-mail: tanmay.singha@postgrad.curtin.edu.au.

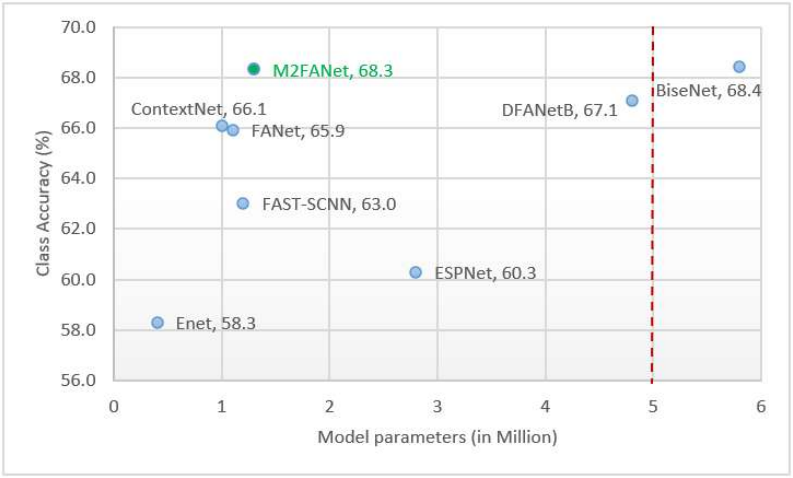

Fig. 1. **Number of parameters vs Class mIoU on Cityscapes test set**- Proposed M2FANet sets a new state-of-the-art performance among semantic segmentation models of having less than 5M parameters.

Previous studies have shown that deep convolution neural networks (DCNNs) are widely used in various domains such as classification [1], [2], object detection [3], instance and semantic segmentation [4], [5], [6], [7]. Due to their high robust nature and handling capability of rich semantic features, they often produce promising results on various publicly available benchmarks [8], [9], mostly under an offline setting with very expensive GPU computing clusters. However, achieving a good performance in a real-time environment, especially for resource-constrained devices, is still a big challenge for all researchers. Over the decade, several real-time semantic segmentation models [10], [11], [12], [13] have been developed to address semantic segmentation for computer vision embedded devices, but the inability of handling high-resolution input images hampers model performance. High-resolution input demands large resources if the model is too deep. To maintain a balance between model complexity and input resolution, the authors of [14], [12], [11] introduce a multi-branch approach in which the deep branch is used for capturing rich contextual information whereas the shallow branch is responsible for retaining boundaries/edges details. Nevertheless, mutually independent branches hardly contribute to the learning ability of the model and also the addition of the shallow branch with high-resolution input image slows down its performance. To speed up the inference speed, some studies [10], [4] prune the redundant channels from the model architecture. Such approach boosts inference speed of the model by costing model's performance.

For semantic segmentation, rich contextual details along spatial and channel dimensions are important. These contextual information can be extracted by a series of convolution layers. The common approach is to adopt a popular DCNN model as a feature extractor to extract details from the input. Many segmentation models use ResNet [15] as a backbone due to its high scalability and robustness. To improve prediction accuracy, several variants of ResNet model are proposed [16], [17]. However, with the in-

crease in depth and width of the model, the number of parameters and floating point operations (FLOPs) also increase drastically which makes the model incapable of handling high-resolution input in real-time environments. Various off-line segmentation models, such as OCR [18], DeepLab (all variants), [19], [20], use ResNet in the encoder as a feature extractor. However, for real-time execution, any model will need a light-weighted backbone to reduce the computational cost. The MobileNet architecture [21], [22] fulfills this requirement, thus it is suitable for embedded devices having low hardware specifications. Due to the optimised structure of residual blocks of MobileNet, it always results in less parameters and FLOPs, hence less memory usage. For that reason, many real-time scene segmentation models, such as ContextNet [14], FAST-SCNN [23], FANet [24], have adopted MobileNet residual blocks. Even in the field of image classification [2] and object detection [3], MobileNet is also widely used. Inspired by the optimized and robust design of MobileNet's architecture, current research designs a light-weighted backbone network by assembling a series of residual bottleneck blocks found in MobileNet [21]. It will help the proposed semantic segmentation model to achieve the desired balance for real-time applications using resource-constrained devices.

Traditionally, an encoder-decoder design is employed in semantic segmentation in which rich contextual feature maps are extracted by the encoder and a series of upsampling layers are used in the decoder to reconstruct the scene and produce the output of the same size as the input. When using upsampling layers, a model can lose certain rich contextual details due to large semantic gap between levels. To reduce this semantic gap and localize the contextual information better, this research proposes a new technique, called Multi-level Multi-path Feature Fusion (M2-FF) method for semantic segmentation. This technique is discussed in detail subsequently. In pursuit of better contextual representation, this

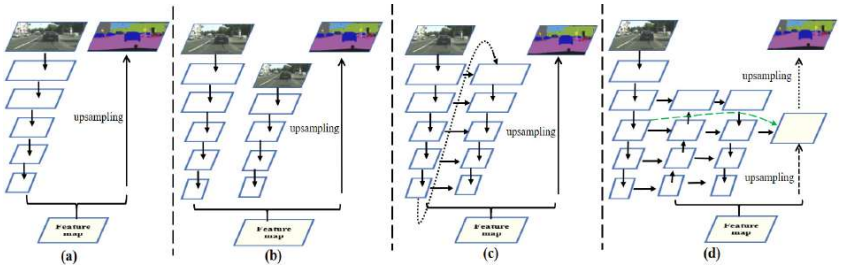

Fig. 2. Different approaches. From left to right: (a) One-branch encoder, (b) Multi-branch encoder, (c) Feature reuse in sub-encoder (d) M2-FF decoder. This approach uses semantic features at different levels from encoder network and progressively map them in the multiple paths at decoder end. Dotted arrows (green color) in the last path defines the presence of feature aggregation path in which features from encoder will be mapped with the features at current path by skip connections.

study exploits different feature scaling techniques such as Pyramid Pooling Module (PPM) [25] and Atrous Spatial Pyramid Pooling (ASPP) module [20]. Due to the optimized design and usage of dilated convolutions, ASPP performance is better than PPM and it also provides better object localization facility by maximizing the size of the receptive field. In ablation study section, the details about the model's performance using both PPM and ASPP are discussed.

The complete architecture of the proposed model is presented in Figure 3. The key points of the proposed study are as follows:

- This research presents an optimized light-weighted backbone to extract rich contextual details from high-resolution input images suitable for resource-constrained real-time semantic scene segmentation applications.
- It also introduces a new, yet effective, feature aggregation technique (M2-FF) at the decoder end for better region localization and context engrossment.
- It shows a better way to maximize the receptive field for capturing the contextual details from the feature map without adding any extra parameters.
- The proposed model is tested and evaluated on two different publicly available benchmarks on urban street scenes.
- The results of the study clearly state that compared to the existing semantic segmentation models of less than 5 million parameters, the proposed model achieves state-of-the-art results on Cityscapes [9] dataset (Figure 1). The number of parameters and GFLOPs of other competitive real-time models are at least five times higher than the proposed model which makes the proposed model (M2FANet) superior than the existing real-time scene segmentation models for resource constrained embedded devices.

2. Related Work

Semantic scene segmentation models typically follow a pyramid architecture. In the encoder stage, a deep CNN typically computes a feature hierarchy layer by layer and develops an inherent multi-scale pyramid shape. At the decoder end, the high-semantic feature map is up-sampled and fused with the previous layers' feature maps through lateral connections to recover higher spatial dimensions. Once spatial details are completely extracted, the model will predict the class label for each pixel to complete the segmentation process. Different networks, such as VGG [7], ResNet [15], Xception [26], and MobileNet [21], are often used as a feature extractor in the encoder end.

2.1. Off-line Segmentation

Most existing semantic scene segmentation models are off-line models due to their deep architectural design. For example, Deeplab [19], [20], PSPNet [25], OCR [18], [27] use deep ResNet101 [16] network as an encoder which has a large number of parameters and GFLOPs. Other DCNNs, such as ResNet other variants[15], [16], MobileNet [21], [22] and Xception [26], are used for image classification, typically with a fully connected layers added on top of the encoder to produce classification output. Inspired by the outstanding performance of such DCNN models in image classification, Fully Convolutional Network (FCN) [6] was the first one to utilize a deep neural network for semantic segmentation. It replaces the fully connected layers from the top of the DCNN by a convolution layer to generate a spatial map and feeds it to the decoder circuit in order to produce segementation output. Based on this foundation, several models, such as UNet [28], SharpMask [29], RefineNet [30], DeepLab [19], are then developed by introducing lateral connections between the low-level feature maps across resolutions and semantic levels. Inheriting the benefits of feature scaling technique at different scales, several other approaches such as PSPNet [25], DeepLabV3+ [20] , ParseNet [31] are also developed in off-line semantic segmentation field.

2.2. Real-time Segmentation:

Due to the growing demand of designing real-time applications in the field of classification [32], clustering [33], segmentation [5], [10] and object detection [3], considerable researches have been conducted to optimize the existing off-line models and to produce acceptable real-time performance on resource-constrained devices. SegNet [4] was one of the pioneering models targeting real-time computation by introducing a small architecture. Later on, ENet [10] proposed an extremely efficient framework by reducing the number of down-sampling operations and introducing high dilation rate in bottleneck blocks. It helps the model reduce the number of parameters and GFLOPs drastically; however, the accuracy of this model is severely compromised. Few more studies, such as ICNet [11], ContextNet [14], BiSeNet [12], GUN [34], were developed to improve the real-time performance. All these models introduced multi-branch approach in which only the deep branch was used for feature extraction, and other branches were employed to capture resolution and boundary details. However, independent branches did not contribute in the learning ability of the model.

By considering the drawback of the independent branches, FAST-SCNN [23] introduced a new technique, called Down-Sampling. Instead of using a completely separate branch from the beginning, it deployed a down-sampling module at the initial stage of the pipeline. This module reduced the input resolution to a quarter of the original input size. After this module, it created two branches - a deep branch for feature extraction and a shallow branch for preserving texture and pattern details. This approach reduced the computational cost and enhances the performance. However, due to less depth of the shallow branch and large semantic gaps between local and global features, this model lacks the ability to retain boundaries information.

In contrast to these approaches, DFANet [35] introduces the concept of sub-networks in the encoder. Traditionally, an encoder follows pyramid structure in which a stack of convolution layers processes an input image and reduces its spatial dimensions as the input reaches to the end. This low resolution high semantic feature map is used as an input for decode. However, in contrast to this traditional approach, DFANet [35] uses this semantic feature as an input for next sub-encoder network and thus the process is repeated for the third sub-network. Layers at the same level but in different sub-encoders share their gradient information among themselves. This approach produced new state-of-the-art results in real-time scene segmentation. However, it still has as many as 7.8 million parameters and large GFLOPs due to multiple sub-networks at the encoder end.

Figure 2 shows the architectural difference of these different approaches. Figure 2(c) clearly shows that [35] upsamples a rich semantic feature map of first encoder by a large factor and reuses it in the next sub-encoder. Upsampling the feature map by a large factor causes a loss in spatial details. Hence, it sets lateral connections from the same level of at different sub-encoders to overcome this loss. This approach shows the flow of feature maps in one direction (top to down) which fuses global features with next lower-level local features by ignoring accurate localization signals through the bottom-up path. This leads to a large semantic gap at the decoder. To reduce this large semantic gap between the spatial dimensions of the local feature map and global feature map, this study develops a new technique, called M2-FF, in the decoder side. The far right diagram (d) of Figure 2 illustrates the proposed approach.

This approach is inspired by FPN [36] which introduces a multi-level feature fusion technique. By adding a top-down path and lateral connections among layers at same level, it tries to reduce the semantic gap between semantic feature maps of different levels. Later on, PAN [37] shows that adding a top-down path reduces semantic gap but introduces a localization issue unfortunately. To boost the localization capability of the entire feature hierarchy, [37] proposes another path from bottom to top

to fuse local features with global features. By adopting this technique, an efficient, scalable object detection model, named EfficientDet [3], has recently been introduced by the Google Brain team. Both studies [37][3] have shown that the addition of a bottom-up path provides better object localization and context accumulation in the feature maps. Motivated by this approach, current study introduces a novel Multi-level Multi-path feature fusion approach at the decoder end in this work.

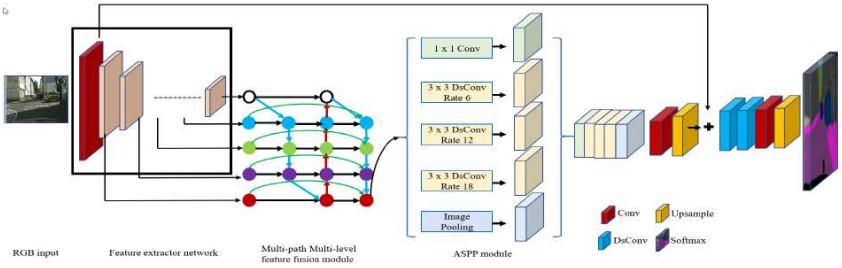

Fig. 3. Complete pipeline of M2FANet

3. Proposed Method

In this section, the complete architecture of the proposed model is discussed. By exploiting dilated convolution, depth-wise separable convolution (DsConv), feature scaling and multi-stage feature fusion techniques, this study significantly extends the preliminary research [24] without drastic a increase of parameters and computational cost.

3.1. Network Architecture

The overall architecture of the proposed M2FANet model is shown in Figure 3. In what follows, the detail of every component of the model, including the backbone network, M2-FF, ASPP, classifier module, dilated convolution, depth-wise separable convolution, and non-linearity functions is addressed.

3.1.1. Encoder Network

Since the main focus of this research is to design an optimized scene segmentation model for resource-constrained devices, it employs the mobile residual bottleneck block (MBConv) of MobileNetV2 [22]. In the preliminary investigation FANet [24], a down-sampling module is deployed at the beginning of the network to reduce the input resolution to 1/8 of the original input size before passing the tensor to residual blocks. However, in the proposed design, it is replaced by MBConv blocks with less channels. Utilization of down-sampling module generates less parameters and GFLOPs. It mainly controls the input resolution and boundary details by not contributing much for holding the spatial details from the input scene. On the other hand, it has shown that deploying MBConv of different expansion ratios at the initial stage preserves more contextual and spatial details due to its squeeze and excitation architecture [22]. Although MBConv block generates more parameters and GFLOPs compare to the down-sample

module, it still has less channels at the initial stage and hence controls the increase of model parameters and GFLOPs.

The proposed study uses two types of bottleneck residual block of MobileNetV2 [22]- MBConv1 and MBConv6, which are named based on their expansion ratio. It is the ratio between the size of the input bottleneck and the inner size, and is either 1 or 6. Each block contains an input followed by several bottlenecks, then followed by an expansion. To reduce the number of parameters, Depth-wise Conv (DwConv) layer is deployed at the expansion stage. Table 1 displays the structure of an MBConv block. Here, h, w, c and c' denote the spatial dimensions of tensors, t denotes the expansion ratio, and s denotes the stride. Non-linearity activation is deployed in the first two layers as as suggested in [22]. However, to retain meaningful information, it is skipped after the last layer of each MBConv. The filter size of the first block is 24 and it is progressively increased in the successive blocks. It is shown in [22] that the use of width multipliers from 0.35 to 1.25 for all resolutions generally produces a better performance. Based on this strategy, the proposed study uses multipliers between 0.35 and 0.5 to set the number of channels in the successive MBConv blocks. It avoids higher values of the width multiplier to control model width and make it computationally efficient for real-time applications. To exploit the architectural advantage

Table 1
Bottleneck residual block

Input	Operator	Output
$h \times w \times c$	1×1 Conv,1/1, Relu	$h \times w \times tc$
$h \times w \times tc$	3×3 DwConv, 3/s, Relu	$h/s \times w/s \times tc$
$h/s \times w/s \times tc$	1×1 Conv,1/1, -	$h/s \times w/s \times c'$

of [38], this research uses a squeeze and excitation module in each residual bottleneck block and test the model's performance. In the result section, it compares the performance of MobileNetV2 and MobileNetV3 residual blocks. Previous publications [39], [38] demonstrate that the addition of the squeeze and excitation module in the bottleneck architecture improves performance; however, the complete architecture of the network totally depends on Network Architecture Search (NAS) [40], which makes network design unpredictable. It also requires high CPU/GPU power to leverage neural architecture search for automatically predicting feature network design. To keep the proposed network design simple, predictable and usable for real-time computation, the final proposed model uses MBConv blocks of MobileNetV2. Experimental results also confirm that the inverted residual block of MobileNetV2 performs better than MobileNetV3 blocks. Moreover, the addition of squeeze and excitation module of MobileNetV3 results in a large number of parameters and GFLOPs, which makes the model more difficult to run in a real-time environment.

The layer architecture of the backbone network is displayed in Table 2. It clearly displays that an input image passes through maximum seven stages in the encoder side to produce a global feature map. For an input image of 1024×2048, the final feature map at encoder end will be 8×16, which may lose contextual details due its low resolution. To address this issue, an effective technique, called Multi-level Multi-path feature fusion is introduced which fuses features of last five stages in the multiple paths. Previous work [20] suggests that an output stride of 2^4 in encoder would be the best for semantic segmentation. However, with 2^4 output stride and a full resolution input, real-time computation would be prohibitive. A global feature map of high resolution (64×128 px for 2^4 stride) leads to a high consumption of memory, making model infeasible to run in resource constrained devices. The work in [20] proposes a very deep off-line semantic segmentation model which can only handle 513×513 px input, whereas the proposed model accepts input of up to 1024×2048 px. Therefore, to reduce the computational cost and

Table 2
Layer architecture of backbone network

Stage (i)	Input	Operators	Layers (n)	stride(s)	Output
1	1024×2048×3	Conv, k3×3	1	2	512×1024×32
2	512×1024×32	MBConv1, k3×3	1	2	256×512×24
	256×512×24	MBConv6, k3×3	2	1	256×512×32
3	256×512×32	MBConv6, k3×3	3	2	128×256×48
4	128×256×48	MBConv6, k3×3	3	2	64×128×64
5	64×128×64	MBConv6, k3×3	3	2	32×64×96
6	32×64×96	MBConv6, k3×3	2	2	16×32×128
7	16×32×128	MBConv6, k3×3	1	2	8×16×160

enable the model to run in a resource-constrained computer vision system such as embedded devices, the proposed method uses maximum 2^7 stride in the encoder side for high resolution input. Most of the real-time scene segmentation models with high input resolution, has a large output stride. The benchmark Cityscapes dataset [9] provides images of 1024×2048 px, whereas CamVid [41] supplies comparably low- resolution images. To handle low resolution datasets, stride can be reduced up to 2^5.

3.1.2. Multi-level Multi-path feature fusion

The literature has shown that fusing features at different levels helps the model combine rich contextual information with spatial details and reconstruct the image. Higher-level neurons strongly respond to entire objects while low-level neurons more likely capture local texture and patterns which stimulates the necessity of adding a top-down path to propagate rich semantic features from high to low level and enrich all features at different levels with sensible classification knowledge.

Traditionally, a high-level rich semantic feature map F_i ($C_i \times H_i \times W_i$) is up-sampled and mapped with the former low-level features F_{i-1} ($C_{i-1} \times H_{i-1} \times W_{i-1}$) [4], [28], [36] to regenerate the scene. Such up-sampling and mapping process will continue as long as the final prediction of the original input size is not produced. After every upsampling, the model loses certain gradient information. Moreover, one direction propagation does not ensure the localization of each object in the scene. This phenomena clearly manifests the need of another bottom-top path to enable the localization capability of the entire feature hierarchy. By adding a bottom-up path augmentation, PAN[37] preserves the local context. Inspired by the path augmentation techniques presented in [36] and [37], a new feature aggregation technique, called Multi-level Multi-path feature fusion is introduced at the decoder end. The design of this module is also influenced by the design of a technique known as BiFPN, which is used in [3] for object detection. The backbone network of the proposed model produces semantic feature maps at multiple stages. In each stage, the input will be convolved to reduce its size by half. For M2-FF, feature maps from the third stage (S3) to the seventh stage (S7) are used. The complete structure of the feature fusion module is shown in Figure 4. Different from [37][3], the proposed method introduces another top-down path to aggregate rich semantic feature maps of M2-FF with the feature maps generated by the encoder through

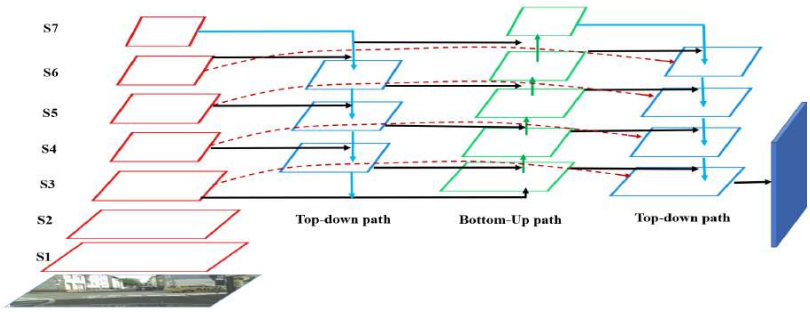

Fig. 4. Design of Multi-level Multi-path Feature Fusion module at decoder. a) Top-down path for multi-scale feature fusion (blue arrows), b) Bottom-up path for better localization (green arrows), c) Top-down path for better feature aggregation with the help of skip connections (red arrows) from backbone. There are lateral connections (black arrows) among the layers at same levels.

lateral connections. This helps the suggested model to retain gradient information from the previous layers.

Formally, given a list of semantic features $F^{in} = (F^{in}_{l_3}, F^{in}_{l_4}, \ldots, F^{in}_{l_i})$ generated by different stages (S3 to S7) in the encoder stage, the aim is to find a transformation f that can effectively map different features at different semantic levels and produce rich semantic feature maps $F^{out} = (F^{out}_{l_3}, F^{out}_{l_4}, \ldots, F^{out}_{l_i})$ at all levels. In the first top-down path, coarse features from the top level (S7) will be up-sampled by factor of 2 and then mapped with the former stage features. Here, simple add function is utilized to map coarse features of two consecutive levels. Thus, this top-down path generates features ($N^{in}_{l_7}$, $N^{in}_{l_6}$,$N^{in}_{l_5}$,$N^{in}_{l_4}$,$N^{in}_{l_3}$) for the next path. In the later path, it starts from the bottom layer (S3) and gradually down-samples features by a factor of 2 before it gets mapped with next higher level feature maps. Thus, this bottom-up path creates features ($M^{in}_{l_3}$, $M^{in}_{l_4}$,$M^{in}_{l_5}$,$M^{in}_{l_6}$,$M^{in}_{l_7}$). Finally, feature aggregation path aggregates these features in top-down direction and also maps with features ($F^{in}_{l_7}$, $F^{in}_{l_6}$,$F^{in}_{l_5}$, $F^{in}_{l_4}$, $F^{in}_{l_3}$) generated by encoder network. Hence, M2-FF produces the output as follows:

$$F^{out}_{l_7} = DsConv(M^{in}_{l_7}), \tag{1}$$

$$F^{out}_{l_6} = DsConv(F^{in}_{l_6}) + DsConv(M^{in}_{l_6}) + DsConv(Upsample(F^{out}_{l_7})), \tag{2}$$

$$F^{out}_{l_5} = DsConv(F^{in}_{l_5}) + DsConv(M^{in}_{l_5}) + DsConv(Upsample(F^{out}_{l_6})), \tag{3}$$

$$F^{out}_{l_4} = DsConv(F^{in}_{l_4}) + DsConv(M^{in}_{l_4}) + DsConv(Upsample(F^{out}_{l_5})), \tag{4}$$

$$F^{out}_{l_3} = DsConv(F^{in}_{l_3}) + DsConv(M^{in}_{l_3}) + DsConv(Upsample(F^{out}_{l_4})). \tag{5}$$

To reduce number of operations and number of parameters, the standard convolution layer is replaced by depth-wise separable convolution layer.

3.1.3. Atrous Spatial Pyramid Pooling

ASPP, introduced by[20], is a powerful tool to explicitly control the spatial dimensions of semantic feature maps produced by DCNNs. It also enhances the ability of DCNNs to handle both large and small objects efficiently by providing a robust mechanism to control the field-of-view of filters in convolution layers.

A dataset usually contains objects of different classes with various sizes. Although DCNNs have shown astonishing capability of classifying an arbitrary region of scene by employing a small kernel, typically 3×3, less frequent and small objects are always overlooked by the presence of large objects in the scene. To overcome this issue, the authors of [19], [20] have shown that objects of an arbitrary scale can be precisely classified by resampling convolutional features with different sampling rates. Motivated by this approach, this study adopts dilated convolution technique with different dilation rates. Formally, an ASPP layer with dilation rate r introduces $(r-1)$ zeros between two consecutive values of the filter in order to enlarge t times the kernel size of a $k \times k$ filter to $k_e = k + (k-1) \cdot (r-1)$ without increasing parameters or GFLOPs. Thus, it provides the best trade-off between small field-of-view and large field-of-view. For an input feature map x, each location i on the output feature map y will be sampled by ASPP as follows:

$$y[i] = \sum_{k=0}^{K} x[i + r \times k]w[k] \tag{6}$$

where the dilation rate r determines the stride with which it convolves the input feature. The filter's field-of-view will be adaptively updated by changing the dilation rate. In this study, ASPP module uses multiple parallel dilated convolution branches with three different dilation rates: 6, 12, and 18. The features extracted from multi-path multi-scale feature fusion module are further processed separately in each atrous branch and then concatenated to each other to produce the final result. Figure 3 demonstrates the whole process of the ASPP module.

Typically, ASPP is deployed on top of the encoder network to control the spatial resolution of the resulting feature maps if the output stride of encoder is 2^4. In the proposed model output stride is 2^7 which means rich semantic feature map with low spatial dimensions. Engrossing rich contextual details with the help of ASPP will be effective if the feature map size is eight to sixteenth times smaller than the original input. High dilation rate at the dilated branch of ASPP helps creating a large receptive field for better localization of small and large objects in the scene. Moreover, it also reduces number of operations drastically in the convolution process. Hence, by looking at the motivation factors of utilizing ASPP, it is deployed on top of the M2-FF module. M2-FF generates final aggregated feature map of 128×256 px which is eight times smaller than the original input. Moreover, high dilation rates (6, 12, and 18) at different branches of feature scaling module process the input quickly with less GFLOPs.

3.2. Classifier module

This is the last module of the whole pipeline. Features produced by the ASPP module are upsampled by a factor of 4 and then simply mapped with the second level (S2) features ($F_{l_i}^{in}$) of the encoder through lateral connections. To avoid the loss of semantic details, every upsample layer is followed by one DwConv layer and a point-wise standard convolution (1×1 Conv). Two 3×3 DsConv layers are exploited

in the classifier module to refine the feature maps, followed by one more upsample layer to produce an output of the same size as the input. Finally, proposed model employs the softmax activation function to assign a class label to every individual pixel. This study uses 19 classes for Cityscapes and 11 classes for CamVid.

3.2.1. Depth-wise Separable Convolution

To achieve the target of designing an efficient scene segmentation model for resource-constrained devices, this study replaces the standard Conv layer by DsConv in many places. In the feature aggregation module, after fusing features of two consecutive levels, it deploys a separable convolution block in which a depth-wise separable convolution is performed, followed by a batch-normalization layer. DsConv factorizes a standard convolution into two stages - a depth-wise convolution (DwConv) and a point-wise convolution (1×1 Conv). It reduces a large number of parameters and GFLOPs. Proposed study also uses dilated depth-wise convolution as it is support by TensorFlow DsConv layer [42]. It uses different dilation rates to increase the size of receptive field.

3.2.2. Nonlinearities

The selection of the activation function in DCNN models has a significant impact on model prediction. It helps the model preserve non-linearity while passing knowledge from one layer to the next layer. The most widely used non-linearity function is Rectified Linear Unit (ReLU) due to its strong convergence rate during gradient descent optimization. Although, other activation functions such as Sigmoid, Tanh are smoother than ReLU, but they have not been as popular as ReLU. Recently, the Google Brain team has proposed a new activation, called `swish` [43]. Unlike ReLU, `swish` is smooth and non-monotonic. Empirically, [43] has shown that `swish` boosts MobileNet [21] performance on ImageNet [44] dataset by 2.2% over ReLU and concludes that for light-weight model, `swish` is the best activation function. It also comes with non-zero cost in real-time environment. Motivated by [43], [2], [3], the proposed study employs `swish` non-linearity after every convolution layer. It is defined as:

$$swish(x) = x \cdot \sigma(x), \tag{7}$$

where $\sigma(x)$ defines sigmoid activation function.

4. Experiment

To evaluate the proposed model in urban street scene analysis, extensive experiments are carried out on two different benchmark datasets: Cityscapes [9], CamVid [8]. Experimental results clearly illustrate that the proposed model outperforms many existing semantic segmentation models which have less than 5 million parameters. In the following subsections, this paper first discusses the datasets and implementation details, followed by a series of ablation studies on the Cityscapes benchmark dataset. Finally, it compares the proposed model with some existing off-line and real-time segmentation models and report the results on both validation and test sets. Consistent with previous work, this study reports model parameters, GFLOPs, class and category meanIoU.

4.1. Datasets

Cityscapes For urban street scene analysis, Cityscapes [9] is the most popular choice for researchers. It is a large-scale dataset, mainly used for object detection, instance, semantic and panoptic segmentation.

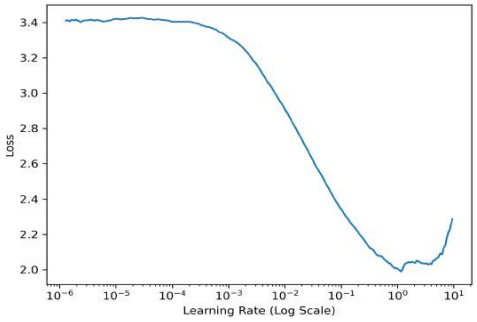

Fig. 5. Finding optimal learning rate

It provides annotated data for more than 30 classes grouped into 8 categories. The dataset consists of around 5,000 fine annotated images and 20,000 coarse annotated ones.

The proposed study mainly used fine-tune images and considered only 19 classes for pixel annotations. The whole dataset is divided into three parts- training set (2,975 images), validation set (500 images) and test set (1,525 images). The labels for the training set and validation set are supplied by the benchmark whereas the labels for test set are not provided. However, test set predictions are submitted to the Cityscapes evaluation server and the results are discussed in the results section.

CamVid It is a small dataset [8], mainly designed for object detection in automated driving vehicle. The images of this dataset were generated from a recorded video, then annotated frames were created by assigning a class colour to each object of the frame by human operators. This dataset contains only 267 images for training, 101 for validation and 233 for testing. Out of 32 classes, current study uses 11 classes (excluding void) for performance evaluation. Due to the small size of the dataset, models were usually under-learned.

4.2. Implementation Details

To conduct this experiment, this study uses a dual Nvidia GeForce RTX 2080Ti GPUs system, each GPU has 11GB of memory. To exploit the parallel processing power of GPUs, it uses CUDA 10.2. The proposed model is developed using `tensorflow` 2.1.0 and `keras` 2.3.1. To utilize both GPUs in data-parallel distributed training environment, it employs the `horovod` framework [42]. `Horovod` takes a single-GPU `tensorflow` program and trains it on multiple GPUs. For instance, in this research, `horovod` divides the whole training set into two sets and runs the training script on individual set in each GPU. Thus, it boosts the run-time performance by effectively utilizing all resources. Stochastic gradient decent (SGD) is used as model optimizer with a momentum of 0.9.

Inspired by [19, 21, 25], the proposed study uses the 'poly' learning rate by setting 0.045 as the base value and 0.9 as the power. To find out the optimal learning rate in each epoch during training the model, model is trained for 5 epochs using a polynomial scheduler and the corresponding losses against different learning rates are plotted. Thus, the upper and lower bound of learning rate for training are set. Figure 5 illustrates the plot of learning rate vs model loss as an example. To calculate model loss, the categorical cross entropy loss function is exploited.

Following the suggestion by MobileNetV2, the proposed model uses l_2 regularization with value 0.00004 for top layers except depth-wise convolution layer. To avoid model over-fitting due to limited data, various data augmentation techniques such as random horizontal flip, vertical flip, random crop, resizing and adjusting brightness, saturation and contrast of images are applied. Finally, a dropout layer is deployed before the final classification in the training process. Adding dropout layer helps to regulate model over-fitting issue and also improve validation accuracy. After training the model several times with different dropout rates, it has been noticed that a dropout rate less than 0.2 cannot address the over-fitting issue whilst a value more than 0.6 value causes under-fitting. Therefore, taking the average of these two values as the dropout rate would be beneficial for model prediction.

4.3. Ablation study

Previous study [24] has shown the effectiveness of using modified design of BiFPN for region localization and context aggregation. Taking that as a starting point for this study, the design of FANet is updated for the better scene segmentation. Table 3 shows the evaluation of the proposed M2FANet model. Clearly, it produces better results than the existing real-time scene segmentation models having less than 5 million parameters.

Table 3
Results of ablation study on Cityscapes validation set

Method	Backbone	Feature fusion module	Feature scaling method	Mean IoU (%)	Parameters (Million)
FANet	9 MBConv6	FPN	-	62.6	1.2
FANet	9 MBConv6	BiFPN	-	65	1.1
FANet	9 MBConv6	Modified BiFPN	-	65.9	1.1
M2FANet	MobileNetV2	M2-FF*	ASPP	67.7	2.7
M2FANet	MobileNetV3*	M2-FF*	ASPP	3.7	67.5
M2FANet	15 MBConv6	M2-FF*	-	67.4	1.25
M2FANet	15 MBConv6	M2-FF*	PPM	68.0	1.37
M2FANet	**15 MBConv6**	**M2-FF***	**ASPP**	**68.5**	**1.28**

*In MobileNetV3, it uses MBConv blocks with squeeze and excitation blocks. This study redesigns the architecture of the modified BiFPN design and proposes a new design: M2-FF.

The first three rows of Table 3 show the preliminary result of FANet already presented in [24]. From the forth row, it shows the additional results obtained from the ablation study using the proposed model. At the initial stage, the backbone of FANet is replaced by an ImageNet pre-trained MobileNetV2 model and performance is measured on Cityscapes validation set. MobileNetV2 [22] has 12 MBConv blocks with increasing channel sizes (16 to 320). The literature shows that increasing the depth and width of the model will likely enhance the performance. However, it also increases model parameters and number of operations. Specifically, MobilenetV2 has more than 2.6M parameters, and this increases model's overall parameters and GFLOPs. Similarly, the fifth row of Table 3 shows that the use of MobileNetV3 [38] as feature extractor of the new design supplements model performance, but at the cost of a drastic increase in computation. Moreover, the addition of squeeze and excitation modules in residual blocks of MobileNetV3 is determined by Neural Architecture Search (NAS) [39] which makes model structure unpredictable. Therefore, optimizing the backbone architecture is really difficult for MobileNetV3.

The above observation motivates this study to stick with MBConv blocks of MobileNetV2. The depth and width of the FANet model is modified by increasing the number of residual blocks and width multiplier. The proposed model replaces down-sampling module of FANet by three MBConv blocks of different expansion ratios and last two MaxPooling layers of FANet model are also replaced by three MBConv blocks. Table 3 clearly illustrates that due to the addition of new blocks, the number of parameters is increased by 0.2M; however, the model's performance is boosted by 2.6%. To control the number of parameters of MBConv blocks, the proposed design uses the width multiplier between 0.35 and 0.5 to increase model's width. Table 3 also shows that without using any feature scaling technique, M2FANet enhances the performance by 1.5% compare to FANet. However, empirically it has been proved that feature scaling at different branches with different scaling rates can further escalate the model's performance.

This study explores two powerful feature scaling techniques: PPM [25] and ASPP [20]. It is noticeable from Table 3 that both techniques boost the performance by a recognised percentage. Compare to PPM, utilizing ASPP at the decoder side is more effective due to its design and the presence of dilated convolution branches. In PPM, features are processed by four ImagePooling branches with different rates whereas in ASPP, one ImagePooling branch, one 1x1 Convolution, and three 3x3 dilated convolution branches process the features with different dilated rates. Such an architecture provides a large receptive field for localization and context assimilation. The design of ASPP can be visualized in Figure 3. By exploiting all these techniques, the current study finally extends the design of FANet with a new backbone, M2-FF and ASPP modules, and produces 68.5% mean IoU on the Cityscapes validation set.

4.4. Model Evaluation

This section presents the performance of the proposed model on the Cityscapes dataset and compares model's performance with other existing off-line and real-time scene segmentation models. It presents model parameters, GFLOPs, pixel accuracy, Class and category mean Intersection Over Union (mIoU) on validation and test sets. It also demonstrates inference time and FPS (frame per second) of the models which are trained under the same system configuration. This study did not pre-train the model with ImageNet [44] dataset. Note that the domain of ImageNet dataset is different from urban street scenes. Due to this different domain knowledge, this study focuses urban street scene benchmarks and trains

Table 4
Model performance on Cityscapes Fine-tune and Coarse datasets

Model	Input	Dataset	Class MeanIoU (%)	Category MeanIoU(%)
FANet [24]	512×1024	Fine-Tune only	59.7	81.4
M2FANet	512×1024	Fine-Tune only	62.3	83.6
M2FANet	512×1024	Fine-Tune + Coarse	62.6	83.8

the model with related domain knowledge datasets. Cityscapes [9] provides 5,000 fine-tune and 20,000 coarse annotated images for training the model. The current study reports both results of fine-tune and fine-tune with weakly annotated data (Table 4) at 512×1024 px. Due to hardware limitation, the proposed model could not be trained with fine-tune and coarse datasets together at full input resolution. It is evident from Table 4 that the performance is slightly improved by 0.3% due to the additional coarse dataset. It is expected that the performance could be further improved by 0.5-1% on the validation and

test sets if the model is trained with both coarse and fine-tune datasets together at full resolution. This study aims to address this in the future.

It also trains the proposed M2FANet on the fine-tune datasets at different input resolutions and present the results in Table 5. At 512×1024 px input resolution, the feature extractor extracts feature maps of 4×8 px, which is very small. Therefore, study suggests not to use input image smaller than this size. Basically, the proposed model targets high-resolution input (1024×2048) for real-time computation. For small resolution input, it can replace stride by 1 in last two blocks on encoder network. However, little modification is also required in the M2-FF design. Table 5 illustrates accuracy of M2FANet for each class at different input resolutions. In the top 5 classes (road, sky, vegetation, building, car), the model's accuracy is more than 90%, and its accuracy is less than 50% only in four classes (wall, fence, rider and motorcycle). Due to uneven class distribution in the dataset, the accuracy among all classes varies like many other methods. For instance, motorcycle class has less than 0.1% pixel in whole dataset which causes low prediction accuracy (33.4%) for this class. Table 5 also illustrates that its performance is also improved at higher input resolutions.

Table 5
Class-wise M2FANet performance on validation set at different input resolutions

Input Size	Road	S.walk	Build.	Wall	Fence	Pole	T.light	T.sign	Veg.	Terrain
1024×2048	**94.9**	76.6	**90.1**	45.7	48.5	55.8	58.9	70.3	**90.7**	56.1
	Sky	Person	Rider	Car	Truck	Bus	Train	M.cycle	Bicycle	**mIoU**
	93.3	74.2	47.8	**92.1**	69.1	73.6	61.8	33.6	68.1	**68.5**
Input Size	Road	S.walk	Build.	Wall	Fence	Pole	T.light	T.sign	Veg.	Terrain
768×1536	**94.7**	74.1	88.2	43.5	45.1	48.6	53.9	68.3	**90.2**	53.3
	Sky	Person	Rider	Car	Truck	Bus	Train	M.cycle	Bicycle	**mIoU**
	90.8	74.5	46.2	**90.4**	50.5	68.9	48.2	36.5	66.8	**64.9**
Input Size	Road	S.walk	Build.	Wall	Fence	Pole	T.light	T.sign	Veg.	Terrain
512×1024	**94.2**	73.7	87.6	34.2	41.8	54.3	50.9	66.2	**90.4**	52.2
	Sky	Person	Rider	Car	Truck	Bus	Train	M.cycle	Bicycle	**mIoU**
	89.4	73.4	44.6	**90.2**	37.7	61.0	34.7	38.2	68.6	**62.3**

Table 6
Category-wise M2FANet performance on Cityscapes validation dataset

Input size	Flat	Construction	Object	Nature	Sky	Human	Vehicle	mIoU
1024×2048	**94.7**	89.6	62.4	**91.0**	**93.3**	75.5	**90.3**	85.3
768×1536	**94.4**	88.5	60.3	**90.7**	**90.8**	75.6	88.6	84.1
512×1024	**94.5**	87.7	59.7	**90.4**	89.4	75.8	88.1	83.7

Table 6 displays the category-wise model performance at different input resolutions. In the Cityscapes dataset, all classes are distributed into seven categories: flat, construction, object, nature, sky, human, and vehicle. The proposed model's performance is outstanding in five categories (flat, construction, nature, sky and vehicle) across all resolutions. , However, the object category has a low accuracy (59.7-62.4%) across all input sizes, possibly due to asymmetrical class distribution and tinny shape of traffic sign, traffic light and pole.

4.5. Performance Comparison on Cityscapes

4.5.1. Results on Validation Set

To compare the proposed model with other existing models, this study trained one off-line model (DeepLab) and three real-time segmentation models (SegNet, ContextNet, FAST-SCNN) under same system configuration. Results of comparison on the validation set are displayed in Table 7. Models marked with the sign * are trained during this study where either public code is available or sufficient implementation details are known. For other models, results are extracted from the literature. Due to a large number of parameters and GFLOPs, standard convolution layers of DeepLab and Segnet are replaced by DsConv layers. For DeepLab, instead of using ResNet101 or VGG-16, a pre-trained Xception model is used as a backbone. This study also incorporated ASPP on top of the encoder as a dense feature extractor. Due to the large size of DeepLab, this study could not train the model at full input resolution. It trained Deeplab with 512×1024 px input and achieved 58.2% validation mean IoU. However in [19], the authors claimed 62.97% accuracy using VGG-16 and large field of view. Similarly, this study achieved

Table 7
Performance evaluation of different models on cityscapes validation set

Type	Model	Input Size	Parameters (Million)	GFLOPs	Class meanIoU (%)	Category meanIoU (%)	Pixel Accuracy (%)
Off-line	PSPNet [25]	713 × 713	250.8	516	81.2	-	-
	HANet [27]	768 × 768	65.4	2138.0	80.3	-	-
	OCR [18]	769 × 769	10.5	340	81.2	-	-
	DeepLab* [19]	512 × 1024	37.9	845.9	58.2	87.7	88.6
Real-time	SegNet* [4]	512 × 1024	29.5	680.4	56.8	77.2	87.3
	ContextNet* [14]	512 × 1024	1.0	9.3	54.2	78.3	87.3
	FAST-SCNN* [23]	1024 × 2048	1.2	14.9	63.3	82.2	89.4
	ENet [31]	1024 × 2048	0.4	3.8	58.3	-	-
	ICNet [11]	1024 × 2048	6.68	58.5	67.7	-	-
	DFANet-B [35]	1024 × 1024	4.7	31.3	58.2	-	-
	FANet* [24]	1024 × 2024	1.1	11.4	65.9	83.6	89.6
Real-time	**M2FANet***	**1024×2048**	**1.3**	**37.3**	**68.5**	**85.3**	**90.2**

*Models marked with * sign are trained by us and empty cell means that data are not found in literature.

Table 8
Efficiency comparison of all trained models

Model	Input size	Inference time (sec.)	FPS	Model size (MB)
DeepLab	512 × 1024	0.35	2.9	90.2
SegNet	512 × 1024	0.76	1.3	225.4
ContextNet	512 × 1024	**0.09**	**11.1**	8.8
FAST-SCNN	512 × 1024	0.11	9.1	9.4
FANet	512 × 1024	0.1	10.0	9.4
M2FANet	512 × 1024	**0.11**	**9.1**	11

56.8% and 54.2% class accuracy for SegNet [4], ContextNet [14] respectively with 512×1024 px input

size. It trained FAST-SCNN [23] model with full input resolution for 1000 epochs with fine-tune dataset of Cityscapes. Table 7 clearly depicts that FASt-SCNN [23] produces better results among the existing models listed in the table, although its performance is lagged by 5-6% compared to the proposed model. This study did not achieve the result claimed by [23]. The implementation of FAST-SCNN by this study is consistent with the information provided in the paper. The implementation of all compared models presented here is readily available on the official GitHub repository. To enable a fair comparison based on the models only, this research did not use any post-processing techniques.

It is noticeable from Table 7 that the proposed M2FANet generates better segmentation results on Cityscapes validation set among all the listed models. Although, ICNet [11] produces an accuracy (67.7%) very close to M2FANet, it has much higher GFLOPs and model parameters than the proposed one. This makes the proposed M2FANet superior than other real-time scene segmentation alternatives. This study achieves 68.5% mean IoU on the Cityscapes validation set without using any post processing techniques. In [35], the authors claimed around 70% class accuracy on Cityscapes validation set; however, the number of parameters is 6 times higher than M2FANet and thus it is not as efficient as the proposed model. Moreover, after studying [35] architecture, it is observed that this model has a much higher GFLOPs than the proposed model. To have a reasonable comparison, current work compares the proposed model performance with DFANet-B variant which has 4.8 million parameters. Table 7 shows that the proposed model performs much better while having less parameters and GFLOPs. This study also found that the authors of [35] claimed much smaller GFLOPs of their best model. However, the current study suggests that it is not possible to have low GFLOPs count with large number of parameters (7.8 million). It calculated the GFLOPs count and reported in Table 79. In off-line scene segmentation model, PSPNet [25], HANet [27], [18] achieve outstanding results (above 81%) due to its deep neural architecture.

The current work also presents the inference time and the rate in terms of FPS (frame per second) of the models which are trained under the same system configuration. It is often the case in the literature that such comparison of inference time and FPS is based on different hardware, which makes it less insightful as these indicators can vary significantly. It also depends on the size of input image and different hyper-parameters of the model. Therefore, comparing the inference time of different models ran in different hardware platforms with different input resolutions does not provide a clear picture model's superiority in terms of efficiency. Therefore, to present a balanced comparison, this study measures the inference time of all the trained models under the same system configuration and with similar input size. The results are reported in Table 8. It demonstrates that ContextNet [14] is quite efficient overall among all the trained models. It is understandable as ContextNet has the least number of parameters among all the listed models in Table 8. It can process on an average 11 frames per second where as the proposed model can process 9.1 frames per second. The inference time is measured using a single Nvidia GeForce RTX 2080Ti GPU system with the same input size. Table 8 also presents model size which is the size of the model after completing the whole training process.

4.5.2. Results on test set

To compare with other existing models, this study exhibits the test set results in Table 9. The results of other models are extracted from their original papers and Cityscapes leader-board. It displays that all off-line and real-time semantic segmentation models except ContextNet [14], FAST-SCNN [23], ENet [10], have at least four times higher number of parameters and GFLOPs than the proposed model. In addition, they all achieved a lower prediction accuracy network than M2FANet did. Table 9 also demonstrates that very few existing real-time models can handle full-resolution input images. Among all listed models, ICNet [11] produced 69.5% class accuracy on full-resolution test images of Cityscapes,

Table 9
Performance evaluation of different models on cityscapes test set

Model	Input size	Class meanIoU (%)	Category meanIoU (%)	Number of parameters (in Million)	GFLOPs
DeepLabV3+ [20]	512×1024	82	91.6	54.8	344.9
DeepLabV2 [19]	512×1024	70.4	86.4	37.9	845.9
PSPNet [25]	713×713	81.2	90.6	65.5	516
BiSeNet [12]	768×1536	74.7	-	5.8	14.8
SegNet extended [4]	360×640	56.1	79.8	29.5	286
ENet [10]	360×640	58.3	80.4	0.4	3.8
ICNet [11]	1024×2048	69.5	-	6.68	
FCN 8S [6]	512×1024	65.3	85.7	57	136.2
SQ [45]	1024×2048	59.8	84.3	-	-
CRFasRNN [46]	512×1024	62.5	82.7	-	-
ESPNet [13]	512×1024	60.3	82.2	0.4	-
ContextNet [14]	1024×2048	66.1	82.8	1.0	18.4
DFANet-B [35]	1024×1024	67.1	-	4.9	31.3
FAST-SCNN [23]	1024×2048	63.0	84.7	1.2	14.9
FANet [24]	1024×2048	64.1	83.1	1.1	11.4
M2FANet	$\mathbf{1024 \times 2048}$	**68.3**	**86.6**	**1.3**	**37.3**

*Empty cell means that data are not found in literature.

whereas the proposed model generates 68.3% accuracy. However, it is distinctly noticeable that ICNet has almost 5 times higher number of parameters than M2FANet. This makes ICNet less favourable for real-time applications. The proposed M2FANet has less parameters and less GFLOPs compared to most models.

Table 9 also presents GFLOPs count of all the models. It is noticeable that instead of counting GFLOPs at full input resolution, most of the models report their GFLOPs count at low input resolution. As GFLOPs count depends on input size, it is expected that the GFLOPs count will be lower in this setting. This study reports 37.3 GFLOPs count of the proposed model with 1024×2048 input resolution, whereas most of the models reported it at low resolution. ENet [10] has the lowest GFLOPs count, however it is reported at 360×640 input resolution.

4.6. Performance Comparison on CamVid

Current research also evaluates M2FANet on the CamVid dataset. Table10 shows the class accuracy and pixel accuracy of M2FANet and other compared models on this dataset. It is clearly evident that M2FANet outperforms the other models: it achieves 60.0% class accuracy despite of having small number of training images. To have a fair and reproducible comparison between the different models based

Table 10
Performance evaluation on validation set of CamVid dataset

Model	Input	Class MeanIoU (%)	Pixel Accuracy(%)
SegNet	360×480	55.6	-
ENet	360×480	51.3	-
DFANet-B	720×960	59.3	-
FAST-SCNN*	512×1024	57.5	86.9
FANet*	512×1024	57.8	87.1
M2FANet*	**640×896**	**60.0**	**88.7**

*empty cell means that results are not found in the literature.
Models marked with * sign are trained by this study.

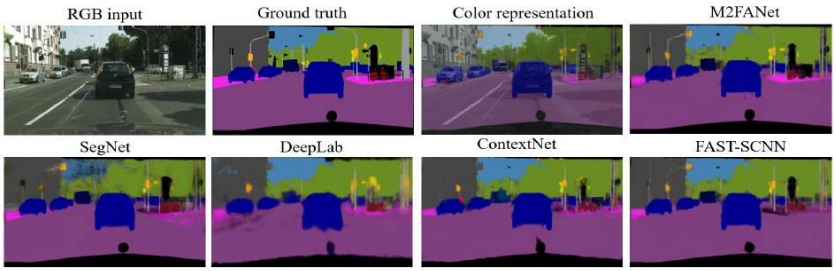

Fig. 6. Output by different models using Cityscapes validation image

on their core architecture, the present study did not use any data augmentation techniques and any other similar datasets in this experiment. Of course, one would expect an increase in performance should any other techniques are used. For example, some literature achieved such improvement by many extras, one of which was additional 3,000 training images from other datasets outside CamVid. However, these results are difficult to reproduce due to the lack of details in these papers. For completeness, this study quotes the original figure from previous publications for those models which are not trained by the current study. It is expected that the proposed model will definitely achieve even better performance when trained with these extra images from related data with similar annotations.

4.7. Qualitative performance comparison

For qualitative assessment, this section presents output images from selected models in Figures 6, 7, 8.

4.7.1. Prediction on Cityscapes benchmark

Figure 6 shows segmented output produced by different models on the validation set of Cityscapes benchmark. The first image is the RGB input followed by ground truth, colour code and predicted outputs by different models. It clearly displays that the output image quality generated by M2FANet is much better than other models and this has been already verified by the quantitative results in Table 7. Similar observation can be drawn from Figure 7. The first row shows RGB test images from Cityscapes, whereas

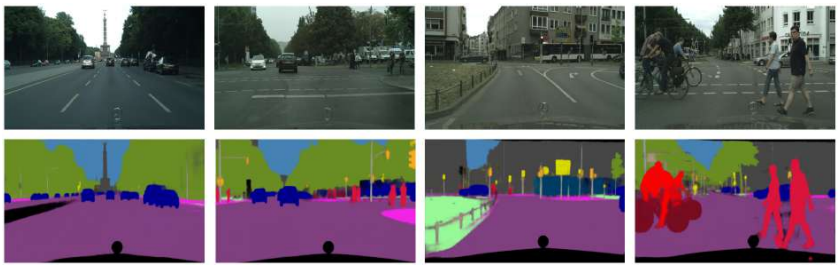

Fig. 7. Output by M2FANet using Cityscapes test images

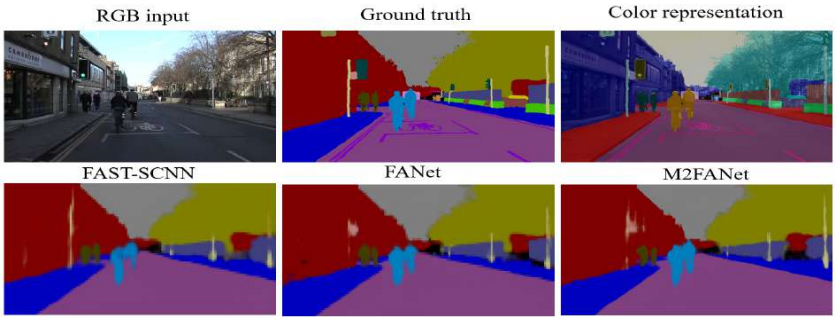

Fig. 8. Output by FAST-SCNN, FANet and M2FANet using CamVid test image

second row displays segmented output by M2FANet. It is observed that the proposed model is capable of identifying and localizing every objects in the scene whilst not overlooking tinny objects such as traffic light, traffic sign, and pole.

4.7.2. Prediction on CamVid benchmark

Figure 8 exhibits the output produced by FAST-SCNN, FANet and the proposed M2FANet on selected images from CamVid validation set. In contrast to Cityscapes, here models are trained with 12 classes (including background class) on this dataset. It is distinctly visible that output by M2FANet is much better than the output generated by FAST-SCNN and FANet. Especially, boundaries of every class are much smoother in the proposed model output compared to others. Though the current study notes that Camvid is much smaller than other scene segmentation datasets and this generally reduces the performance of any models, an observation that has also been established in previous works. Again, one may address this limitation by using extra images from other datasets or using data augmentation techniques, which will benefit all models. However, the focus of the current research is on a fair comparison between the models and thus this direction could be pursued elsewhere.

5. Conclusion

This study presented a light semantic scene segmentation model, named M2FANet for resource-constrained devices, capable of handling high-resolution input images in the real-time environment. The goal of this research was to develop a model that achieves the best performance among models with low computational requirements, which is defined as having less than 5 million parameters. The optimized backbone network and proposed multi-level multi-path feature aggregation module at decoder efficiently produce a new state-of-the-art results compare to existing models in the same category. Current research also demonstrates the usefulness of feature scaling and feature fusion techniques for better localization and contextual engrossment. The proposed model is evaluated on two public scene segmentation benchmarks. The results show that the proposed model is suitable for urban street scene analysis in real-time mode. In the future, this study plans to extend the proposed model for indoor scene analysis and expand the evaluation on other public benchmark datasets. For reproducing the results presented in this work, this study has made the implementation of the proposed model and selected models on the official Github repository https://github.com/tanmaysingha/M2FANet.

References

[1] Y. Liu, L. Ji, R. Huang, T. Ming, C. Gao and J. Zhang, An attention-gated convolutional neural network for sentence classification, *Intelligent Data Analysis* **23**(5) (2019), 1091–1107.

[2] M. Tan and Q.V. Le, Efficientnet: Rethinking model scaling for convolutional neural networks, *arXiv preprint arXiv:1905.11946* (2019).

[3] M. Tan, R. Pang and Q.V. Le, EfficientDet: Scalable and Efficient Object Detection, *ArXiv* **abs/1911.09070** (2019).

[4] V. Badrinarayanan, A. Kendall and R. Cipolla, Segnet: A deep convolutional encoder-decoder architecture for image segmentation, *IEEE transactions on pattern analysis and machine intelligence* **39**(12) (2017), 2481–2495.

[5] R. Lalchhanhima, G. Saha, M.V. Nunsanga and D. Kandar, Synthetic aperture radar image segmentation using supervised artificial neural network, *Multiagent and Grid Systems* **16**(4) (2020), 397–408.

[6] J. Long, E. Shelhamer and T. Darrell, Fully convolutional networks for semantic segmentation, in: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[7] K. Simonyan and A. Zisserman, Very deep convolutional networks for large-scale image recognition, *arXiv preprint arXiv:1409.1556* (2014).

[8] G.J. Brostow, J. Fauqueur and R. Cipolla, Semantic object classes in video: A high-definition ground truth database, *Pattern Recognition Letters* **30**(2) (2009), 88–97.

[9] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth and B. Schiele, The Cityscapes Dataset for Semantic Urban Scene Understanding, in: *Proc. CVPR*, 2016.

[10] A. Paszke, A. Chaurasia, S. Kim and E. Culurciello, Enet: A deep neural network architecture for real-time semantic segmentation, *arXiv preprint arXiv:1606.02147* (2016).

[11] H. Zhao, X. Qi, X. Shen, J. Shi and J. Jia, Icnet for real-time semantic segmentation on high-resolution images, in: *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 405–420.

[12] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu and N. Sang, Bisenet: Bilateral segmentation network for real-time semantic segmentation, in: *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 325–341.

[13] S. Mehta, M. Rastegari, A. Caspi, L. Shapiro and H. Hajishirzi, Espnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation, in: *Proceedings of the european conference on computer vision (ECCV)*, 2018, pp. 552–568.

[14] R.P. Poudel, U. Bonde, S. Liwicki and C. Zach, Contextnet: Exploring context and detail for semantic segmentation in real-time, *arXiv preprint arXiv:1805.04554* (2018).

[15] S. Targ, D. Almeida and K. Lyman, Resnet in resnet: Generalizing residual architectures, *arXiv preprint arXiv:1603.08029* (2016).

[16] Z. Wu, C. Shen and A. Van Den Hengel, Wider or deeper: Revisiting the resnet model for visual recognition, *Pattern Recognition* **90** (2019), 119–133.

[17] X. Zhang, Z. Li, C. Change Loy and D. Lin, Polynet: A pursuit of structural diversity in very deep networks, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 718–726.

[18] Y. Yuan, X. Chen and J. Wang, Object-contextual representations for semantic segmentation, *arXiv preprint arXiv:1909.11065* (2019).

[19] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy and A.L. Yuille, Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs, *IEEE transactions on pattern analysis and machine intelligence* **40**(4) (2017), 834–848.

[20] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff and H. Adam, Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation, in: *Proc. ICCV*, 2018.

[21] A.G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto and H. Adam, Mobilenets: Efficient convolutional neural networks for mobile vision applications, *arXiv preprint arXiv:1704.04861* (2017).

[22] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov and L.-C. Chen, Mobilenetv2: Inverted residuals and linear bottlenecks, in: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[23] R.P. Poudel, S. Liwicki and R. Cipolla, Fast-scnn: fast semantic segmentation network, *arXiv preprint arXiv:1902.04502* (2019).

[24] T. Singha, D. Pham and A. Krishna, FANet: Feature Aggregation Network for Semantic Segmentation, in: *Proceedings of The International Conference on Digital Image Computing: Techniques and Applications, DICTA 2020, Melbourne, Australia, November 29 - December 2, 2020*, IEEE, 2020, pp. 1–8. doi:10.1109/DICTA51227.2020.9363370.

[25] H. Zhao, J. Shi, X. Qi, X. Wang and J. Jia, Pyramid scene parsing network, in: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2881–2890.

[26] F. Chollet, Xception: Deep learning with depthwise separable convolutions, in: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.

[27] S. Choi, J.T. Kim and J. Choo, Cars Can't Fly Up in the Sky: Improving Urban-Scene Segmentation via Height-Driven Attention Networks, in: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9373–9383.

[28] O. Ronneberger, P. Fischer and T. Brox, U-net: Convolutional networks for biomedical image segmentation, in: *International Conference on Medical image computing and computer-assisted intervention*, Springer, 2015, pp. 234–241.

[29] P.O. Pinheiro, T.-Y. Lin, R. Collobert and P. Dollár, Learning to refine object segments, in: *European Conference on Computer Vision*, Springer, 2016, pp. 75–91.

[30] G. Lin, A. Milan, C. Shen and I. Reid, Refinenet: Multi-path refinement networks for high-resolution semantic segmentation, in: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1925–1934.

[31] W. Liu, A. Rabinovich and A.C. Berg, Parsenet: Looking wider to see better, *arXiv preprint arXiv:1506.04579* (2015).

[32] P.K. Mallick, S.H. Ryu, S.K. Satapathy, S. Mishra, G.N. Nguyen and P. Tiwari, Brain MRI image classification for cancer detection using deep wavelet autoencoder-based deep neural network, *IEEE Access* **7** (2019), 46278–46287.

[33] R. Callister, M. Lazarescu and D.-S. Pham, RobustRepStream: Robust stream clustering using self-controlled connectivity graph, *Intelligent Data Analysis* **24**(4) (2020), 799–830.

[34] H. Zhang, H. Zhang, C. Wang and J. Xie, Co-occurrent features in semantic segmentation, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 548–557.

[35] H. Li, P. Xiong, H. Fan and J. Sun, Dfanet: Deep feature aggregation for real-time semantic segmentation, in: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9522–9531.

[36] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan and S. Belongie, Feature pyramid networks for object detection, in: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[37] S. Liu, L. Qi, H. Qin, J. Shi and J. Jia, Path aggregation network for instance segmentation, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8759–8768.

[38] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan et al., Searching for mobilenetv3, in: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1314–1324.

[39] T.-J. Yang, A. Howard, B. Chen, X. Zhang, A. Go, M. Sandler, V. Sze and H. Adam, Netadapt: Platform-aware neural network adaptation for mobile applications, in: *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 285–300.

[40] G. Ghiasi, T.-Y. Lin and Q.V. Le, Nas-fpn: Learning scalable feature pyramid architecture for object detection, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7036–7045.

[41] K. He, X. Zhang, S. Ren and J. Sun, Deep residual learning for image recognition, in: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[42] A. Sergeev and M. Del Balso, Horovod: fast and easy distributed deep learning in TensorFlow, *arXiv preprint arXiv:1802.05799* (2018).

[43] P. Ramachandran, B. Zoph and Q.V. Le, Searching for activation functions, *arXiv preprint arXiv:1710.05941* (2017).

[44] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.

[45] M. Treml, J. Arjona-Medina, T. Unterthiner, R. Durgesh, F. Friedmann, P. Schuberth, A. Mayr, M. Heusel, M. Hofmarcher, M. Widrich et al., Speeding up semantic segmentation for autonomous driving, in: *MLITS, NIPS Workshop*, Vol. 2, 2016.

[46] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang and P.H. Torr, Conditional random fields as recurrent neural networks, in: *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1529–1537.

Mr. Tanmay Singha is currently a PhD research scholar in the field of Computing at Curtin University, Australia. He has double Master's Degrees- Master of Technology in Information Technology and Master of Computer Applications from University of Calcutta, India. Before joining at Curtin University, he served nine years as a lecturer in the department of IT at Royal University of Bhutan, Bhutan. His current research interest focuses on computer vision tasks such as semantic and instance segmentation, object detection, scene graph generation, indoor and outdoor scene analysis, human pose estimation, facial landmark detection and medical image processing using deep neural networks. He has presented and published several scientific research papers in reputed conferences.

Dr. Aneesh Krishna is currently an Associate Professor with the School of Electrical Engineering, Computing and Mathematical Sciences, Curtin University, Australia. He holds a PhD in computer science from the University of Wollongong, Australia. He was a lecturer in software engineering at the School of Computer Science and Software Engineering, University of Wollongong, Australia (from February 2006 - June 2009). His research interests include AI for software engineering, model-driven development/evolution, requirements engineering, agent systems, formal methods, data mining, computer vision, machine learning, bioinformatics and renewable energy systems. He has published more than 130 articles in reputed journals and international conferences. His research is (or has been) funded by the Australian Research Council (ARC), and various Australian government agencies (like NSW State Emergency Service) as well as companies such as Woodside Energy, Amristar Solutions, Autism West Incorporated, BW Solar Australia, Western Australia Dementia Training Center and Andrew Corporation. He serves as an assessor (Ozreader) for the ARC. He has been on the organising committee, served as invited technical program committee member of many conferences and workshop in the areas related to his research.

Dr. Duc-Son Pham received the PhD degree from Curtin University of Technology in 2005. He is currently a Senior Lecturer with the Discipline of Computing, Curtin University, Perth, Western Australia. His current research interests include sparse learning theory, large-scale data mining, convex optimization, and advanced deep learning with applications to computer vision and image processing. He is a Senior Member of the IEEE. He is a recipient of the Young Author Best Paper Award 2010 for a publication in IEEE Transactions on Signal Processing.

.6 Publication 6

2022 International Conference on Digital Image Computing: Techniques and Applications (DICTA) | 978-1-6654-5642-5/22/$31.00 ©2022 IEEE | DOI: 10.1109/DICTA56598.2022.10034629

SC-CrackSeg: A Real-time Shared Feature Pyramid Network for Crack Detection and Segmentation

Tanmay Singha, Moritz Bergemann, Duc-Son Pham, and Aneesh Krishna
School of Electrical Engineering, Computing and Mathematical Sciences
Curtin University, Perth, Western Australia
tanmay.singha@postgrad.curtin.edu.au, moritz.bergemann@student.curtin.edu.au,
dspham@ieee.org, a.krishna@curtin.edu.au

***Abstract*—Detecting cracks is important in a number of civil engineering applications. Recent advances in computer vision have enabled automatic crack detection and fine-grained segmentation using deep learning. However, the models used in previous work are often large and are therefore mainly suitable for offline structure monitoring where images taken from a site are analysed later by a powerful computer. In this work, we address the segmentation problem in an online setting, which permits the use of mobile inspection devices such as drones with limited computing power to monitor structures independently in real-time. We propose SC-CrackSeg, which has a very small number of parameters and can provide very high segmentation accuracy. Our main contribution is a multi-branch information-sharing architecture that efficiently manages global perspective while maintaining the fine and high-resolution details key in crack detection. SC-CrackSeg extends a previously proposed model but optimized specifically for this application: reduction to a single-input, a more efficient context mining module, and a simpler feature fusion module. We evaluate SC-CrackSeg on large crack detection data sets and the results show that our proposed model is competitive against the existing methods.**

***Index Terms*—semantic segmentation, crack detection, multi-branch, feature fusion, real-time models**

I. Introduction

Crack segmentation is a growing area of research that applies semantic segmentation, the pixel-wise classification of images, to the identification of structural cracks. This process helps automate the labour-intensive task of structural crack identification while providing significantly more detail compared to pure classification or object detection approaches. Fully convolutional neural networks (FCNs) [21] have been most successful in this task. As compared to traditional computer vision approaches, FCNs are far more agnostic to dataset feature distributions as they do not rely on specifically crafted feature extractors [20]. Therefore, many approaches have applied existing semantic segmentation architectures [10] [8] to crack segmentation. There are some key issues specific to the crack segmentation task, however. The fine and thin nature of cracks means features must be maintained at high resolution to avoid the loss of crack boundaries, and the small proportion of input image pixels that actually contain cracks results in class imbalance (as seen in Figure 1). These problems have been addressed in existing approaches by maintaining high feature resolutions in separate model "branches" [20] [22], and by applying specialised losses that encourage classification of the minority class [31] [20].

(a) Original crack image (b) Crack label

Fig. 1: Crack image and corresponding segmentation mask from the CFD dataset [28]. Note the white 'crack' class only makes up $\sim 1.5\%$ of pixels in the mask.

Computational efficiency is also valuable for crack segmentation due to the large scale of image data that may need to be scanned for cracks, and as real-world approaches would likely require segmentation to be performed on resource-constrained embedded devices, such as those attached to a UAV [18]. Many approaches still suffer from the high computational cost of pixel-wise segmentation [23], though methods focusing on efficiency have been introduced [9] [17].

We have identified the use of a separate branch for maintaining high-resolution features as similar to the approach of many low-resource semantic segmentation architectures [33] [25]. Particularly, the maintaining of key features at high resolution via repeated upsampling from a lower-resolution branch (as applied in MR-CrackNet [22]) is similar to the information sharing scheme applied in SCMNet [29]. In SCMNet, deep (low-resolution) and shallow (high-resolution) features are repeatedly merged and feature-extracted before being fed back into their respective branches, maintaining both global and local features while permitting both to learn from one another. We posit applying this approach to the crack segmentation task would allow for the efficient learning of high-resolution crack features while retaining the computational efficiency such approaches often struggle to achieve. Therefore, we propose a modified version of SCMNet optimised for the crack segmentation task. We move from a two-input model to a single-input model to improve efficiency and simplicity of implementation. We modify SCMNet's Context Mining Module (CMM) by

eliminating its image pooling components which contribute to boundary degeneration - especially significant with crack segmentation's fine boundaries and shapes. We also deploy a simplified feature fusion module, inspired by BiSeNet [33], to aggregate the final features extracted from the two branches.

In summary, our key contributions are as follows:

- We introduce SC-CrackSeg, a new efficient crack segmentation architecture, inspired by SCMNet [29].
- We demonstrate SC-CrackSeg's performance and versatility on a publicly available combined crack dataset.
- We confirm SC-CrackSeg's state-of-the-art performance via direct comparison against existing architectures.

II. Related Work

A. Semantic Segmentation

Semantic segmentation is a fundamental research area in computer vision. Early approaches such as ENet [23], ESPNet [30] focused on lowering the resolution of features within the model, though this particularly affected the segmentation quality of fine boundaries. Recent approaches have improved the performance by denoting different components of the segmentation task to different branches of the model ICNet [35]. BiSeNet [33] identified two such branches, one deep and one shallow, as being most efficient. ContextNet [25] maintains this overall architecture while introducing depthwise separable convolutions [16] and bottleneck residual blocks [27] for the high (shallow) and low (deep) resolution branches respectively. SCMNet [29] proposes that the deep and shallow branches each produce valuable information the other may exploit, and enables this by introducing repeated information sharing between the branches.

B. Crack Segmentation

Crack detection has been a growing area of research in recent years [14]. Zhang *et al.* [34] proposed an early CNN for performing crack detection on a dataset of 500 crack images. While many approaches initially focused on object detection and classification, the majority of recent studies apply semantic segmentation [14] due to the additional valuable information provided on crack shape and size. Initial approaches applied existing state-of-the-art segmentation networks or backbones to crack detection, such as U-Net [10] and SegNet [8].

There are two unique issues in crack segmentation compared to many other segmentation tasks. First, the thin and fine-grained nature of cracks means especially dense features must be maintained throughout the model, as any reduction in feature resolution will cause major inaccuracies in the fine crack shapes. Secondly, cracks typically make up a minority of a given input image even when present ($< 5\%$ of pixels), resulting in class imbalance. DeepCrack [20] computes multi-scale features to ensure large and thin cracks are accurately segmented and applies a class-balanced loss computed multiple times throughout the network to address class imbalance. FPHBN [31] instead derives multi-scale information via a feature pyramid, and compares their approach with other models across multiple datasets. MR-CrackNet [22] applies a modified ResNet [15] backbone, maintaining feature resolution by upsampling extracted features after each block. Many of the above approaches are evaluated on unique datasets not publicly available, making comparisons challenging. A number of datasets have been proposed for crack segmentation, including CrackBgData [22], CRACK500 [32], and others [11] [28] [3] [36]. Due to the small number of images available in each dataset, our approach applies a merged combination of images from multiple datasets for training.

C. Efficient Crack Segmentation

Processing efficiency is significant when performing infrastructure crack detection due to the large number of images that must be processed when applied at scale. Various studies [19] [18] have investigated the use of UAVs for high-speed crack segmentation, a system only viable with a highly efficient neural network such as Faster R-CNN [2]. Recent studies have investigated efficient crack detection. SDDNet [9] maintains high-resolution features using a simple skip connection, and utilises depthwise separable convolutions and modified atrous spatial pyramid pooling techniques as introduced in the MobileNets [16] and DeepLabV3+ [6] respectively. STRNet [17] applies multi-head attention to the crack detection task, as well as a squeeze and excitation method for maintaining high feature detail.

III. Proposed Method

A. Shared encoder design

The complete architecture of the proposed SC-CrackSeg is exhibited in Figure 2. Like SCMNet, we adopt an information sharing scheme in the encoder, where features from both branches are periodically concatenated to share useful information. However, instead of having two inputs of different scales, the proposed model accepts only one full-sized input image. This reduces the training time per epoch. Table I displays the layered architecture of the proposed backbone. It shows that a 3×3 ConvX block is deployed to down-sample the input image by two. Each ConvX block contains a standard convolution (Conv), Batch Normalization (BN) and ReLU layer to filter the input by a $k \times k$ kernel. After down-sampling the input, two parallel branches are created: a shallow branch for extracting boundary and texture details and a deep branch for extracting global contextual details from the scene.

Deep branch is designed using the inverted bottleneck residual (MBConv) blocks introduced in [27]. The operation of MBConv block is opposite to the bottleneck residual block of ResNet [15]. The MBconv block first expands the input tensor along the channel dimension by an expansion ratio t and then squeezes it along its channels. To reduce the computational cost at the expansion stage, it introduces a Depth-wise Convolution (DwConv) which contributes at least k^2 times fewer parameters than a standard Conv layer. The layer architecture of the MBConv block is displayed in Table II. Motivated by the optimized design of the MBConv block, we deploy one ConvX block and 11 MBConv blocks at the deep branch of the proposed model. We also deploy a

TABLE I: Layer architecture of deep branch of encoder

Stage	Input	Operation						Output
1	448×448×3	k3×3 ConvX						224×224×24
		Deep branch			Shallow branch			
		Operation	N	Output Channels	Operation	N	Output Channels	
2	224×224×24	k3×3 ConvX	1	32	k3×3 DSConvX	1	32	112×112×32
3	112×112×32	k3×3 MBConv1	1	32	k3×3 DSConvX	1	48	
	56×56×32	k3×3 MBConv6	2	48	-	-	-	56×56×48
Fuse and refine deep and shallow feature by the first shared CRM								
4	56×56×48	k3×3 MBConv6	3	64	k3×3 DSConvX	1	96	
	28×28×64	k3×3 MBConv6	1	96	-	-	-	28×28×96
Fuse and refine deep and shallow feature by the second shared CRM								
5	28×28×96	k3×3 MBConv6	3	128	k3×3 DSConvX	1	160	
	14×14×160	k3×3 MBConv6	1	160	-	-	-	14×14×160
Fuse and refine deep and shallow feature by the third shared CRM								
6	14×14×160	MaxPooling	1	160	-	-	-	7×7×160

MaxPooling layer on top of the deep branch to create an additional stage. Thus, the deep branch consists of 5 stages.

TABLE II: Bottleneck residual block

Input	Operator	Output
$h \times w \times c$	1×1 Conv,1/1, ReLU	$h \times w \times tc$
$h \times w \times tc$	3×3 DwConv, 3/s, ReLU	$h/s \times w/s \times tc$
$h/s \times w/s \times tc$	1×1 Conv,1/1, -	$h/s \times w/s \times c'$

Shallow branch is mainly designed for extracting the local features such as boundary and texture details. To achieve this, we ensure a low-channel and simple design for this branch. We deploy four DSConvX (Depth-wise Separable Convolution) blocks. Each contains a DSConv layer, one BN and one ReLU. Each DSConv layer filters the input by a 3×3 kernel with stride 2. A DSConv layer is more efficient than a standard Conv layer at the middle stages of the shallow branch. This can be observed from the equations 1 to 4, which show the total FLOPs (Floating Point Operations) count and the number of parameters associated with a standard Conv and DSConv layer. $(K_w \times K_h \times C_i)$ and $(K_w \times K_h \times C_i - 1)$ in Equation 1 present the amount of multiplication and addition operations performed by a standard Conv layer where K_w, K_h define the kernel size and (C_i, C_o) define the number of channels of input and output feature maps and (H_o, W_o) define the height and width of the output feature map. For a square filter, K_h and K_w can be replaced by K. Comparing the FLOPs and parameter count of a standard Conv and DSConv layer, the latter reduces computational complexity by approximately K^2 times. Hence, it helps us to achieve real-time performance for crack segmentation. We deploy 4 DSConvX blocks to create 4 stages in the shallow branch.

$$\begin{aligned}
\text{FLOPs}_{\text{Conv}} &= [(K_w \times K_h \times C_i) + (K_w \times K_h - 1) + 1] \\
&\quad \times C_o \times H_o \times W_o \\
&= 2 \times K \times K \times C_i \times H_o \times W_o \times C_o \qquad (1) \\
\text{FLOPs}_{\text{DSConv}} &= 2 \times K \times K \times C_i \times H_o \times W_o \times 1 + \\
&\quad 2 \times 1 \times 1 \times C_i \times H_o \times W_o \times C_o \qquad (2) \\
\text{params}_{\text{Conv}} &= [(K \times K \times C_i) + 1] \times C_o \qquad (3) \\
\text{params}_{\text{DSConv}} &= [(K \times K \times C_i) + 1] \times 1 + [(1 \times 1 \times C_i) \\
&\quad + 1] \times C_o \qquad (4)
\end{aligned}$$

Like SCMNet, we deploy three shared points after stage 3, 4 and 5. At each shared point, we fuse and refine the deep and shallow feature maps using a shared Context Refinement

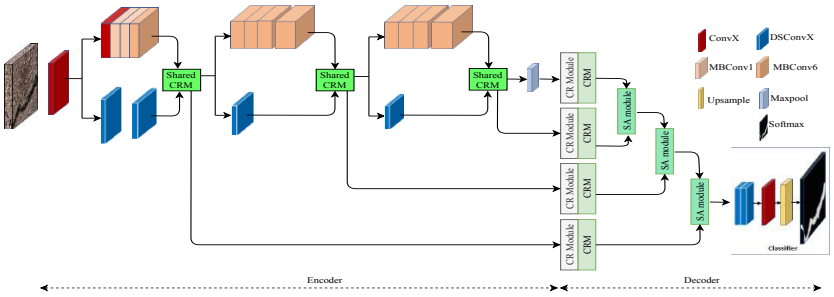

Fig. 2: Complete architecture of SC-CrackSeg

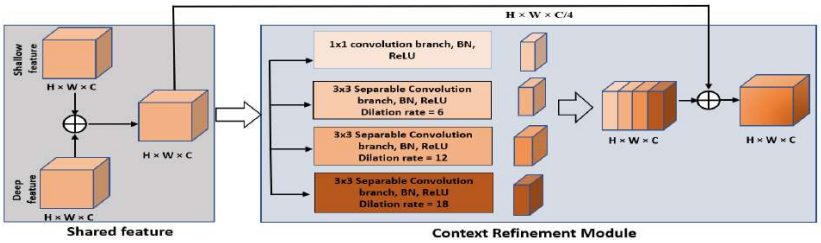

Fig. 3: Shared Context Refinement Module

Module (CRM). The layered architecture of the shared CRM is displayed in Figure 3. It shows that features from deep and shallow branches at the same stages are fused together by a simple element-wise matrix addition operation. Then, it passes through the CRM for better semantic representation. Both feature maps (deep and shallow) have coarse information which needs to be refined properly before it passes to the next level and the design of the CRM provides better receptive fields for coarse-to-fine refinement. The CRM's design is slightly different to SCMNet's Context Mining Module (CMM). In the CMM, out of four parallel branches, two are image pooling branches with a large pool size which can result in a boundary degeneration effect. The pooling branch also does not learn anything as it does not have any parameters. Therefore, we replace the image pooling branches with two additional separable branches. Motivated by the DeepLab's Atrous Spatial Pyramid Pooling (ASPP) [5], we also utilize higher dilation rates (6, 12, and 18) in the separable branches for enhancing the field of view. This allows the model to capture the contextual details of varied sizes of crack in the scene. After processing the input through four different branches, we concatenate all outputs along the channel dimension for better contextualization. In contrast to ASPP, we finally deploy a residual connection between the shared input feature map and the concatenated output for preventing any information loss. Hence, the noisy features are processed and refined by the encoder at three different stages to produce the multi-scale global feature maps.

B. Decoder design

The design of our decoder is completely different to SCMNet. We keep our decoder design simple and effective for real-time crack detection and segmentation. Features from stages 6,5,4, and 3 are used at the decoder. Figure 2 shows how all these feature maps first pass through Channel Reduction (CR) modules to produce a uniform number of channels for each (64). Each CR module has one 1×1 Conv layer, followed by a BN layer. After CR, features are refined by CRM again. Motivated by the simple design of the Feature Fusion Module (FFM) of BiseNet [33], we deploy a Semantic Aggregation (SA) module which aggregates rich features together for

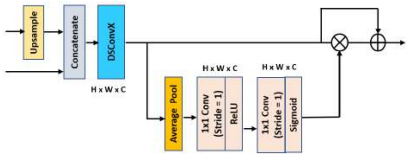

Fig. 4: Semantic Aggregation Module

better region identification. The layered architecture of the SA module is displayed in Figure 4. After refinement, features from $6th$ and $5th$ stages pass through the SA module. Before concatenating the two feature maps, the $6th$ stage features are bi-linearly upsampled. After concatenation, we deploy a DSConvX block instead of a ConvX block. This reduces the number of parameters and FLOPs of each SA module. The rest of the layered architecture is similar to BiSeNet's FFM. We deploy three SA modules for aggregating features at different stages.

Classifier is the last module added on top of the decoder for assigning a class to each pixel of the image based on the classification score. Two 3×3 DSConvX blocks, one standard 1×1 ConvX block, one softmax layer and one upsampling layer with 2^3 scale factor are deployed for building the classifier. Upsampling the feature map by 2^3 times after softmax produces an output of similar size to the input and speeds up the inference time. We also deploy a dropout layer with 0.35 dropout rate and ℓ_2 regularization with 0.00004 lambda value in the DSConv layers to reduce over-fitting.

IV. Experiment

To evaluate the proposed model performance for crack detection, a series of extensive experiments are carried out on publicly available road crack datasets. To compare the proposed model performance with the existing models, we also replicate the design of some existing models and measure their performance on the same datasets in an identical environment.

A. Datasets

Many existing crack detection models are trained with private datasets which are not publicly available. Hence, to conduct our experiment, we use the crack segmentation dataset from the Kaggle data science community. This dataset contains almost 11,200 images with fine-tune annotations. It merges 12 different crack segmentation datasets [3], [11], [28], [31], [34], [36] together to form a single dataset. We consider this the most challenging crack segmentation dataset as it contains various types of cracks on different surfaces, including deep or light cracks on road, pavement, wall, building, and concrete surfaces. The size of all images is $448 \times 448 \times 3$. The whole dataset is divided into two parts: a training set with 9505 images and a test set with 1695 images. We report our test set results in the result section.

B. Performance metrics

For semantic segmentation, mean Intersection Over Union (mIoU) metric [12] is used for performance evaluation. However, here we have only two classes (background and crack). Therefore, alongside mIoU, we also present other necessary metrics such as Precision, recall and accuracy for measuring model performance in crack detection. Due to the highly imbalanced class distribution, the F1, precision and recall metrics are more useful than the accuracy metric.

As this study targets resource-constrained embedded devices for crack detection and segmentation, hence we also report model parameters, GFLOPs (giga Floating Point Operations) and FPS (frames per second) for determining model efficiency.

C. Implementation details

We deploy a polynomial learning rate strategy [5] to find the optimal learning rate (LR) at each epoch. We first train for 5 epochs using a base rate of 0.01 and plot the model loss against LR. We find the maximum change in loss occurs between 0.5 to 10^{-3} LR and therefore set these as the maximum and minimum LR for training. We also use a polynomial learning rate schedule, which decreases LR within this range as the training progresses. To avoid early-stage gradient descent issues in our distributed platform, we deploy gradual warm-up strategy [13]. We use the stochastic gradient descent (SGD) algorithm for optimizing the loss. Different loss functions are explored during the study. Finally, standard data augmentation techniques, ℓ_2 regularization, and dropout are used to avoid model over-fitting.

D. Ablation study

Our encoder design is motivated by the backbone of SCMNet [29]. Therefore, in this section, we first demonstrate the performance of both model's encoder designs and then compare the performance of the complete pipelines of both the models. SCMNet accepts two inputs of different sizes and utilizes the CMM module for feature mining. In contrast, the proposed model accepts one input and uses the CRM module for feature refinement. The first two rows in Table III illustrate the difference in performance when the number of inputs is

TABLE III: Result of ablation study

No. of input in the backbone	CMM	CRM	Decoder	mF1(%)	P(%)	R(%)
Two	✓	-	-	73.0	75.6	70.5
One	✓	-	-	75.9	78.3	73.6
One		✓	-	82.2	84.6	80.0
One		✓	✓	85.3	88.6	82.3

TABLE IV: Comparison of SCMNet to SC-CrackSeg

Model	Param.	GFLOPs	mF1(%)	TT(s)	IT(ms)
SCMNet	1.23M	3.29	83.5	182	8.8
SC-CrackSeg	1.24M	2.79	85.3	93	4.5

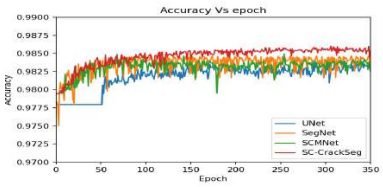

Fig. 5: Accuracy plot of few models against epochs

different at the encoder. The third row shows the proposed encoder's performance with the CRM, showing that use of the CRM increases the model's performance by 6-7%. We replaced SCMNet's decoder with an efficient decoder that not only reduces model size, but also enhances performance by $2-3\%$. The overall comparison between SCMNet and the proposed SC-CrackSeg is exhibited in Table IV. TT and IT denote training time and inference time respectively. Both models were trained under the same system configuration using a batch size of 16 in each GPU. The average IT and TT of the proposed SC-CrackSeg are 4.5 milliseconds (ms) and 93 seconds, almost half of SCMNet's. The proposed model also produces almost 2% more F1 score than SCMNet. Thus, the results of our ablation study helps us to finalize the complete design of our proposed model.

E. Model evaluation

We trained all the models under the same system configuration using the same dataset. This section presents all the outcomes of our study. During data processing, we noticed that the dataset contains noisy pixels which do not fall into either background or crack class category. It can be observed in Figure 6. The first image of the Figure 6 shows the RGB image, the second displays the actual annotation provided by the dataset, and the third exhibits the presence of noisy pixels. The dataset's background and crack classes have pixel values of 0 and 255 respectively, while the noisy pixels range in value between 0-9 and 245-254. We address this by grouping noisy pixels into a void class. For training, We assign 0 as a train-ID to the background class, 1 for crack class and 255 for

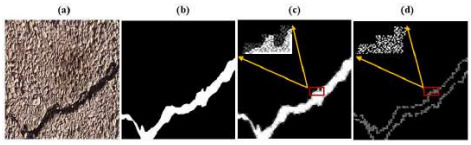

Fig. 6: Processing of dataset. (a) RGB input, (b) Actual annotation, (c) Displaying noisy pixel, (d) Processed annotation

TABLE V: Performance evaluation of different models on crack test set

		IoU			**F1 score**			**Precision**			**Recall**		
Model	Accuracy	BG	Crack	mIoU	BG	Crack	mF1	BG	Crack	aP	BG	Crack	aR
DeepLab [7]	98.43	98.50	52.28	75.39	99.24	68.66	83.95	98.97	79.25	89.11	99.52	60.57	80.05
Unet []	98.18	98.2	48.97	73.58	99.09	65.74	82.42	98.96	70.62	84.79	99.23	61.49	80.20
SegNet [8]	98.27	98.30	52.17	75.23	99.14	68.57	83.85	99.09	71.15	85.12	99.19	66.17	82.63
DeepCrack [20]	**98.52**	98.57	55.89	**77.23**	99.28	71.70	**85.49**	99.13	77.95	84.54	99.44	66.38	**82.91**
MR-CrackNet [22]	97.88	97.19	47.69	72.80	98.95	64.58	81.76	99.11	62.59	80.85	98.79	66.71	82.70
SCMNet [29]	98.39	98.41	51.33	74.87	99.20	67.84	83.52	98.91	77.51	88.21	99.48	60.32	79.30
SC-CrackSeg	**98.52**	98.52	55.55	77.04	99.25	71.43	85.34	99.05	78.14	**88.59**	99.46	65.78	82.32

void. This can be observed in the fourth image of Figure 6. While training all the models, we ignore the void class pixels. Although, void class pixels are not considered to calculate the performance metrics, the presence of void class in training can wrongly influence the model's performance. Hence, we decided to ignore it in training.

Table V displays the performance of all the models which are evaluated under the same system configuration. All the models are trained for a large number of epochs depending on the saturation point, typically 300 epochs as seen in in Figure 5. To avoid multiple overlapping lines, we plot only four models' test accuracy against the number of epochs. Figure 5 also shows that in contrast of other models, the accuracy of UNet [26] was constant for first 50 epochs, and only then began increasing. It seems that at the initial stage, UNet takes time to learn the features from the scene. In Table V we report class-wise all the performance measurement metrics. Among the existing models, DeepCrack [20] produces the best results, followed by DeepLabV3+ [7], SegNet [4], and SCMNet [29]. It produces 77.23% mIoU, 85.49% mean F1 score (mF1), 84.54% average precision (aP) and 82.91% average recall (aR) value. Among all, MR-CrackNet [22] produces low performance on the test set. Our reproduction of FRRN [24] has a similar architecture to MR-CrackNet. Hence, we did not train FRRN model with the crack dataset. Literature [22] also shows that there is less than 0.5% difference between the performance of MR-CrackNet and FRRN models on other private datasets. In comparison to all, the proposed model SC-CrackSeg generates 98.52% accuracy, 77.04% mIoU, 85.34% mF1, 88.59% aP and 82.62% aR, similar performance like DeepCrack. However, DeepCrack is over 12 times larger and 4 times slower than the proposed model. This clearly illustrates the superior performance by the proposed model in terms of real-time inference.

For qualitative performance analysis, we exhibit all models' output in Figure 7 using CFD [28] test sample. Second image of Figure 7 displays the actual annotation which is processed to eliminate the noisy pixels (refer third image of Figure 7). At first glance, it would be difficult to differentiate the output produced by all the models. However, with a close look, it can be observed that DeepCrack [20], DeepLab [5], SCMNet [29], and the proposed SC-CrackSeg generate better curvature of the crack, whereas the output produced by MR-CrackNet [22], UNet [26] and SegNet [4] are more flat and thick. Moreover, the edges of the MR-CrackNet's output exhibit more boundary degeneration as compared to others. Certain sections of the crack, especially the start and end, are not correctly segmented by UNet [26]. Among all outputs, the output of DeepCrack is most close to the actual annotation, followed by DeepLab and the proposed model. However, all the models fail to detect the disjoint section in the crack due to the lack of contrast among the pixels within the disjoint section.

As we are using a public crack segmentation dataset that combines images from 12 different datasets, Figure 8 displays the output produced by the proposed SC-CrackSeg using test samples from eight of these datasets. The name of the dataset is mentioned at the top of each RGB image. The second column in Figure 8 exhibits the actual annotation of the RGB image. It is observed that across all eight datasets, the proposed model fairly performs well. However, some wrong classifications can be observed due to the tiny shape of the crack.

We utilised several strategies to mitigate class imbalance, but surprisingly found these approaches reduced performance. We implemented a per-pixel class-weighted loss as in other crack segmentation literature [20], using the inverse of each class' pixel presence as the loss weight. This made the model significantly over-predict the crack class. While crack accuracy improved, it also introduced many false positives, and this trend was seen even as the class imbalance was made less extreme. We also implemented Focal Tsverky Loss [1] to address the class imbalance, but found this caused significant

boundary degeneration.

Alongside efficacy, we also focus on the model's efficiency as we target real-time crack segmentation using resource-constrained embedded devices. Table VI displays the various aspects of scene processing speed and efficiency for tested models. Among all listed models, DeepLab is the largest with 41 million (M) parameters and 78.8 GFLOPs. GFLOPs depend not only on model size but also on input resolution and the channel dimensions of feature maps. Hence, it can be observed that although UNet has 31 M parameters but 1740 GFLOPs. Similarly, SegNet and MR-CrackNet also have large GFLOP counts. Among all, SCMNet and the proposed model have the lowest number of parameters and GFLOPs. As we used `tensorflow` and `keras` to build all the models, hence we saved the checkpoints in `keras` model format. It is observed from the literature that while presenting the inference time, the model is optimized using the `TensorRT` engine and the corresponding FPS is presented. Instead of presenting directly the `TensorRT` optimized result, we report the FPS count of each type of model. We convert our `keras` model to `tensorflow` (TF), `TensorRT float-32` (TF-TRT32), `TensorRT float-16` (TF-TRT16) and `TensorRT integer-8` (TRT-INT8) and present the result in Table VI. It clearly demonstrates that the proposed final `TensorRT` optimized model (TRT-INT8) can count 220 frames per second at 448×448 resolution, followed by SCMNet (114 FPS). Among large models, DeepLab generates 82 FPS, followed by DeepCrack (65), SegNet (56), UNet (50), and MR-CrackNet (45). The training time column defines the time required for training at each epoch, though this also depends on the batch-size. Batch-size varies from model to model depending on their size. The proposed model

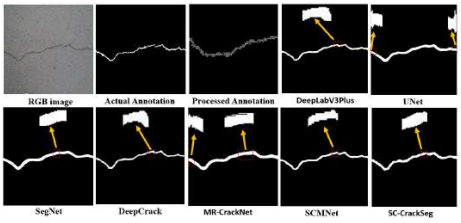

Fig. 7: Output by all using CFD crack dataset test sample

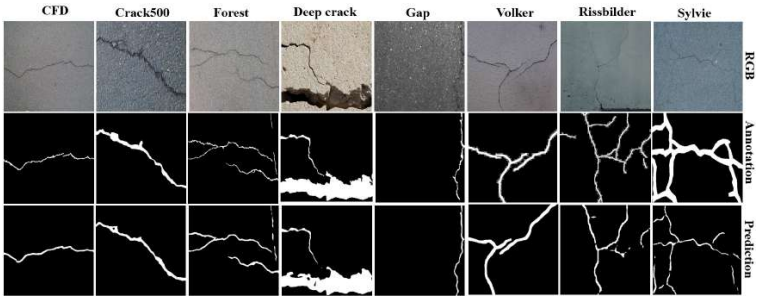

Fig. 8: SC-CrackSeg predictions for different datasets samples in the test set

TABLE VI: Efficiency analysis

			FPS of different types of model						
Model	Param.(M)	GFLOPs	Keras	TF	TF-TRT32	TF-TRT16	TRT-INT8	Training time(s)	Model size (MB)
DeepLab	41.0	78.8	20	21	32	48	82	415	158
Unet	31.0	1740.0	31	30	39	39	50	348	355
SegNet	29.4	245.0	32	32	44	45	56	316	225
DeepCrack	14.7	123.0	33	33	48	50	65	257	112
MR-CrackNet	17.7	331	14	14	25	28	45	1013	203
SCMNet	**1.2**	3.3	61	67	111	111	114	182	10.7
SC-CrackSeg	1.24	**2.8**	**70**	**78**	**216**	**216**	**220**	**93**	**10.8**

took 93 seconds to complete an epoch which is the lowest among all the models. From the model size column, it can be clearly seen that the size of the `keras` checkpoint is small for SCMNet and SC-CrackSeg. Hence, the data in Table VI distinctly illustrates the proposed model efficiency compared to the other models.

V. Conclusion

We have presented SC-CrackSeg, a new model developed specifically for crack detection and segmentation. It is based on a previous semantic segmentation model (SCMNet), but contains significant adaptations to suit the mobile monitoring of civil structures: we simplify the design from two to one input, the context mining module is simpler but still effective, and we also include a light-weight decoder with a new semantic aggregation module which makes the model more efficient. These new contributions have made SC-CrackSeg both light and high performing on challenging crack detection datasets. We have justified our choice of model design through a comprehensive ablation study and demonstrated its superior performance against other alternatives. Our implementation is available at our official GitHub repository: https://github.com/tanmaysingha/SC-CrackSeg.

VI. Acknowledgement

The authors would like to acknowledge Pawsey supercomputing centre for giving us the access of computing resources.

References

[1] N. Abraham and N. M. Khan, "A Novel Focal Tversky loss function with improved Attention U-Net for lesion segmentation," Oct. 2018.

[2] R. Ali, D. Kang, G. Suh, and Y.-J. Cha, "Real-time multiple damage mapping using autonomous UAV and deep faster region-based neural networks for GPS-denied structures," *Automation in Construction*, vol. 130, p. 103831, Oct. 2021.

[3] R. Amhaz, S. Chambon, J. Idier, and V. Baltazart, "Automatic crack detection on two-dimensional pavement images: An algorithm based on minimal path selection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 10, pp. 2718–2729, 2016.

[4] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE TPAMI*, vol. 39, no. 12, pp. 2481–2495, 2017.

[5] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE TPAMI*, vol. 40, no. 4, pp. 834–848, 2017.

[6] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.

[7] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proc. ECCV*, 2018, pp. 801–818.

[8] T. Chen, Z. Cai, X. Zhao, C. Chen, X. Liang, T. Zou, and P. Wang, "Pavement crack detection and recognition using the architecture of segNet," *Journal of Industrial Information Integration*, vol. 18, p. 100144, Jun. 2020.

[9] W. Choi and Y.-J. Cha, "SDDNet: Real-time Crack Segmentation," *IEEE Transactions on Industrial Electronics*, vol. PP, pp. 1–1, Oct. 2019.

[10] M. David Jenkins, T. A. Carr, M. I. Iglesias, T. Buggy, and G. Morison, "A Deep Convolutional Neural Network for Semantic Pixel-Wise Segmentation of Road and Pavement Surface Cracks," in *Proc. EUSIPCO*, Sep. 2018, pp. 2120–2124, iSSN: 2076-1465.

[11] M. Eisenbach, R. Stricker, D. Seichter, K. Amende, K. Debes, M. Sesselmann, D. Ebersbach, U. Stoeckert, and H.-M. Gross, "How to get pavement distress detection ready for deep learning? a systematic approach." in *Proc. IJCNN*, 2017, pp. 2039–2047.

[12] M. Everingham, S. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *IJCV*, vol. 111, no. 1, pp. 98–136, 2015.

[13] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch SGD: Training ImageNet in 1 hour," *arXiv preprint arXiv:1706.02677*, 2017.

[14] Y. Hamishebahar, H. Guan, S. So, and J. Jo, "A Comprehensive Review of Deep Learning-Based Crack Detection Approaches," *Applied Sciences*, vol. 12, p. 1374, Jan. 2022.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," Dec. 2015, arXiv:1512.03385 [cs].

[16] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[17] D. H. Kang and Y.-J. Cha, "Efficient attention-based deep encoder and decoder for automatic crack segmentation," *Structural Health Monitoring*, p. 14759217211053776, Dec. 2021.

[18] D. Kang and Y.-J. Cha, "Autonomous UAVs for Structural Health Monitoring Using Deep Learning and an Ultrasonic Beacon System with Geo-Tagging," *Computer-Aided Civil and Infrastructure Engineering*, vol. 33, no. 10, pp. 885–902, 2018.

[19] N. Kerle, F. Nex, M. Gerke, D. Duarte, and A. Vetrivel, "UAV-Based Structural Damage Mapping: A Review," *ISPRS International Journal of Geo-Information*, vol. 9, no. 1, p. 14, Jan. 2020.

[20] Y. Liu, J. Yao, X. Lu, R. Xie, and L. Li, "Deepcrack: A deep hierarchical feature learning architecture for crack segmentation," *Neurocomputing*, vol. 338, pp. 139–153, 2019.

[21] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. CVPR*, 2015, pp. 3431–3440.

[22] F. Nayyeri and J. Zhou, "Multi-Resolution ResNet for Road and Bridge Crack Detection," in *Proc. DICTA*. IEEE, 2021, pp. 1–8.

[23] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "Enet: A deep neural network architecture for real-time semantic segmentation," 2016.

[24] T. Pohlen, A. Hermans, M. Mathias, and B. Leibe, "Full-resolution residual networks for semantic segmentation in street scenes," in *Proc. CVPR*, 2017, pp. 4151–4160.

[25] R. P. Poudel, U. Bonde, S. Liwicki, and C. Zach, "ContextNet: Exploring context and detail for semantic segmentation in real-time," *arXiv preprint arXiv:1805.04554*, 2018.

[26] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Proc. MICCAI*. Springer, 2015, pp. 234–241.

[27] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proc. CVPR*, 2018, pp. 4510–4520.

[28] Y. Shi, L. Cui, Z. Qi, F. Meng, and Z. Chen, "Automatic road crack detection using random structured forests," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 12, pp. 3434–3445, 2016.

[29] T. Singha, M. Bergemann, D.-S. Pham, and A. Krishna, "SCMNet: Shared Context Mining Network for Real-time Semantic Segmentation," in *Proc. DICTA*. IEEE, 2021, pp. 1–8.

[30] T. Singha, D.-S. Pham, A. Krishna, and J. Dunstan, "Efficient segmentation pyramid network," in *Proc. ICONIP*. Springer, 2020, pp. 386–393.

[31] F. Yang, L. Zhang, S. Yu, D. Prokhorov, X. Mei, and H. Ling, "Feature pyramid and hierarchical boosting network for pavement crack detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 4, pp. 1525–1535, 2019.

[32] ——, "Feature pyramid and hierarchical boosting network for pavement crack detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 4, pp. 1525–1535, 2019.

[33] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "Bisenet: Bilateral segmentation network for real-time semantic segmentation," in *Proc. ECCV*, 2018, pp. 325–341.

[34] L. Zhang, F. Yang, Y. D. Zhang, and Y. J. Zhu, "Road crack detection using deep convolutional neural network," in *Proc. ICIP*. IEEE, 2016, pp. 3708–3712.

[35] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, "ICNet for real-time semantic segmentation on high-resolution images," in *Proc. ECCV*, 2018, pp. 405–420.

[36] Q. Zou, Y. Cao, Q. Li, Q. Mao, and S. Wang, "Cracktree: Automatic crack detection from pavement images," *Pattern Recognition Letters*, vol. 33, no. 3, pp. 227–238, 2012.

.7 Publication 7

2022 International Conference on Digital Image Computing: Techniques and Applications (DICTA) | 978-1-6654-5642-5/22/$31.00 ©2022 IEEE | DOI: 10.1109/DICTA56598.2022.10034634

SDBNet: Lightweight Real-time Semantic Segmentation Using Short-term Dense Bottleneck

Tanmay Singha, Duc-Son Pham, and Aneesh Krishna
School of Electrical Engineering, Computing and Mathematical Sciences
Curtin University, Perth, Western Australia
tanmay.singha@postgrad.curtin.edu.au, dspham@ieee.org, a.krishna@curtin.edu.au

***Abstract*—A popular choice when designing a semantic segmentation model is to adopt a pre-trained Deep Convolution Neural Network (DCNN) as a backbone and add extra modules for better semantic representation and competitive segmentation results. However, the large number of parameters and substantial memory footprint of these DCNN architectures make these large models unsuitable for real-time applications on mobile devices. To address the issue, this study proposes a very lightweight model, called Short-term Dense Bottleneck Network (SDBNet). By staging a series of bottleneck blocks, an efficient module, termed SDB, is carefully designed and it provides diverse field-of-views for better contextualization of varied geometrical objects in a complex scene. For precise localization, a shallow branch is deployed in parallel to SDB which shares the spatial details with the SDB module at multiple stages. At the decoder end, a simple, yet effective feature refinement and semantic aggregation module is deployed for better context assimilation and region identification. The proposed model is evaluated using three public benchmarks and the results on Cityscapes (70.8%), Camvid (73.2%) and KITTI (51.8%) test sets clearly demonstrate a competitive performance under the real-time category. Among the real-time scene parsing models under 1.5 million parameters, the proposed SDBNet produces the state-of-the-art (SOTA) results on all three datasets.**

***Index Terms*—semantic segmentation, encoder-decoder, bottleneck, feature aggregation, real-time models, DCNNs**

I. Introduction

Semantic segmentation is one of the most challenging tasks in computer vision. Essentially, it assigns a class to *each* pixel of an image, which is fundamental for higher level scene analysis. For various real-time applications, such as autonomous driving vehicles [3], medical image processing [11], indoor-outdoor scene analysis [21], robotics [22], human-computer interaction [42], semantic segmentation plays an important role.

Traditionally, an encoder-decoder architecture is used in scene parsing models. Basically, a succession of convolution layers are deployed at the encoder which filters an input image in the pyramidal layout format. It provides multi-scale field-of-views of the input and helps the model extracting meaningful information from the scene. For fusing the spatial and contextual details and completing the pixel labelling task, a decoder is deployed by combining a series of upsampling and convolution layers. Relying on this pyramidal architecture, a number of semantic models [4], [5], [16], [34], [40] have been proposed, all of which employ a popular large DCNN [27], [35] as the encoder. Such a large backbone usually leads to good segmentation performance. For instance, DeepLab [4] has shown around 4% gain in model performance when DCNN VGG-16 [27] is replaced by ResNet-101 [35]. Others, such as [13], [33], [38], also employ a large backbone for good performance, but they all have a large memory footprint.

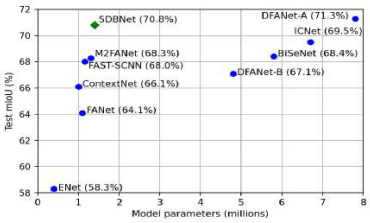

Fig. 1. Test mIoU vs Parameters (real-time model having less than 8M parameters

To reduce the computational cost, a multi-branch encoder-decoder design is introduced [15]. Instead of having a single pyramidal branch at the encoder, multiple branches are deployed for different input resolutions. Following this architecture, numerous models [23], [24], [28], [39] have achieved improved efficiency. On the other hand, increasing the depth of the multi-branch encoder [19], [36] improves accuracy but also increases computational cost.

We aim at achieving a much better trade-off between model performance and model efficiency by introducing a lightweight semantic segmentation model, called SDBNet. Our major contributions are:

- A new module, called Short-term Dense Bottleneck (SDB) which provides a better receptive field than existing bottleneck blocks.
- A careful design of the shallow branch to guide the deep branch effectively.
- A novel feature refinement and semantic aggregation modules at the decoder.
- A lightweight segmentation model, called SDBNet, with only 1.4 million parameters and a comprehensive evaluation using three public benchmarks. Among existing models having less than 1.5 million parameters, SDBNet

produces the best results (Refer Figure 1).

II. Related Work

In semantic segmentation, the encoder-decoder structure was first known with FCN-8S [16] which revolutionized the field by replacing the top Fully Connected (FC) layers with the convolution layers to obtain a spatial map. This architecture then inspired many others models, such as DeepLab [4], DeepLabV3+ [5], PSPNet [40], HANet [6], PAG [13], HRNetV2 [33], OCR [38]. These models designed different modules on top of the encoder to achieve improved performance. For instance, DeepLabV3+ [5], PSPNet [40] have shown the benefits of multi-scale feature scaling for extracting contextual details of various geometrical objects in the scene. They proposed Atrous Spatial Pyramid Pooling (ASPP) and Pyramid Pooling Module (PPM) for feature scaling. Models like HANet [6], PAG [13] introduce an attention mechanism which provide an additional information to guide the extraction of semantic details. Models that employ these modules generally achieve improved performance but they are not suitable for real-time applications as their backbone is large.

A more viable approach for real-time computation is the multi-branch encoder-decoder design [15]. Essentially, a model has one dedicated deep branch which takes low resolution input image and multiple shallow branches in parallel which accept comparatively higher resolutions input image. Models under this architecture such as ICNet [39], BisenetV1, [37], ContextNet [23], BiseNetV2 [36], SCMNet [28], SwiftNetV2 [19] have been proven to improve the efficiency during inference as compared to offline models. However, they still struggle during training due to pre-processing of multiple resolutions input.

In contrast to the multi-branch approach, models like ESPNet [31], FANet [29], M2FANet [30], FSFFNet [32] are introduced, based on a single-branch encoder. For better contextual representation, these models focus on multi-stage feature aggregation at the decoder which fuses spatial and contextual details at different stages and reduces the spatial gap among the feature maps. Most of these models have less than 1.5 million parameters and less than 70% test accuracy on Cityscapes [7].

Recently, DFANet [14] introduced a new design in which feature maps are re-used by multiple sub-encoders. This approach also exploits an FC attention module on top of each sub-encoder for providing better receptive fields. DFANet requires only one input of a fixed resolution and processes it through the first sub-encoder. The global feature map produced by the first sub-encoder is processed by the FC attention module and then upsampled by a large factor for further use by the next sub-encoder. Another new model, called STDC [9] is introduced very recently which removes the structure dependency of BiseNet [37] model and provides a dense feature aggregation architecture in single-stream manner. It uses ResNet [35] blocks and offers two different variants like DFANet, ranging parameters from 8.4 - 12.5M. Due to this large number of parameters, both of these models requires large memory space.

III. Proposed Method

Figure 2 displays the complete pipeline of the proposed model, named SDBNet. The following sub-sections describes each part of our proposed model.

A. Encoder Design

1) Down-Sampling: We use down-sampling in the first two stages of our proposed model. First, we deploy a 3×3 ConvX block which contains a standard convolution (Conv), a batch normalization (BN) and a ReLU layer. The Conv layer filters the input image by a 3×3 kernel with stride 2 and produces an output tensor of $512 \times 1024 \times 32$ for an input of $1024 \times 2048 \times 3$. Next, we deploy a DSConvX block which has a similar layer architecture except for the Conv one. Here, the standard Conv layer is replaced by a Depth-wise Separable Convolution (DSConv) layer. It also reduces the semantic dimension of the input by half and produces an output of $256 \times 512 \times 48$. Hence, the fist two stages down-sample the original input size to a quarter.

2) Deep Branch: The deep branch of the proposed encoder is designed by assembling a series of 6 SDB module of two types: SDB1 with stride 1 and SDB2 with stride 2 (see Figure 2). The Short-term Dense Bottleneck Residual blocks with stride 2 reduces the input size by 2. Figure 3(a) and 3(b) exhibit SDB1 and SDB2 respectively. We use 3 Bottleneck blocks (Bblocks) and one upsampling layer in each SDB module. The layered architecture of each Bblock, as shown in Figure 3(c), is motivated by the Mobile Bottleneck Convolution (MBConv) block of MobileNetV2 [25] except for the expansion ratio and residual connection. Whereas MobileNetV2 proposes MBConv1 and MBConv6 blocks with residual connection where expansion ratio is either 1 or 6 respectively, we use three different expansion rates (t): 2, 3 and 4 in our SDB modules. SDB2 accepts an input of size $H \times W \times C$ and the first Conv layer of a Bblock produces an output of size $H \times W \times 2 \times C'$. Here, H, W defines the spatial dimensions and C, C' symbolizes input and output channels respectively. As the first layer of a Bblock expands the channel of the output by t, we call it an expansion layer. Immediately after that, a Depth-wise Convolution (DwConv) layer is used to filter the input tensor along the depth. It also reduces the number of parameters more than the standard Conv one does. Finally, a point-wise Conv layer (projection layer) is used for further refinement of the feature map. Another purpose of deploying this layer is to assign the exact number of output channels C' to the output tensor. Thus, each Bblock expands the channel first, then filters the tensor along the expanded depth and later squeezes the channel by a projection layer. Hence, we achieve multiple receptive fields along the spatial and channel dimensions inside each Bblock. The first two Bblocks have a stride of 2 which means that the original input size of SDB2 will be reduced to one quarter. We deploy a upsampling layer with a pooling size of 2 in each SDB2,

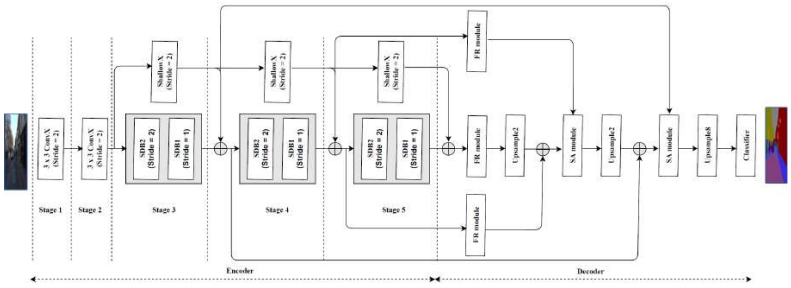

Fig. 2. The complete architecture of SDBNet

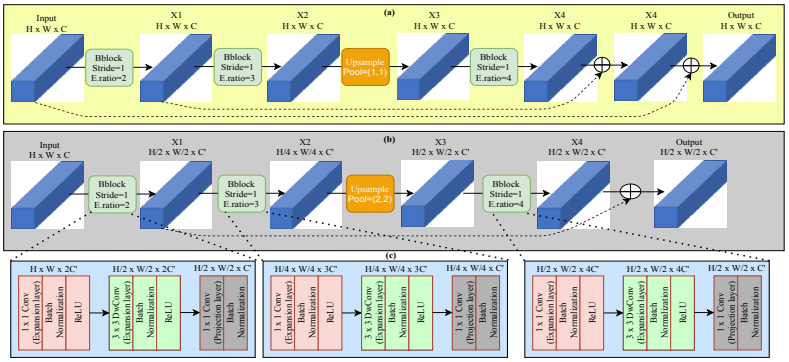

Fig. 3. The layered architecture of the SDB module

followed by a Bblock with a stride of 1. Hence, the feature map X4 ($H/2 \times W/2 \times C'$) in Figure 3 achieves similar dimensions of X1 ($H/2 \times W/2 \times C'$). Finally, one lateral connection is added between X1 and X4 to avoid any gradient vanishing issues.

SDB1 with a stride of 1 has a similar layered architecture like SDB2, except for one residual connection from the the end of the previous SDB2 module to the end of the current SDB1 module. If the SDB module in one stage is going to repeat more than once, then this residual connection will help the model retain all information and will forward it to the next stage. Figure 3(a) illustrates the complete architecture of SDB1. Here, we use an upsampling layer with pool size 1. Figure 2 shows that at each stage we deploy 2 SDB modules, in which each SDB2 is followed by SDB1. At each stage, the input feature map has three different spatial and channel dimensions based on the expansion ratio and stride respectively. Thus, our proposed design of SDB modules provides different receptive fields for capturing more contextual details of different object shapes in the scene.

3) Shallow Branch: Figure 2 also displays the shallow branch after the stage 2. It is parallel to the deep branch and contains only three shallow blocks (ShallowX), one in each stage. The layered architecture of ShallowX is simple and shallow: two DSConvX blocks are staged together with a 3×3 kernel at the main branch. The first DSConvX block has stride 2, followed by the second one with stride 1. This branch is designed to extract the boundary and texture details from the low level feature maps and pass the knowledge to the deep branch. Thus, ShallowX supplements the spatial details to the deep branch at three shared points of the encoder.

B. Decoder Design

The decoder is after the stage 5 in Figure 2. Here, the size of the global feature is $32 \times 64 \times 128$ for an input image of size $1024 \times 2048 \times 3$. Although the size of the feature map is small, but it contains rich contextual details which needs to be decoded by the model for generating the segmented output of similar input size. We mainly use three modules: Feature Refinement (FR_block), Semantic Aggregation (SA_block) and classifier.

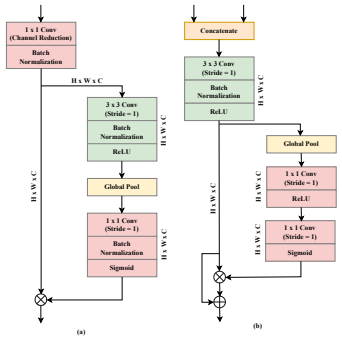

Fig. 4. Layered architecture of (a) FR module and, (b) SA module

TABLE I
LAYERED ARCHITECTURE AND PARAMETERS AND FLOPS OF EACH SECTION OF ENCODER

				Deep branch		Shallow branch			
Stage/Op	Input	C	S	Module	N	Module	N	Params.	FLOPs
DS1	1024×2048	32	2					1K	1.0G
DS2	512×1024	48	2					2K	0.5G
Stage3	256×512	64	2	SDB	2	SBlock	1	172K	11.6G
Stage4	128×256	96	2	SDB	2	SBlock	1	371K	6.3G
Stage5	64×128	128	2	SDB	2	SBlock	1	650K	2.8G
Total parameters and FLOPs of the encoder								1196K	22.2G

TABLE II
RESULT OF ABLATION STUDY

Encoder		Decoder			mIoU	Params.	FLOPs	FPS
DB	SB	FR	SA	Cl.	(%)	M	G	
✓				✓	65.3	1.2	6.0	142
✓	✓			✓	66.4	1.2	6.3	139
✓	✓	✓		✓	66.9	1.3	7.3	105
✓	✓	✓	✓	✓	67.5	1.4	11.5	80

1) Feature Refinement: The Feature Refinement (FR) module refines the semantic feature maps produced by the encoder at different levels. Figure 2 shows that deep features after stage 4 and 5 and shallow feature after stage 4 pass through the FR module. This module is similar to Attention Refinement Module (ARM) of BiseNet [37] except for the channel reduction layer. We introduce a Channel Reduction (CR) layer inside ARM. Moreover, a CR layer is also deployed to provide equal depth of each semantic feature. The layered architecture of the FR module can be seen in Figure 4(a). Features at stage 3, 4, and 5 are down sampled by $1/8, 1/16$, and $1/32$ times. Moreover, they also have different channel dimensions. Hence, at first feature gets uniform channel dimensions and then the noisy semantic details are filtered out through the FR module. After refining the 5^{th} stage deep global feature, it is scaled up a bi-linear upsampling layer and then, fused with the 4^{th} stage-refined feature map.

2) Semantic Aggregation: By exploiting the Feature Fusion Module (FFM) of [37], we design a Semantic Aggregation (SA) module to be used twice in the pipeline. Comparing with FFM, we replace the ConvX block after the concatenation by a DSConvX block. After channel reduction, features from stage 4 and 3 have 64 channels, but the spatial dimensions are upsampled by 2 and 4 times. Hence, to reduce the number of parameters, deploying DSConvX is necessary. The layered architecture of SA is shown in Figure 4(b). The symbols $\bigotimes$ and $\bigoplus$ denote element-wise multiplication and addition on a list of tensors of similar dimensions and return a single tensor. For semantic aggregation, we use refined deep and shallow features from the 4^{th} and 3^{rd} stages.

3) Classifier: We design the classifier with two DSConvX blocks, one standard point-wise Conv layer, one dropout and One softmax layer. Point-wise Conv assigns an equal number of channels similar to the number of classes of the dataset, dropout layer addresses over-fitting and softmax produces the class prediction based on the pixels' softmax scores.

IV. EXPERIMENT

A. Datasets

We use three popular urban street scene benchmarks: Cityscapes [7], KITTI [1], Camvid [2]. As Cityscapes and KITTI follow the same class labeling scheme, we train and evaluate the proposed model using 19 classes, whereas for Camvid, we use 11 classes. For Cityscapes, only fine annotated images are utilized for training. The fine-tune dataset is furnished with 2,975 training images, 500 validation images and 1525 test images. Although, it does not provide annotations for test set. Like Cityscapes, KITTI also provides annotations for 200 training samples only. Both data sets provide an online evaluation server to measure the model's performance on the test set. Our test results of Cityscapes and KITTI are independently obtained from it and are available on the servers. Camvid does not provide an evaluation server for test set. It has 267 images for training, 101 for validation and the remaining 233 for testing. Following the literature, we use only 11 classes out of 32 for the experiment. As KITTI and Camvid are comparatively smaller data sets, so model is pre-trained on Cityscapes.

B. Implementation Details

We train our model using a computer equipped with NVIDIA TITAN RTX5000 GPUs. The software includes CUDA 10.2 and `horovod 19.5` [26] for parallel processing. We select a batch size of 8 for all experiments. Following [4], [24], [40] a polynomial learning rate strategy with a base rate of 0.045 and a power of 0.9 is utilized. To avoid the gradient descent challenge due to the distributed `horovod` framework, we utilize the gradual warm-up strategy [10]. To achieve the optimum model loss, the stochastic gradient decent (SGD) optimizer is employed. We apply ℓ_2 regularization at a few top layers. A dropout layer with 0.35 rate is also deployed after the final convolution layer of the model. Several on-the-fly data augmentation techniques, such as random horizontal and vertical flip, adjusting brightness and contrast, random cropping, resizing, clipping by value, are also utilized.

TABLE III
PERFORMANCE ANALYSIS ON CITYSCAPES VALIDATION AND TEST SET

Type	Model	Parameters (Million)	FLOPs	Val. Class mIoU(%)	Val. Category mIoU(%)	Test Class mIoU(%)	Test Class iIoU(%)	Test Category mIoU(%)	Test Category iIoU(%)	FPS
	PAG [13]	-	-	-	-	-	75.7	-	55.8	-
	PSPNet [40]	250.8	412.2G	-	-	81.2	59.6	91.2	79.2	0.78
	OCR [38]	>76.0	>1087G	-	-	81.8	65.9	92.7	83.9	-
	HRNetV2 [33]	65.9	747.3G	81.1	-	81.6	61.2	92.2	82.1	-
	HANet [6]	65.4	2138.0G	80.3	-	80.9	58.6	91.2	79.5	-
	DeepLabV3+ [5]	43.0	1550.0G	-	-	82.1	62.4	92.0	81.9	0.25
Real time	SwiftNetV2 [19]	12.0	128G	72.2	-	75.9	-	-	-	41
	STDC1 [9]	8.4	2910	74.5	-	75.3	-	-	-	4
	DFANet [14]	7.8	3.4G	71.9	-	71.3	-	-	-	100
	ICNet [39]	6.7	28.3G	-	-	69.5	-	-	-	30.5
	BiseNet [37]	5.8	-	-	-	68.4	-	-	-	105
	BiseNetV2 [36]	5.2	21.2G	73.4	-	72.6	-	-	-	156
	FSFFNet [32]	1.3	50.8G	71.8	86.4	69.4	-	87.1	-	-
	M2FANet [30]	1.3	37.3G	68.5	85.3	68.3	38.7	86.9	67.8	9
	FAST-SCNN [24]	1.2	14.9G	68.6	-	68.0	37.9	84.7	63.5	124
	SCMNet [28]	1.2	38.3G	66.5	84.2	67.9	37.1	86.8	68.0	11
	FANet [29]	1.1	11.4G	65.9	83.6	64.1	33.2	83.1	61.1	22
	ContextNet [23]	1.0	37.5G	67.3	-	66.1	36.8	82.8	64.3	101
	ENet [20]	0.4	3.8G	-	-	58.3	34.4	80.4	64	21
Real time	SDBNet	1.4	42.3G	72.3	87.5	70.8	42.0	87.2	69.4	98

TABLE IV
CLASS-WISE PERFORMANCE ON CITYSCAPES TEST SET

Model	Road	S.walk	Build.	Wall	Fence	Pole	T.light	T.sign	Veg.	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	M.cycle	Bicycle	mIoU
ENet	96.3	74.2	85.0	32.2	33.2	43.5	34.1	44.0	88.6	61.4	90.6	65.5	38.4	90.6	36.9	50.5	48.1	38.8	55.4	58.3
ContextNet	97.6	79.2	88.8	43.8	42.9	37.9	52.0	58.9	90.0	66.9	91.9	72.2	53.9	91.7	54.0	66.5	58.4	48.9	61.1	66.1
F.SCNN	97.4	**81.6**	89.7	46.4	**48.6**	48.3	53.1	60.5	90.7	67.2	94.3	73.9	54.6	92.9	**57.4**	65.5	58.2	50.0	61.2	68.0
FANet	96.7	75.3	88.2	35.4	37.8	45.7	51.3	57.4	90.4	64.3	92.9	71.8	50.4	91.6	48.9	62.0	51.9	46.3	49.0	64.1
M2FANet	97.4	78.6	90.1	37.1	42.9	46.8	63.8	68.3	92.1	66.5	**94.5**	79.3	58.8	**93.6**	48.5	62.1	53.2	48.7	65.9	68.3
SCMNet	95.4	71.8	87.1	34.9	36.8	44.2	43.6	58.0	89.4	56.5	90.8	66.7	38.3	88.1	41.5	55.8	41.4	32.1	61.9	59.7
SDBNet	**97.9**	81.4	**91.0**	**47.4**	47.6	**55.2**	**64.5**	**69.7**	**92.0**	**68.5**	94.4	**79.4**	**59.1**	**93.6**	53.2	**70.4**	**62.9**	**51.6**	**66.1**	**70.8**

C. Ablation study

Table I displays the layered architecture of each stage at the encoder and corresponding parameters and FLOPs contribution by each section. The first two stages down-sample (DS) the input image by a quarter. C, S and N in Table I define the number of output channels, stride and number of repetition of each module/block. The proposed encoder has total 5 stages out of which last three stages contribute almost 86% of total model parameters. To keep the parameters and FLOPS count low, we deploy DSConv layers wherever required. The proposed encoder contributes 22.2G FLOPs at resolution 1024×2048.

We conduct an experiment using Camvid [2] to see the efficacy of each module of the complete pipeline (see Table II). At the initial stage, we deploy the Deep Branch, a basic decoder, and the classifier (Cl) model which produces 65.3% mIoU on Camvid's validation set while having only 1.2 million (M) parameters, 6.0G FLOPs and 142 FPS. By adding the Shallow Branch, our model achieves 1.1% increase in mIoU. Parameters and FLOPs counts are slightly increased which causes a small drop in FPS. It proves that the shallow branch provides spatial guidance to the deep branch by sharing the spatial details at three shared points. In the later stage, we employ Feature Refinement (FR) module at the decoder and 0.5% performance growth has been recorded. Finally, we deploy Semantic Aggregation (SA) module at two positions in the decoder which improves the performance by 0.6% at the cost of 1.4M parameters and 11.5G FLOPs. All the results are obtained at an input resolution of 640×896.

D. Model evaluation

To keep things self-contained, we do not pre-train our proposed backbone with ImageNet [8] like many existing models. We only perform transfer learning from Cityscapes to KITTI and Camvid as they are all about urban images.

1) Performance on Cityscapes: For completeness, Table III shows the performance of both off-line and real-time models. Apart from class and category IoU, the Cityscapes evaluation server also generates instance based IoU in both class and category-wise, which is included here. We also present validation set results, model parameters, FLOPs and FPS. The sign '-' in all the following tables indicates the missing results in either the literature or Cityscapes' public leaderboard.

TABLE V
FLOPS AND FPS AT DIFFERENT INPUT RESOLUTION

Resolution	384×1280		640×896		1024×2048	
Model	FLOPs	FPS	FLOPs	FPS	FLOPs	FPS
ENet	8.7G	51	10.2G	49	37.3G	9
ContextNet	2.3G	121	2.7G	114	9.9G	55
FAST-SCNN	3.5G	144	4.1G	136	14.9G	62
FANet	2.7G	86	2.7G	75	11.4G	22
M2FANet	9.0G	39	10.4G	37	37.3G	9
SCMNet	8.9G	59	9.9G	56	38.3G	11
DFANet	55.3G	45	64.4G	40	236G	8
BiseNetV2	31.1G	95	35.8G	77	133G	24
SDBNet	9.9G	100	11.6G	80	42.3G	26

TABLE VI
PERFORMANCE EVALUATION ON CAMVID TEST SET

Model	Pre-train	Input size	Test mIoU(%)
DeepLab [4]	ImageNet	720×960	61.6
PSPNet [40]	ImageNet	-	69.1
$VideoPro$ [41]	Cityscapes	720×960	81.7
ENet [20]	-	360×480	51.3
ICNet [39]	-	720×960	67.1
BiseNetV1 [37]	Cityscapes	720×960	65.6
BiseNetV2 [36]	Cityscapes	720×960	76.7
DFANet [14]	ImageNet	720×960	64.7
FAST-SCNN [24]	-	512×1024	57.5
FANet [29]	-	512×1024	57.8
M2FANet [30]	-	640×896	58.2
SCMNet [28]	Cityscapes	640×896	71.3
SwiftNetV2 [19]	ImageNet	448×448	73.7
SDBNet	Cityscapes	640×896	73.2

Among *offline* models, $DeepLabV3+$ [5] produces an excellent result with 82.1% test class mIoU, followed by OCR [38] (81.8%). Note that OCR is designed based on the backbone HRNetV2-W48 [33] and its backbone was excluded when measuring FLOPs and model parameters. Hence, we could not present the exact parameters and FLOPs of OCR. We note importantly that all these offline models are very large and have high computational cost.

Among *real-time* models, SwiftNetRN-18 [19] produces 75.9% test class accuracy, followed BiseNetV2 [36] (73.4%) and DFANet [14] (71.9%). However, all models are 3.5 to 9 times larger than the proposed model SDBNet. Our proposed model competitively accomplishes 73.2% and 74.5% mIoU on Cityscapes test and validation sets while having only 1.4M parameters and 42.3G FLOPs at resolution 1024×2048. Existing real-time scene parsing models such as ENet [20], ContextNet [23], FAST-SCNN [24], FANet [29], M2FANet [30], SCMNet [28] also have less than 1.5 million parameters, similar to the proposed model. Hence, we focus more on comparing the proposed model performance with all these lightweight models. Table IV exhibits the class-wise model performance on Cityscapes test set by the above listed models. It can be observed that the proposed model achieves the best result on 15 classes out of all 19 classes and also produces more than 90% class mIoU on road, building, vegetation, sky, and car classes.

Consistent with the literature, we also measure FPS to compare the efficiency. The last column of Table III shows the FPS count of all models: the proposed model achieves 98 FPS at input resolution 512×1024. To have a fair and meaningful comparison, we reproduce some existing models based on the literature and measured the FPS and FLOPs count at different input resolutions on the *same hardware and system configuration*. The results are depicted in Table V which clearly demonstrate that the FLOPs count rapidly increases and FPS count falls as the input resolution increases.

We note that the FPS and FLOPs counts of existing models reported in Table V may not match with what reported the literature as we suspect that other authors were not consistent with the way they measured these metrics. For instance, DFANet [14] claims 3.4 GFLOPs and 100 FPS at 1024×1024 resolution, whereas our reproduction of DFANet has 236 GFLOPs and 8 FPS at 1024×2048 input resolution with same number of parameters. Based on our experiment performed under the same hardware and system configuration, FAST-SCNN [24], ContextNet [23] are more efficient than the other existing real-time semantic models. However, these models have low efficacy. In comparison, the proposed model SDBNetachieve 100, 80 and 26 FPS at three different input resolutions and the FLOPs counts at the corresponding resolutions are also moderately small than DFANet [14] and SwiftNetV2 [19].

2) Performance on Camvid: As shown in Table VI, VideoPro [41] produces an outstanding results (81.7%) among *offline* models, possibly due to its use of a large synthetic dataset and specialised pre-processing techniques. Among *real-time* models, BiseNetV2 [36] produces the best test mIoU (76.7%), followed by SwiftNetV2 [19] (73.7%). However, SwiftNetV2 performs better than BiseNetV2 on Cityscapes. These models are 3.5 to 9 times larger than the proposed model, which competitively achieves 73.2% test mIoU and 80 FPS on Camvid. When compared to models having less than 1.5 million parameters, the proposed SDBNetindeed produces the SOTA results on Camvid test set as evident in Table VI.

3) Performance on KITTI: The result of KITTI test set obtained independently by its public evaluation server is displayed in Table VII. Although KITTI provides input images at 375×1280, we used an input size of 384×1280 due to tensor size compatibility when training. Note that we did not find any existing real-time semantic segmentation models trained and evaluated by KITTI. Models listed in Table VII are large and mainly designed for depth analysis. For instance, PAG [13], SDNet [18], SGDepth [12] have a semantic decoder along with the depth decoder. A semantic decoder is utilized for providing semantic guidance while estimating the depth. From the literature, we did not find model parameters, FLOPs count and FPS of all these models. Semantic head of PAG, SDNet and SGDepth (segmentation) generate 47.96%, 51.14% and 43.1% mIoU on KITTI test set. Model VideoPro [41] proposes a joint framework in which a video prediction model is deployed to generate a large synthetic training set for semantic training. Then it trains a deep existing semantic model with the synthetic dataset as well as Cityscapes [7] and Mapillary [17]. It also exploits a boundary label relaxation

TABLE VII
PERFORMANCE EVALUATION ON KITTI TEST SET

Type	Model	Pre-train	Class mIoU(%)	Class iIoU(%)	Cat. mIoU(%)	Cat. iIoU(%)	FPS	FLOPs
Off-line	SGDepth (Seg.) [12]	Cityscapes	43.1	-	-	-	-	-
	PAG [13]	City. fine-tune	47.96	17.86	78.11	49.17	-	-
	SDNet [18]	Cityscapes	51.14	17.74	79.62	50.45	-	-
	$VideoPro$ [41]	Mapillary, Cityscapes	72.8	48.7	88.9	75.3	-	-
Real-time	SDBNet	Cityscapes	51.8	18.7	78.0	44.5	100	9.9G

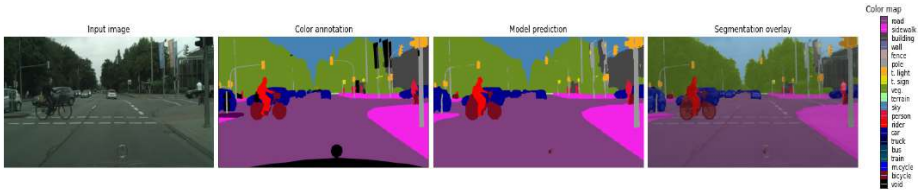

Fig. 5. Colour map and prediction using Cityscapes validation sample

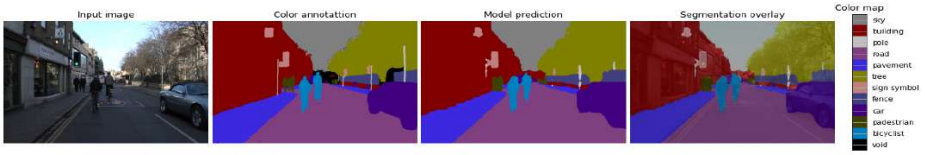

Fig. 6. Colour mapping and prediction using of Camvid validation sample

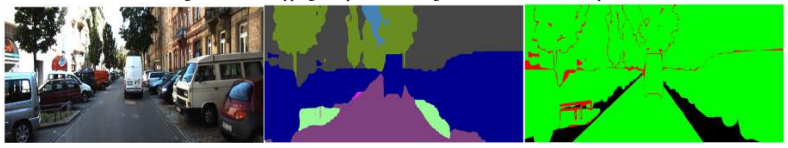

Fig. 7. Prediction using KITTI test sample

pre-processing technique. Thus, this joint strategy helps this model achieve an excellent semantic result (72.8%) on KITTI test set. In comparison, our proposed model generates 51.8% test class mIoU while having 9.9G FLOPs, 1.4M parameters and 100 FPS at 384×1280.

4) Qualitative results and analysis: Due to a lack of space, we only present few samples. We refer the reviewer to the supplemental material for more qualitative results.

Figures 5, 6, and 7 present the segmented output produced by the proposed model using all three datasets. Figure 5 also exhibits model prediction using Cityscapes validation samples (third image in Figure 5). For Camvid, 11 classes are used and the corresponding color codes are depicted in Figure 6. The third image in Figure 6 displays the prediction on Camvid validation set. Figure 7 displays model prediction generated by KITTI evaluation server using KITTI test samples. Along with the segmented output, server also provides a corresponding error image which visually unveils the correct and incorrect pixel labelling done by the model. It contains four colors: red defines wrong class and wrong category, yellow represents wrong class but correct category, green indicates correct class, and black means that the ground truth label is not used for evaluation. All these output clearly demonstrate the superior quality of prediction generated by the proposed SDBNet.

V. CONCLUSION

We have proposed SDBNet, a novel semantic segmentation model that achieves a best trade-off between efficacy and efficiency, making it very suitable for real-time applications on mobile devices. To achieve this, we design a very lightweight encoder by assembling SDB modules in the deep branch. The efficient design of SDB module provides better and wider field-of-view for extracting contextual details from the complex scene. Our lightweight shallow branch extracts low-

level semantics and provides a spatial guidance to deep branch for better semantic representation. It achieves the best performance among real-time models having less than 1.5 million parameters on three popular benchmarks. Our implementation is available at our official GitHub repository: https://github.com/tanmaysingha/SDBNet.

VI. Acknowledgement

The authors would like to acknowledge Pawsey supercomputing centre for giving us the access of computing resources.

References

[1] H. Alhaija, S. Mustikovela, L. Mescheder, A. Geiger, and C. Rother, "Augmented reality meets computer vision: Efficient data generation for urban driving scenes," *International Journal of Computer Vision (IJCV)*, 2018.

[2] G. J. Brostow, J. Fauqueur, and R. Cipolla, "Semantic object classes in video: A high-definition ground truth database," *Pattern Recognition Letters*, vol. 30, no. 2, pp. 88–97, 2009.

[3] X. Chang, H. Pan, W. Sun, and H. Gao, "Yoltrack: Multitask learning based real-time multiobject tracking and segmentation for autonomous vehicles," *IEEE TNNLS*, vol. 32, no. 12, pp. 5323–5333, 2021.

[4] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE TPAMI*, vol. 40, no. 4, pp. 834–848, 2017.

[5] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proc. ECCV*, 2018, pp. 801–818.

[6] S. Choi, J. T. Kim, and J. Choo, "Cars can't fly up in the sky: Improving urban-scene segmentation via height-driven attention networks," in *Proc. CVPR*, 2020, pp. 9373–9383.

[7] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. CVPR*, June 2016.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proc. CVPR*. Ieee, 2009, pp. 248–255.

[9] M. Fan, S. Lai, J. Huang, X. Wei, Z. Chai, J. Luo, and X. Wei, "Rethinking BiSeNet for real-time semantic segmentation," in *Proc. CVPR*, 2021, pp. 9716–9725.

[10] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch sgd: Training imagenet in 1 hour," *arXiv preprint arXiv:1706.02677*, 2017.

[11] T. M. Khan, A. Robles-Kelly, and S. S. Naqvi, "T-net: A resource-constrained tiny convolutional neural network for medical image segmentation," in *Proc. WACV*, 2022, pp. 644–653.

[12] M. Klingner, J.-A. Termöhlen, J. Mikolajczyk, and T. Fingscheidt, "Self-supervised monocular depth estimation: Solving the dynamic object problem by semantic guidance," in *Proc. ECCV*. Springer, 2020, pp. 582–600.

[13] S. Kong and C. Fowlkes, "Pixel-wise attentional gating for scene parsing," in *Proc. WACV*. IEEE, 2019, pp. 1024–1033.

[14] H. Li, P. Xiong, H. Fan, and J. Sun, "Dfanet: Deep feature aggregation for real-time semantic segmentation," in *Proc. CVPR*, 2019, pp. 9522–9531.

[15] G. Lin, A. Milan, C. Shen, and I. Reid, "Refinenet: Multi-path refinement networks for high-resolution semantic segmentation," in *Proc. CVPR*, 2017, pp. 1925–1934.

[16] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. CVPR*, 2015, pp. 3431–3440.

[17] G. Neuhold, T. Ollmann, S. Rota Bulo, and P. Kontschieder, "The mapillary vistas dataset for semantic understanding of street scenes," in *Proc. ICCV*, 2017, pp. 4990–4999.

[18] M. Ochs, A. Kretz, and R. Mester, "Sdnet: Semantically guided depth estimation network," in *German conference on pattern recognition*. Springer, 2019, pp. 288–302.

[19] M. Oršić and S. Šegvić, "Efficient semantic segmentation with pyramidal fusion," *Pattern Recognition*, vol. 110, p. 107611, 2021.

[20] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "Enet: A deep neural network architecture for real-time semantic segmentation," *arXiv preprint arXiv:1606.02147*, 2016.

[21] Q.-H. Pham, B.-S. Hua, T. Nguyen, and S.-K. Yeung, "Real-time progressive 3d semantic segmentation for indoor scenes," in *Proc. WACV*. IEEE, 2019, pp. 1089–1098.

[22] T. Pohlen, I. Badami, M. Mathias, and B. Leibe, "Semantic segmentation of modular furniture," in *Proc. WACV*. IEEE, 2016, pp. 1–9.

[23] R. P. Poudel, U. Bonde, S. Liwicki, and C. Zach, "Contextnet: Exploring context and detail for semantic segmentation in real-time," *arXiv preprint arXiv:1805.04554*, 2018.

[24] R. P. Poudel, S. Liwicki, and R. Cipolla, "Fast-scnn: fast semantic segmentation network," *arXiv preprint arXiv:1902.04502*, 2019.

[25] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proc. CVPR*, 2018, pp. 4510–4520.

[26] A. Sergeev and M. Del Balso, "Horovod: fast and easy distributed deep learning in tensorflow," *arXiv preprint arXiv:1802.05799*, 2018.

[27] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[28] T. Singha, M. Bergemann, D.-S. Pham, and A. Krishna, "SCMNet: Shared Context Mining Network for Real-time Semantic Segmentation," in *Proc. DICTA*. IEEE, 2021, pp. 1–8.

[29] T. Singha, D.-S. Pham, and A. Krishna, "Fanet: Feature aggregation network for semantic segmentation," in *Proc. DICTA*. IEEE, 2020, pp. 1–8.

[30] ——, "Urban street scene analysis using lightweight multi-level multi-path feature aggregation network," *Multiagent and Grid Systems*, vol. 17, no. 3, pp. 249–271, 2021.

[31] T. Singha, D.-S. Pham, A. Krishna, and J. Dunstan, "Efficient segmentation pyramid network," in *Proc. ICONIP*. Springer, 2020, pp. 386–393.

[32] T. Singha, D.-S. Pham, A. Krishna, and T. Gedeon, "A lightweight multi-scale feature fusion network for real-time semantic segmentation," in *Proc. ICONIP*. Springer, 2021, pp. 193–205.

[33] K. Sun, Y. Zhao, B. Jiang, T. Cheng, B. Xiao, D. Liu, Y. Mu, X. Wang, W. Liu, and J. Wang, "High-resolution representations for labeling pixels and regions," *arXiv preprint arXiv:1904.04514*, 2019.

[34] P. Wang, P. Chen, Y. Yuan, D. Liu, Z. Huang, X. Hou, and G. Cottrell, "Understanding convolution for semantic segmentation," in *Proc. WACV*. Ieee, 2018, pp. 1451–1460.

[35] Z. Wu, C. Shen, and A. Van Den Hengel, "Wider or deeper: Revisiting the resnet model for visual recognition," *Pattern Recognition*, vol. 90, pp. 119–133, 2019.

[36] C. Yu, C. Gao, J. Wang, G. Yu, C. Shen, and N. Sang, "Bisenet v2: Bilateral network with guided aggregation for real-time semantic segmentation," *International Journal of Computer Vision*, vol. 129, no. 11, pp. 3051–3068, 2021.

[37] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "Bisenet: Bilateral segmentation network for real-time semantic segmentation," in *Proc. ECCV*, 2018, pp. 325–341.

[38] Y. Yuan, X. Chen, and J. Wang, "Object-contextual representations for semantic segmentation," *arXiv preprint arXiv:1909.11065*, 2019.

[39] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, "Icnet for real-time semantic segmentation on high-resolution images," in *Proc. ECCV*, 2018, pp. 405–420.

[40] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proc. CVPR*, 2017, pp. 2881–2890.

[41] Y. Zhu, K. Sapra, F. A. Reda, K. J. Shih, S. Newsam, A. Tao, and B. Catanzaro, "Improving semantic segmentation via video propagation and label relaxation," in *Proc. CVPR*, 2019, pp. 8856–8865.

[42] H. Zuo, "Implementation of hci software interface based on image identification and segmentation algorithms," in *IC-GET*. IEEE, 2016, pp. 1–6.

.8 Publication 8

Pattern Recognition 140 (2023) 109557

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

A real-time semantic segmentation model using iteratively shared features in multiple sub-encoders

Tanmay Singha*, Duc-Son Pham, Aneesh Krishna

School of EECMS, Curtin University, Bentley, 6102, Western Australia, Australia

ARTICLE INFO

Article history:
Received 25 July 2022
Revised 20 February 2023
Accepted 24 March 2023
Available online 29 March 2023

Keywords:
Semantic segmentation
Deep convolution neural networks
Multi-encoder
Decoder
Feature scaling
Feature aggregation
Feature reuse
Resource-constrained applications
Mobile devices

ABSTRACT

Recent studies show a significant growth in semantic segmentation. However, many semantic segmentation models still have a large number of parameters, making them unsuitable for resource-constrained embedded devices. To address this issue, we propose an efficient Shared Feature Reuse Segmentation (SFRSeg) model containing several novelties: a new yet effective shared-branch multiple sub-encoders design, a context mining module and a semantic aggregating module for better context granularity. In particular, our shared-branch approach improves the entire feature hierarchy by sharing the spatial and context knowledge in both shallow and deep branches. After every shared point in each sub-encoder, a proposed cascading context mining (CCM) module is deployed to filter out the noisy spatial details from the feature maps and provides a diverse size of receptive fields for capturing the latent context between multi-scale geometric shapes in the scene. To overcome the gradient vanishing issue at the early stage, we reduce the number of layers in the first sub-encoder and employ a unique multiple sub-encoders design which reprocesses the rich global feature maps through multiple sub-encoders for better feature refinement. Later, the rich semantic features generated by the efficient sub-encoders at different levels are fused by the proposed Hybrid Path Attention Semantic Aggregation (HPA-SA) module that effectively reduces the semantic gap between feature maps at different levels and alleviate the well-known boundary degeneration effect. To make it computationally efficient for resource-constrained embedded devices, a series of lightweight methods such as a lightweight encoder, a squeeze-and-excitation design, separable convolution filters, channel reduction (CR) are carefully exploited. With an exceptional performance on Cityscapes (70.6% test mIoU) and CamVid (74.7% test mIoU) data sets, the proposed model is shown to be superior over existing light real-time semantic segmentation models whilst having only 1.6 million parameters.

1. Introduction

Due to the growing demand of building real-time applications for scene understanding in different industries such as in medicine, agriculture, civil engineering, video surveillance, robotics and car industries, an enormous amount of research work has been conducted in computer vision, especially in semantic segmentation field. It is a fundamental task for high-level scene understanding as it provides pixel-level recognition of each class of objects and background in a given image. This can be visualized in Fig. 1 where pixels belonging to the same class are clustered together and assigned a specific color code. Compared to object detection, semantic segmentation is known to provide better contextualization and more detailed information about the contents of the scene. However, it is usually more challenging and expensive than object detection and hence it is an open challenge to design semantic segmentation models that perform well in real time and have low computational cost [1] and are robust to different lighting conditions [2].

Many recent semantic segmentation models adopt an encoder-decoder architecture, with the encoder extracting contextual details from the scene and the decoder projecting the extracted semantic details back to the input space. Designing the encoder effectively is crucial in semantic segmentation. Several high-performing semantic segmentation models, such as DeepLab [5], PSPNet [6], PAG [7], HANet [8], and CENet [9], rely on popular large backbones, for example ResNet [10] and Xception [11], which have been proven effective for generic classification tasks. However, these backbone networks are often too large, making them unsuitable for real-time performance on resource-constrained applications such as mobile and edge devices due to their large memory footprint and high computational cost.

* Corresponding author.
E-mail address: tanmay.singha@postgrad.curtin.edu.au (T. Singha).

https://doi.org/10.1016/j.patcog.2023.109557
0031-3203/

Fig. 1. Pixel labelling in semantic segmentation. First column in the figure shows the RGB images taken from Cityscapes [3] and KITTI [4] data sets and the second column exhibits the corresponding colored ground truth in which pixels from same class are clustered together and received a specific color code based on the class.

By leveraging smaller variants of the ResNet architecture (ResNet-18/ResNet-29/ResNet-50), many semantic segmentation models such as SwiftNetV2 [1], and DFANet [12] have attempted to address the above issue. Nevertheless, they still have a significant number of parameters ranging from 7 million to 14 million. To further reduce the cost, some recent models, such as ShiftNet-MV2 [1], SCMNet [13], ContextNet [14], Fast-SCNN [15], FANet [16], and ESPNet [17] have employed a more efficient backbone architecture known as MobileNet [18], which offers outstanding generic classification performance on mobile devices.

All these existing semantic segmentation models mainly focus on improving contextual and spatial details of the scene by deploying different feature scaling techniques (Pyramid Pooling Module (PPM) [6], Atrous Spatial Pyramid Pooling (ASPP) [5]), different attention mechanisms (Height driven Attention (HA) [8], Fully Connected (FC) [12]), and different feature fusion techniques [15]. Feature scaling at different scales provides different sizes of the receptive field for filtering the scene and provides a better field of view. Feature aggregation combines shallow and global feature features for better semantic representation and the attention mechanism guides the feature learning process with high-level information. However, due to the complex nature of the scene, it is often visible that small objects having inconsistent geometric shapes are occluded by large objects or appeared as a part of the large objects. For instance, the presence of a tiny pole in front of a large building may be ignored by the model, although the texture and color of the pole and building are different. Such effect can be partially reduced by feature scaling. The features extracted by the encoder will have rich contextual details but also have low resolution. Therefore, they should be scaled at all levels differently for finer semantic granularity. This will segment the regions by different sizes of the receptive field to address objects of varying appearances in a scene.

To achieve real-time performance, both efficacy and efficiency need to be improved. A desirable semantic segmentation model should therefore be small and found in the top right corner of the performance-inference speed figure as shown in Fig. 2. Efficiency can be achieved by reducing the size of the backbone network, but this likely reduces efficacy. Similarly, handling higher-resolution input images can improve efficacy, but hampers model efficiency. To balance these objectives, the latest approaches, such as multi-branch encoder (ContextNet [14], ICNet [19]), customized lightweight encoders and decoders [12] have been proposed. Another recent approach is to achieve a smaller network through a process known as knowledge distillation [20]. However, the balance is still far from being satisfactory.

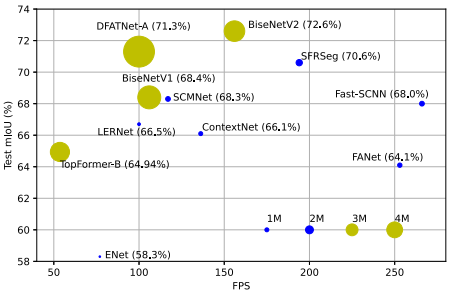

Fig. 2. The mIoU against the inference speed (FPS) for the Cityscapes test images (512×1024-pixel) using the proposed SFRSeg and the state-of-the-arts of real-time semantic segmentation models. The point size depicts the number of model parameters. Light models ($\leq$ 2M (Million)) and large models ($>$ 2M) are shown in Blue and Yellow, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

As the number of parameters affects how a model can be deployed in an application, the literature on semantic segmentation usually divides models into two groups: *offline* models having a very large number of parameters (hereinafter we define as more than 30 million based on the semantic segmentation literature) and *real-time* models having less number of parameters (less than 30 million). In this work, we further divide the real-time category into two sub-groups: *large* real-time models that still contain a large number of parameters (more than 2 million) and *light* real-time models that have less than 2 million parameters. Quite often offline models offer outstanding segmentation accuracy but at a huge computational cost whilst real-time models have lower performance with high efficiency. The open research question is how to design a real-time model effectively for reducing the performance gap. As this work specifically targets resource-constrained applications, such as mobile, embedded, and edge devices, we focus on light real-time models having a much smaller number of parameters, which is more suitable for these special applications. To address this gap, the study proposes a novel semantic segmen-

tation model, named SFRSeg, suitable for resource-constrained devices through the following contributions:

- A new yet effective lightweight shared multiple sub-encoders design for feature sharing and feature reusing across the backbone. This approach is inspired by the shared multi-branch encoder design [13] and provides better balance between efficiency and efficacy.
- A lightweight context mining module, called Cascading Context Mining (CCM) that helps the model segment regions with different receptive fields at different spatial dimensions of the input tensor. This addresses the issue of inconsistent visual appearances in the scene.
- A new Hybrid Path Attention Semantic Aggregation (HPA-SA) module for identifying the objects and localizing them accurately. Effectively, it reduces the semantic gap among the feature maps and produces more accurate semantic segmentation.

The paper is organized as follows. Section 2 surveys related work on different designs for semantic segmentation. Section 3 describes our proposed model in detail. We then present comprehensive experiments in Section 4 which justify the proposed design and demonstrate that our proposed model SFRSeg achieves state-of-the-art performance among the existing light semantic models having 2 million parameters or less on well-known benchmark data sets. Finally, concluding remarks are given in Section 5.

2. Related work

The literature has seen a number of approaches for designing the semantic segmentation models. Whilst early models are generally deep and computationally expensive to run [21], many recent models have been constructed with much less number of parameters and hence are more suitable for deployment on resource-constrained devices. We observed the following designs and techniques:

2.1. One-branch deep encoder design

An early model in this approach is FCN [21] which effectively converted a classification network to a segmentation one by replacing its last fully-connected layer with a convolution layer so that a spatial map is obtained. It also marked the introduction of the encoder-decoder architecture in semantic segmentation. The encoder may employ an existing deep CNN backbone such as Xception [11] and ResNet [22], to extract features whilst the decoder performs a series of upsample and convolution layers, followed by a softmax layer at the end. In this design, there is only a single path from the encoder to the decoder. As it is simple and effective, many models have been built following this general architecture, such as DeepLab [5], PSPNet [6], CENet [9], SegNet [10], and DenseASPP [23]. However, models that perform well, such as DeepLab, CENet always employs a very large backbone and hence they are computationally inefficient in many real-time applications, especially when the computational resources are limited. To address the drawback of these models, lightweight scene parsing models such as FANet [16], and ENet [24] have been proposed by designing a more efficient backbone. However, the complexity-performance trade-off has not been satisfactory.

2.2. Multi-branch encoder design

Whilst being simple, the one-branch design is generally inefficient as it requires a very large backbone network to resolve the semantic information across scales. For that reason, researchers soon realized a better approach to address this problem through a multi-branch design. This is particularly important for resource-constrained applications. The multi-branch design was first started by RefineNet [25], then followed by various models such as SwiftNetV2 [1], ContextNet [14], ICNet [19], and LERNet [26]. A multi-branch encoder design has one dedicated deep branch taking low-resolution input for contextual information and one or more shallow branches taking high-resolution input for local feature maps. By dealing with only low-resolution input in the deep branch, this design is more computationally efficient whilst retaining high performance through the extraction of local details in the shallow branch. However, there are two outstanding issues with this design: firstly, it needs to pre-process multiple copies of an input image for different branches independently which is not completely efficient; secondly, the branches often process data independently which does not promote the sharing of semantic knowledge across branches.

To address these issues, BiseNetV1 [27] introduced a two-branch encoder design that requires only a single input image and then two intermediate branches (deep and shallow) are created for deep and local feature maps. In this case, the efficiency during training and inference is further improved. However, both branches process the input tensor independently until the feature fusion stage. Following this two-branch approach with single-size input, other models such as Fast-SCNN [15], FANet [16], BiseNetV2 [28], STDC [29] were introduced.

Recently a new model, called SCMNet [13] has been developed based on a new approach called, shared-branch design. It has two separate branches at the encoder which takes inputs at two different resolutions. However, instead of having independent deep and shallow branches, SCMNet proposed a knowledge sharing approach between the deep and shallow branches. Such a shared design enhances the semantic and contextual details at the encoder side.

2.3. Multi-feature scaling technique

The encoder produces rich semantic feature maps at different scales. The global feature map at the end of encoder network has more contextual details despite having the lowest spatial dimensions. Extracting more surrounding details from the scene at this lowest resolution would be difficult due to the presence of varied objects of different geometrical shapes. Hence, several multi-scale feature scaling techniques such as ASPP [5], and PPM [6] were introduced to provide a better field of view at different scales. These scaling techniques have been utilized by several real-time semantic segmentation models such as ContextNet [14], Fast-SCNN [15], FANet [16], and DenseASPP [23],.

Similar to this feature scaling technique, some semantic models such as SCMNet [13], and BiseNet [28] introduced a feature refinement module which not only provides various sizes of the receptive field but also refines the feature maps at different levels of the encoder network.

2.4. Attention mechanism

The attention mechanism is a useful technique in computer vision, similar to feature scaling and feature refinement. Usually, an attention module is deployed on top of the encoder to better capture feature dependencies in the spatial and channel dimensions. Following this strategy, CDN [30] introduced two attention modules, called channel and spatial contextual modules, which exploits feature inter-dependencies in both dimensions for a better feature representation at the decoder side. Different from CDN [30], DFANet [12] exploited the attention module for enlarging the receptive field by adding one FC module on top of each sub-encoder. There are many existing deep models which utilize the attention mechanism as a guiding vector for better scene parsing. HANet

[8] introduced a height-driven attention vector which helps the feature maps learn more contextual details from the surrounding. GANet [31] proposed a Spatial Gated Attention (SGA) module for pixel-level attention information and highlighting the regions of interest for semantic pixel localization. Similarly, PAG [7] introduced a Pixel-wise Attentional Gating (PAG) unit that can be integrated with any existing semantic models to learn spatially varying pooling fields for enhancing model performance. In the literature [7], PAG is integrated with each residual block of ResNet.

2.5. Other techniques

There are some pre-processing or post-processing techniques which can be used for improving semantic segmentation performance. For instance, VideoPro [32] proposed a joint strategy in which a video prediction model is utilized to synthetically generate an additional training set for a semantic segmentation model. It also deployed a boundary label relaxation pre-processing task which makes training robust to label noise and propagation artifacts along object boundaries. Such pre-processing techniques improve the semantic performance of any existing scene parsing models. Similar, like boundary label relaxation, STDC [29] deployed a detail head along with the semantic head which generates a detailed ground truth (GT) of object boundaries from the actual semantic segmentation GT. This detailed boundary GT is used as a guidance vector for spatial details during the training. It is discarded in the inference phase.

Other models such as SDNet [33], SGDepth [34] followed the one branch encoder design. These models are mainly designed for stereo and depth analysis. However, they have multiple decoders: one decoder is used for depth analysis and the other one is used for semantic segmentation. Each decoder head complements each other. More recently, TopFormer [35] has been specifically designed for mobile devices, significantly reducing the computational burden of vision transformer-based approaches whilst achieving a good trade-off between accuracy and latency.

3. Proposed method

The previous study [13] has shown the effectiveness of the shared branch design over the independent multi-branch design for semantic segmentation. Inspired by the knowledge sharing approach among the shallow and deep branches, we adopt the shared branch technique for designing the proposed encoder. For better coarse-to-fine refinement, we also use a new cascading multi sub-encoders architecture with successively decreasing stages but increasing repetition of convolution blocks in top stages. Here, a stage defines a level where the size of the input tensor is down-sampled by 2. Hence, the rich semantic feature maps at different levels from the first sub-encoder are processed multiple times by the successive sub-encoders for better contextualization. The detailed description of each part of the proposed network architecture is given below.

3.1. Shared-branch approach

We design our first sub-encoder, in which the deep and shallow branches share their extracted feature maps at three points. Unlike SCMNet [13], our model only requires one high-resolution input image and creates both deep and shallow branches at the second stage using down-sampling technique. This reduces the training complexity and makes the model more efficient for real-time applications. Table 1 illustrates the layered architecture of the down-sampling module, which consists of a standard convolution (Conv) layer followed by a depth-wise separable convolution (DSConv) layer. After down-sampling, two separate branches (deep and shallow) are created. The shallow branch extracts spatial details, and the deep branch extracts contextual details from the input tensor. The layered architecture of the shallow branch of the first sub-encoder is shown in Table 1, and the complete architecture of the deep branch in each such-encoder is displayed in Table 2. The successive sub-encoders do not have separate shallow branches; they retrieve shallow features at different levels from the first-sub-encoder's shallow branch.

Table 1
Layered architecture of the down-sampling module and shallow branch.

Down sampling module			
Stage	Input	Operator	Output
1	$1024 \times 2048 \times 3$	3×3 Conv	$512 \times 1024 \times 24$
2	$512 \times 1024 \times 24$	3×3 DSConv	$256 \times 512 \times 32$
Shallow branch			
3	$256 \times 512 \times 32$	3×3 DSConv	$128 \times 256 \times 48$
Pass through KSB at 3rd stage			
4	$128 \times 256 \times 48$	3×3 DSConv	$64 \times 128 \times 80$
Pass through KSB at 4th stage			
5	$64 \times 128 \times 80$	3×3 DSConv	$32 \times 64 \times 96$
Pass through KSB at 5th stage			

Each branch of the sub-encoders is designed for the use in mobile devices with low computational power. In the shallow branch, three DSConv layers are assembled together to extract boundary details from the input at three different levels. In contrast to the standard Conv layer, DSConv first convolves a three dimension (3D) input tensor with a two dimension (2D) filter along each channel of the tensor. Later, the output of each channel is stacked together to get the final output of the 3D input tensor. The advantage of employing DSConv instead of Conv is best shown in terms of the floating point operations (FLOPs) and parameters as follows:

$$\begin{aligned}\text{FLOPs}_{\text{Conv}} &= [(K_w \times K_h \times C_i) + (K_w \times K_h \times C_i - 1) + 1] \\ &\quad \times C_o \times H_o \times W_o = 2 \times K \times K \times C_i \times H_o \times W_o \times C_o, \quad (1)\end{aligned}$$

$$\begin{aligned}\text{FLOPs}_{\text{DSConv}} &= 2 \times K \times K \times C_i \times H_o \times W_o \times 1 \\ &\quad + 2 \times 1 \times 1 \times C_i \times H_o \times W_o \times C_o, \quad (2)\end{aligned}$$

$$\text{params}_{\text{Conv}} = ((K \times K \times C_i) + 1) \times C_o, \quad (3)$$

$$\text{params}_{\text{DSConv}} = ((K \times K \times C_i) + 1) \times 1 + ((1 \times 1 \times C_i) + 1) \times C_o. \quad (4)$$

In Eq. (1), $(K_w \times K_h \times C_i)$ and $(K_w \times K_h \times C_i - 1)$ represent the amount of multiplication and addition operations performed by a standard Conv layer, where K_w, K_h define the kernel size, C_i, C_o define number of filters of input and output tensors, and H_o, W_o represent the spatial dimensions of the output tensor. If a square filter is used for convolution, then K_h and K_w are replaced by K. By comparing FLOPs and the number of parameters, it can be seen that DSConv reduces the complexity by approximately K^2 times. After every DSConv layer in the shallow branch of first sub-encoder, the feature map passes through a Knowledge Sharing Block (KSB). It consists of a simple addition layer followed by a Cascading Context Mining (CCM) module which refines the added feature for better semantic representation. Their details are given subsequently.

3.2. Deep branch

The design of our proposed deep branch is different from SCMNet [13]. Due to our cascading multiple sub-encoders design, uti-

Table 2
Layered architecture of the deep branch. Here, KSB defines Knowledge Sharing Block and FRB defines Feature Reuse Block.

Stage	Input	Operators	Width mul. (M_w)	Depth mul. (M_d)	stride	Dilation rate (r)	Output
Layered architecture of the deep branch of sub-encoder1							
3	$256 \times 512 \times 32$	MBC1,k3 × 3	1.0	1	2	1	$128 \times 256 \times 32$
-	$128 \times 256 \times 32$	MBC6,k3 × 3	1.5	2	1	2	$128 \times 256 \times 48$
Pass through KSB at 3rd stage							
4	$128 \times 256 \times 48$	MBC6,k3 × 3	1.33	2	2	1	$64 \times 128 \times 64$
-	$64 \times 128 \times 64$	MBC6,k3 × 3	1.25	1	1	1	$64 \times 128 \times 80$
Pass through KSB at 4th stage							
5	$64 \times 128 \times 80$	MBC6,k3 × 3	1.2	1	2	1	$32 \times 64 \times 96$
Pass through KSB at 5th stage							
6	$32 \times 64 \times 96$	MBC6,k3 × 3	1.33	1	2	1	$16 \times 32 \times 128$
Upsample it by 8 times and pass through FRB							
Layered architecture of the deep branch of sub-encoder2							
Pass through KSB at 3rd stage							
4	$128 \times 256 \times 48$	MBC6,k3 × 3	1.33	3	2	1	$64 \times 128 \times 64$
-	$64 \times 128 \times 64$	MBC6,k3 × 3	1.25	1	1	1	$64 \times 128 \times 80$
Pass through KSB at 4th stage							
5	$64 \times 128 \times 80$	MBC6,k3 × 3	1.2	1	2	1	$32 \times 64 \times 96$
Pass through KSB at 5th stage							
6	$32 \times 64 \times 96$	MBC6,k3 × 3	1.33	1	2	1	$16 \times 32 \times 128$
Upsample it by 4 times and pass through FRB							
Layer architecture of the deep branch of sub-encoder3							
Pass through KSB at 4th stage							
5	$64 \times 128 \times 80$	MBC6,k3 × 3	1.2	2	2	1	$32 \times 64 \times 96$
Pass through KSB at 5th stage							
6	$32 \times 64 \times 96$	MBC6,k3 × 3	1.33	1	2	1	$16 \times 32 \times 128$
Upsample it by 2 times and pass through FRB							
Layered architecture of the deep branch of sub-encoder4							
Pass through KSB at 5th stage							
6	$32 \times 64 \times 96$	MBC6,k3 × 3	1.33	2	2	1	$16 \times 32 \times 128$

Table 3
Bottleneck residual block.

Input	Operator	Output
$h \times w \times c$	1 × 1 Conv,1/1, Relu	$h \times w \times tc$
$h \times w \times tc$	3 × 3 DwConv, 3/s, Relu	$h/s \times w/s \times tc$
$h/s \times w/s \times tc$	1 × 1 Conv,1/1, -	$h/s \times w/s \times c'$

lizing the SCMNet backbone would significantly increase the number of parameters and FLOPs. For instance, our proposed deep branch of the first sub-encoder has only 0.44 million (M) parameters and 1.99G FLOPs, while SCMNet has 1.06 M parameters and 2.96G FLOPs at a resolution of 640 × 896 px. Therefore, we propose here a very lightweight deep branch at the first sub-encoder. However, extracting rich contextual details by using such a lightweight deep branch is difficult. As a result, we propose a cascading multiple sub-encoders design where subsequent sub-encoders reprocess features from the first sub-encoder. The detailed layered architecture of the deep-branches of all sub-encoders is shown in Table 2.

A fundamental difference between our model and many others is that we design the backbone from scratch. Each sub-encoder is designed by stacking a series of MobileNetV2 [18] bottleneck blocks (MBConv). Studies have shown that MBConv blocks are more efficient for mobile devices compared to other existing residual blocks [11,22]. Due to the optimized bottleneck architecture and the effective utilization of Depth-wise Convolution (DwConv) layer at the expansion stage, it produces fewer parameters and consumes less memory. Table 3 shows the layered architecture of MBConv block of MobileNetV2 [18] with an expansion factor t. For an input feature map F_i with the shape of $h \times w \times c$, an MBConv block of MobilenetV2 first expands the feature map along the channel and produces an output of $h \times w \times tc$. In the next layer, an MBConv block filters the feature map along the channel dimension by a 3×3 filter. Using a DwConv layer significantly reduces the computational cost compared to a Conv layer. Thus, we control the computational cost of the proposed backbone and make the model more efficient in real-time environments, by incorporating MBConv blocks in the deep branch of every sub-encoder. MobileNetV2 provides two different types of bottleneck blocks: MBConv1 and MBConv6. The suffixes 1 and 6 define the expansion ratio t at the intermediate stage of the bottleneck block. The depth multiplier (M_d) in Table 2 specifies the number of repetition of MBConv blocks of varying channels in the deep branch of each sub-encoder. The proposed model employs 8, 6, 3, and 2 MBConv blocks in sub-encoder1, sub-encoder2, sub-encoder3 and sub-encoder4 respectively. The width multiplier (M_w) in Table 2 regulates the number of channels of the output feature map (F_o) of each MBConv block. The range of the width multiplier is from 1.0 to 1.5 and the final feature map has maximum 128 channels, which is known to be sufficient for containing color information of each pixel [12]. We have also attempted to increase the number of channels to 160 and could not achieve any further improvement whilst, incurring additional cost. The stride in Table 2 defines the shift amount when filtering an input with a kernel. Thus, it reduces the size of the input feature map by 2 and creates 6 stages in the first sub-encoder. We use a dilation rate of 2 wherever the stride becomes 1 to provide a larger receptive field.

Motivated by [12], this work reuses the final global feature of each sub-encoder in the next sub-encoder. For instance, the global feature of sub-encoder 1 is upsampled by 8 times before it is fed as an input to sub-encoder 2. Likewise, the global features of sub-encoder 2 and sub-encoder 3 are upsampled by 4 and 2

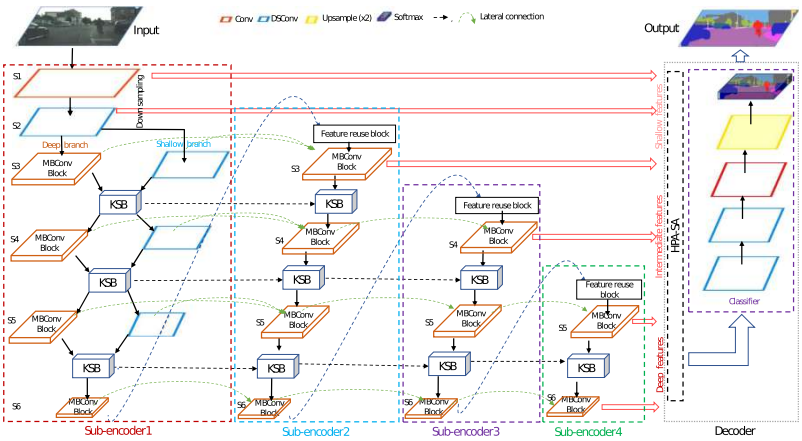

Fig. 3. Multiple sub-encoder design. Blue dotted arrows denote upsampling between sub-encoders. Green dotted arrows denote lateral connections between equivalent layers in different sub-encoders. Black dashed arrows denote lateral connections between equivalent Knowledge Sharing Blocks (KSB) in different sub-encoders. Best viewed in color. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

times respectively before they get re-filtered by the subsequent sub-encoders. The blue dotted arrows in Fig. 3 shows the process of upsampling from one sub-encoder to successive sub-encoder. After upsampling the global feature map, it passes through Feature Reuse Block (FRB). Detail about FRB is discussed later. Reusing global feature map is known to extract more contextual details of every pixel. The literature has shown that the global feature map contains more surrounding information of each pixel than the local one. Therefore, filtering deep features repeatedly through the multiple sub-encoders will extract more context from the scene compared to a sub-encoder. In the proposed multiple sub-encoders design, deep features at the 6th, 5th and 4th stages are re-filtered by 4, 4, and 3 times respectively. Deeper features are re-processed more number of times than the intermediate and shallow features. Fig. 3 demonstrates such feature reuse. The horizontal green dotted arrows exhibit the lateral connections among the feature maps of different sub-encoders at the same level. As each KSB block takes two input(one shallow and one deep feature maps), a lateral connection (black dotted arrow) can be seen in Fig. 3 from the previous sub-encoder to next sub-encoder among the KSB modules at same level.

Fig. 3 displays that, compared to the first sub-encoder, the number of total stages in the successive sub-encoders is reduced, although the number of MBConv blocks in the deeper stages of successive sub-encoders are increased. For instance, in sub-encoder 2, features at the 4th, 5th and 6th stages from sub-encoder1 are reused and the number of MBConv blocks at the 4th stage is increased. Similarly, features at the 5th and 6th stages are reused by sub-encoder3 and the number of MBConv blocks at the 5th stage is increased. Thus, we make the deep branch in the successive sub-encoders slightly deeper mainly for the global feature maps.

3.3. Knowledge sharing block

Tables 1, 2 show that after the 3rd, 4th and 5th stages, shallow and deep features pass through the Knowledge Sharing Block (KSB), which shares more semantic details among the branches. The shallow branch is designed to extract texture details from the input image whereas the deep branch is exploited for mining more contextual details from the scene. However, both branches have the coarsest details initially that need to be refined properly at the early stages. Therefore, to filer out the noisy details and for the accurate object localization, KSB is deployed after every shared point. It contains a simple addition operation followed by a CCM module. The addition operation combines the knowledge from the shallow and deep branches, whereas CCM filters the fused feature map through its cascading design and produce a refined output (see Fig. 4(c)).

Cascading Context Mining (CCM) module is developed to better refine the feature map through its own cascading multiple branches. Inspired by ASPP [5] and DenseASPP [23], we first review them and then explain key architecture differences in our design of CCM. As shown in Fig. 4, ASPP uses 5 parallel branches (one image pooling and four atrous Conv branches with higher dilation rates) to scale the same input with different field of views, whereas DenseASPP [23] uses a dense combination of sequential and parallel atrous Conv branches with five different dilation rates. The dense cascading design, which is beneficial for enlarging the receptive fields, does improve the performance. However, it is computationally demanding: a single DenseASPP module generates 10.6 M parameters with 128 input and output channels, which is prohibitive for resource-constrained applications. To specifically address this limitation, we make the following changes when designing CCM: 1) we deploy 4 branches instead of 5; 2) we utilize one point-wise Conv and three dilated DSConv branches with higher dilation rates instead of image pooling and Conv branches; 3) we use a single skip connection from the output of point-wise Conv branch to the input of second and third DSConv branches instead of using dense cascaded connections; 4) we use addition operations at the third and forth branch instead of concatenation; and 5) we establish a lateral connection between the initial input to final concatenated output to avoid the gradient vanishing problem. All these modifications reduce the number of parameters and GFLOPs count of CCM. The proposed module has only 8128 parameters, which is 22 and 130 times smaller than ASPP and DenseASPP, respectively.

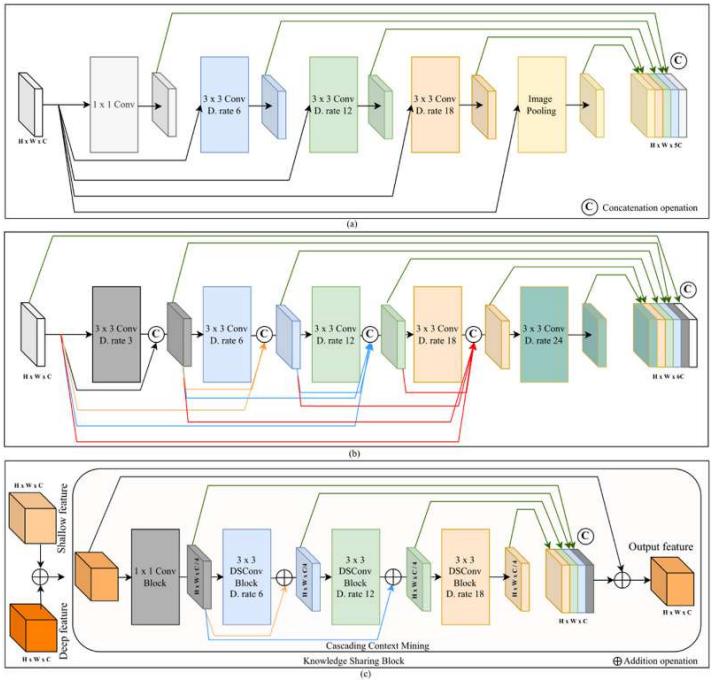

Fig. 4. ASPP vs DenseASPP vs CCM. Different colors represent different branches.

Fig. 4 illustrates the design of ASPP, DenseASPP and the proposed CCM module, clearly showing the differences between these three modules. Each branch is defined by a different color in Fig. 4. In ASPP, all four branches receive an input from the same source (black arrows), and output of each parallel branch is concatenated at the end (green arrows). In contrast, DenseASPP and CCM have branches that are connected sequentially, with each branch receiving input from the previous branch. However, in DenseASPP, there are dense lateral connections from the input to the output of all the branches (displayed by different colored arrows in Fig. 4(b)) for feature concatenation. In CCM, only two lateral connections exist, from the output of the first branch to the output of the second and third DSConv branch (yellow and blue arrows in Fig. 4(c)). The first branch of CCM is a fine-tuned branch which uses a standard 1×1 Conv layer for feature refinement. To lower the computational cost, it reduces the number of channels of the input feature map by 4 times. In the DSConv branches, we use (6, 12, 18) dilation rates respectively. Due to the cascaded design, instead of receiving the same input, each DSConv branch obtains the input from the predecessor branch fused with the fine-tuned branch output by addition operation. Thus, the shared feature gets more refinement and passes to the next stage of the sub-encoder.

3.4. Feature reuse block

Deploying the feature reuse block has two main reasons. First, excessive upsampling of the global feature map by 2^n times where $n = 3, 2, 1$ causes boundary degeneration effect. Second, channel reduction (CR) is required after upsampling to provide the same number of channels so that features from the same stages of the previous sub-encoder can be fused together. Fig. 5 displays the layered architecture of FRB. Here, the global feature of the first sub-encoder is up-scaled by 2^3 times to produce the similar height and width of the third stage feature map. A bi-linear interpolation method is used for up-scaling the global feature map. Later, a Depth-wise convolution (DwConv), a batch normalization and a ReLU layer are deployed to reduce the boundary degeneration effect. To provide the exact channel dimension of the feature map in the third stage, we deploy a point-wise Conv layer with 48 filters. Thus, it makes the global feature map of the sub-encoder1 reusable by the sub-encoder2. Once, the feature map is ready for use by the next sub-encoder, then features from the same stages of previous sub-encoder are fused with it through lateral connections. Such lateral connections reduce the gradient vanishing issue and eliminate the need for shallow branch in the successive sub-encoders. After each residual correction, we deploy KSB for better feature representation.

3.5. Decoder

The proposed decoder is divided in two parts: a Hybrid Path Attention Semantic Aggregation (HPA-SA) module and a classifier. The first part takes all the feature maps from the different levels of

Fig. 5. Feature Reuse Block. BN means batch normalization.

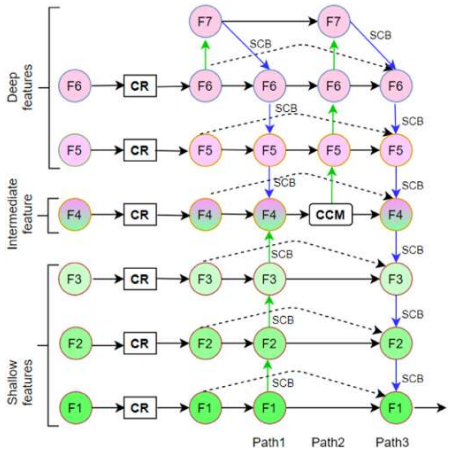

Fig. 6. Decoder design. Up-pointing arrows mean upsampling. Down-pointing arrows mean downsampling. Dashed arrows mean lateral connections. CR means channel reduction. CCM means cascade context mining module. SCB means separable convolution block. Best viewed in color.

the encoder and fuses them together, and the second part assigns a class label to each pixel based on the softmax score.

3.5.1. Hybrid path attention-semantic aggregation

The complete architecture of HPA-SA is displayed in Fig. 6. The encoder extracts rich features at different levels, and we use the entire feature hierarchy from the encoder to allow the decoder to reconstruct the segmented output based on it. Features at different stages have different spatial and channel dimensions. We deploy channel reduction (CR) to provide an equal depth to all the feature maps from the different stages. A CR block contains one 1×1 Conv layer with 64 filters having a stride of 1. After CR, we create an additional stage feature map F_7 by simply down sampling the F_6 feature map by the max pooling operation. This additional level feature map loses texture details due to the low resolution, but it contains more details about the background of the scene. It also adds another feature hierarchy for multi-scale feature scaling. For instance, the proposed encoder produces a global feature with the shape of $16 \times 32 \times 128$ for an input image of $1024 \times 2048 \times 3$. The CR block of HPA-SA transforms it to a $16 \times 32 \times 64$ feature map. Later, it creates a F_7 feature map with the shape of $8 \times 16 \times 64$. Thus, it creates three deep (F_7, F_6, F_5), one intermediate (F_4) and two shallow (F_3, F_2, F_1) feature maps for semantic aggregation through a hybrid path attention mechanism. The first path in HPA-SA is a mixed directional path, in which the top features are bi-linearly upsampled to acquire the same spatial dimension of the previous level feature map through a top-down approach, and the bottom feature maps are down-sampled to provide the same size of the next level feature map through a bottom-up approach. The top-down signal propagation helps fuse global features with the local one. On the other hand, the bottom-up signal flow enhances the object localization by blending local spatial details with rich semantic feature maps. Thus, this hybrid approach in the first path blends the top-down and bottom-up signals with the intermediate feature map F_4 at the $4th$ level. After fusing the features at the $4th$ stage, the output is refined by our CCM for better semantic representations in the next successive paths. The down-sampling process continues through path2 till the top stage ($7th$) for better region identification at different feature hierarchy levels. Lastly, a top-down path (path3) is introduced for final assimilation of contextual details. As the signal passes through multiple paths, the model may encounters the gradient vanishing problem. To address this, lateral connections are used, which are illustrated by the dotted lines in Fig. 6. The top-down movement is displayed by the blue arrow and bottom-up direction is shown by the green arrow. After every upsample or down-sample operation, the features pass through a Separable Convolution Block (SCB) that prevents boundary degeneration effects due to scaling the feature map up or down. It contains one DSConv layer followed by a BN layer. Thus, HPA-SA produces the finest semantic feature map by aggregating multi-levels features through the hybrid path attention mechanism.

3.5.2. Classifier

The classifier module is shown in Fig. 3. It takes the output from HPA-SA, which is a feature map with the shape of $H/2 \times W/2 \times 64$ where H and W define the height and width of the original input image. Classifier filters it through a sequentially connected two DSConv layers and one point-wise Conv layer. Inspired by the literature [15], DSConv layers are utilized for better feature refinement and a point-wise Conv layer is deployed for assigning C_c channels to the output feature map, where C_c characterizes the number of classes of the data set. Finally, one upsample and one softmax layer are deployed on top of the decoder. Softmax is used as an activation function which assigns a label or class to each pixel based on their weighted sum values. Thus, the segmented output is generated by the proposed SFRSeg.

4. Experiments

The proposed model SFRSeg is evaluated on four data sets, including three urban street scenes data sets (Cityscapes [3], KITTI [36] and CamVid [37]), and one indoor object scenes dataset [38]. We strictly follow the evaluation protocols of these data sets and use the following performance metrics: class and category mean Intersection over Union (mIoU), instance Intersection over Union for each category (iIoU), the number of parameters, Floating Point Operations (FLOPs), and Frames Per Second (FPS).

As mentioned in the introduction, this work is focused on developing a light model that is more suitable for resource-constrained applications such as mobile and embedded devices. Therefore, our primary interest is to compare SFRSeg with other *light* models having 2 million parameters or less, which are more likely to achieve real-time performance on mobile devices. However, we also include other large real-time and even offline models to demonstrate that the performance gap with these larger models is reduced.

4.1. Data sets

4.1.1. Cityscapes

Cityscapes [3] is the most widely used data set for semantic segmentation. It consists of around 5000 finely and 20,000

coarsely annotated images of urban street scenes from cities in Germany. All objects are grouped into 35 classes and 8 different categories. Following the literature, we used 19 classes for semantic experiments. Among 5000 fine-tuned data set, 2975 images are used for training, 500 for validation and the remaining 1525 for testing. We employed only the finely annotated images to evaluate the model performance. The test results were generated by the online evaluation server and are available from the following anonymous link: https://bit.ly/3cfcgLQ.

4.1.2. KITTI

KITTI [36] is another data set for semantic segmentation in addition to other computer vision tasks. It supplies 200 training images with fine annotation and 200 test images without annotations. Its class labelling scheme is compatible with Cityscapes. Hence, similar training and evaluation protocols were followed for KITTI. To address the small number of samples in this data set, we also used transfer learning by starting with pre-trained weights on Cityscapes in the first epoch. The test performance was obtained objectively by submitting prediction to its evaluation server.

4.1.3. CamVid

CamVid [37] was mainly designed for motion-based segmentation and recognition. It provides 701 fine-tune images, out of which 267 images are used for training, 101 for validation and 233 for testing. Following the literature, this study used 11 classes out of 32. Like KITTI, this data set contains a small number of samples. Therefore, we also utilized the pre-trained weights on Cityscapes when training the model on CamVid. As this data set does not have an evaluation server, we randomly split the data set to obtain a subset of 233 samples for testing.

4.1.4. Indoor objects data set

We also evaluate our proposed model using a recent indoor objects data set [38], which was created for indoor navigation, such as wheelchair users and service robots. This data set is useful as it provides the surrounding details of corridor objects of various sizes. The data set provides 1548 images with fine annotations and objects are grouped into 9 classes. We use 1328 images for training and 220 samples for validation.

4.2. Implementation details

We train our model using a computer equipped with three NVIDIA Titan RTX GPUs. For high-resolution (Cityscapes, Indoor objects) input images, we set the batch size to 4, and for low-resolution (KITTI and CamVid) images, we set it to 8. We used `CUDA` 10.2, `tensorflow` 2.1.0, `keras` 2.3.1 and `horovod` 0.19 to implement and train the proposed model. For experiments measuring inference speed, we used TensorRT 6.0.1 and followed the common practice of converting a keras model to a TRT-based deployment model, then used the TensorRT engine to measure the inference. We also measured the size of the deployment model as an indication of GPU memory requirement.

To set the learning rate (LR), We used the polynomial decay strategy [6] with the power set to 0.9. To minimize the model loss, the distributed synchronous stochastic gradient decent (SGD) optimizer was used. To improve convergence during training, we also employed a gradual warm-up strategy [39]. As is standard in segmentation, this study employed various data augmentation techniques, including cropping, resizing, clipping by value, horizontal and vertical flipping, adjusting brightness, contrast, and saturation of the input, to address the limited sample size problem. To avoid overfitting, we deployed ℓ_2 regularization and a dropout layer with a dropout rate of 0.35. For more implementation details, reader can refer our official repository at https://github.com/tanmaysingha/SFRSeg/tree/main/Supplementary.

4.3. Ablation study

In what follows, we examine different aspects of the proposed model and justify our choice. We select CamVid for this ablation study. For better compatibility of the tensor size, we resize the input image to 640×896 when training and the result of each submodule training is reported after 500 epochs on CamVid validation set.

The design of the proposed backbone is conceptualized based on two ideas: shared multi-branch approach for enhancing the learning process and reusing global features for better contextualization. The previous study [13] has shown the effectiveness of shared multi-branch design over the independent multi-branch design. Motivated by it, the current study also adopted the shared branch concept, however our the design of our multiple sub-encoders is fundamentally different: the model only take an input image of a single size as opposed to two different sizes as the case with SCMNet [13]. Such a change will reduce the computational cost while training the model. It has been observed that the majority amount of training time per epoch is taken by data pre-processing. Thereby, when two different sizes input are given to the model, both input will be processed first before the actual training gets started. It takes longer time to pre-process both the input. On the other hand, for the single input shared multi-branch approach, only one input will be augmented. To validate the justification, this study conducted an experiment and reported the corresponding result in Table 4. Only the first sub-encoder without CCM and DIS-CAM module is utilized to carry out this experiment. Table 4 clearly illustrates that the shared multi-branch approach with a single input reduces the training time per epoch by almost a half compared to that with dual inputs of the same image with different sizes, and so is the time to converge. GPU memory usage, model parameters and GFLOPs (giga FLOPs) are almost the same in both cases, which indicates that both approaches will have almost similar efficiency during inference. The observations from this ablation study supports the choice to use the shared multi-branch approach with a single high-resolution input image, which will be down-sampled by a quarter before it gets processed by two parallel shared branches (see Fig. 3).

Table 5 shows the effectiveness of multiple sub-encoders (SEs) for feature reuse. It is noticeable that successive additions of sub-encoders clearly enhance the performance with the best mIoU being as high as 74.4%, whilst only incurring a minor increase in computational cost. Even when all the four sub-encoders are included, our model has only 1.44 million parameters and 4.81 GFLOPs. Similar observation can be drawn from Fig. 7. It clearly shows that with the successive addition of sub-encoders in the backbone, region localization and segmentation are more accurate.

The yellow rectangle boxes in Fig. 7 show such a refinement process through multiple sub-encoders. It also shows that with the addition of each sub-encoder, the detection of the tiny objects (highlighted by the green boxes in Fig. 7) is improved. Although, some incorrect classifications can be observed (highlighted by red boxes) in the scene. Here, SFRSeg fails to detect the presence of the bicyclist in front of the bus. This could be due to the lack of texture differences between the bicyclist and bus class objects. As we ignore the void class (defined by the black color in the annotation), pixels in the void region are influenced by the large neighboring objects in the scene in each prediction, e.g. the bonnet of the car is labelled by the road class. We note however that void class pixels are not usually included in the evaluation metric. Overall, SFRSeg produces good performance, both qualitatively and quantitatively, through reusing feature maps in multiple sub-encoders.

Table 5 also displays the frames-per-second (FPS) count for each setup when input images of 640×896 from the CamVid data set

Table 4
Multiple input sizes vs a single input size (using CamVid).

No. of inputs	Param.	GFLOPs	Training time (S) per epoch	Memory Consum.	mIoU(%)
2	0.49M	2.89	26.0	4.8 GB	72.2
1	0.49M	2.96	14.3	4.8 GB	72.7

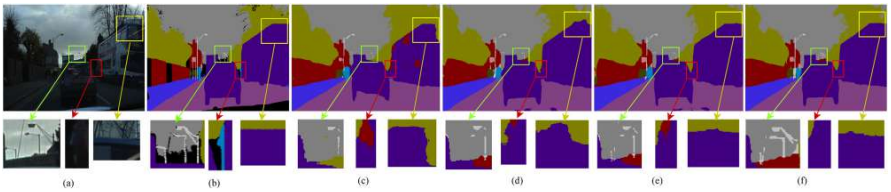

Fig. 7. Model's predictions by successive addition of each sub-encoder. (a) RGB input, (b) Annotation, (c) Prediction by SE1, (d) Prediction by SE1 + SE2, (e) Prediction by SE1 + SE2 + SE3, (f) Prediction by SE1 + SE2 + SE3 + SE4.

Table 5
Model performance analysis with multiple sub-encoders (using CamVid data set).

SE1	SE2	SE3	SE4	Param.(M)	GFLOPs	mIoU(%)	FPS
✓				0.49	2.96	72.7	338
✓	✓			0.93	4.51	73.6	302
✓	✓	✓		1.08	4.65	74.0	301
✓	✓	✓	✓	**1.44**	**4.81**	**74.4**	**290**

Table 6
Model performance analysis with different number of channels (using CamVid data set).

No. of max. channels	Param.	GFLOPs	mIoU(%)
64	0.84	4.19	73.7
128	**1.44**	**4.81**	**74.4**
256	2.89	5.89	74.4
512	9.45	10.1	74.2
1024	36.98	33.3	75.5
2048	146.91	129.0	76.5

are used. It shows that with the addition of each sub-encoder, the FPS count decreases as the model size increases. However, such architecture reuses the feature maps through multiple sub-encoders and produces better refined segmented output. Thereby, to obtain the best performance, this study decided to use the multiple sub-encoder design. It is concluded that processing the input through multiple sub-encoders of moderate depth is more effective than processing it through a single very large backbone. It reuses deep features multiple times and helps each layer retain more contextual details.

To reduce the number of parameters and GFLOPs, we also have studied the maximum depth of the global feature map. Many offline segmentation models, such as DeepLab [5], PSPNet [6], have a maximum 2^{11} number of channels for an input of $1024 \times 2048 \times 3$, which produces a large number of parameters and GFLOPs. A ResNet residual block first squeezes the channel of the input feature map and then expands it by 2. On the other hand, real-time semantic models which exploit bottleneck residual block of MobileNetV2 to build the backbone, have the input with 2^7 channels. A MBConv block of MobileNetV2 first expands the channel dimension by 6, hence the global feature map at the lowest spatial dimension can obtain large number of channels for capturing the more contextual details. Later, MBConv block squeezes the channel dimension of the output feature maps to reduce the parameters and FLOPs. Thus, it controls the computational cost.

Our ablation study shows that the channel number of 2^7 would be the best for the proposed model to keep a balance between model efficiency and efficacy. Table 6 displays the performance with different number of channels. With 2^7 channels, the proposed model produces 74.4% mIoU with 1.44 M parameters and 4.81 GFLOPs. With 2^8 and 2^9 channels, its performance becomes saturated and its computational cost has increased. With 2^{11} channels, a further 2% increase in performance has been recorded, however the number of parameters and GFLOPs have increased 102 and 27 times respectively. Therefore, we deploy 128 channels in the backbone network for an optimal trade-off between efficiency and performance.

Table 7
Results of first ablation study (using CamVid data set).

Refinement module	HPA-SA	Param.(M)	GFLOPs	mIoU(%)
		1.44	4.81	74.4
CMM		1.51	4.97	74.3
CCM		1.48	4.94	74.9
CCM	✓	**1.56**	**10.3**	**75.9**

All the above experiments have been conducted on CamVid without using CCM and HPA-SA. The following experiment examines the impact of these two modules on the segmentation performance. We also compare the impact of CCM and Context Mining Module (CMM) [13] on the model's performance. In Table 7, the first row displays the results of the model without any refinement and feature aggregation module (similar to Tables 5 and 6).

CMM has 2 image pooling branches, one separable branch and one fine-tuned branch. These branches are parallel to each other. Although these branches produce various sizes of the receptive field, the boundary degeneration effects can still be noticed in the image pooling branches due to a larger pool size. Instead of image pooling branch, separable branches with a larger dilation rate are more efficient for context mining. Due to these reasons, we propose a new design CCM for context mining and refinement. It also has four branches: one fine-tune and 3 separable branches with different dilation rates, but the feed-backward connection from the predecessor output to the successor branch input improves the refinement process in each successor branch. This performance enhancement can be noticed in Table 7. The model with CMM achieves 74.3% validation mIoU, whereas the model with CCM enhances model's performance by 0.5 percent-point while having less parameters and FLOPs. Later, we deploy HPA-SA on top of the CCM module which further increases the performance by 1 percent-

Table 8
Class-wise SFRSeg performance on Cityscapes validation and test sets.

D. set	Road	S.walk	Build.	Wall	Fence	Pole	T.light	T.sign	Veg.	Terrain
Val. set	**98.1**	83.3	**91.5**	44.9	50.9	61.3	68.3	72.4	**92.1**	70.1
	Sky	Person	Rider	Car	Truck	Bus	Train	M.cycle	Bicycle	**mIoU**
	95.1	80.5	58.4	**94.3**	52.4	66.5	59.7	56.1	68.4	**71.8**
D. set	Road	S.walk	Build.	Wall	Fence	Pole	T.light	T.sign	Veg.	Terrain
Test set	**98.0**	82.6	**90.9**	45.9	49.9	49.0	62.1	66.8	**91.4**	68.3
	Sky	Person	Rider	Car	Truck	Bus	Train	M.cycle	Bicycle	**mIoU**
	94.7	78.0	54.5	**93.1**	57.9	69.1	67.8	54.7	66.0	**70.6**

Table 9
Category-wise SFRSeg performance on Cityscapes validation and test sets.

Data set	Flat	Construction	Object	Nature	Sky	Human	Vehicle	mIoU
Val. set	**98.3**	**92.0**	68.3	**92.2**	**95.1**	80.5	**93.7**	**88.6**
Test set	**98.2**	**91.1**	58.0	**94.7**	**90.9**	78.5	**92.4**	**86.3**

point. Without HPA-SA, the global feature map is excessively upsampled to produce the final segmented output. Most of the existing real-time semantic segmentation models such as DFANet [12], ContextNet [14], and Fast-SCNN [15] upsample the global feature maps by 2^3 times to reduce the computational cost and improve model's efficiency. However, it has been observed that upsampling global feature map by a large factor directly causes considerable boundary degeneration effects and compromises the performance. This phenomena can be observed in the Table 7. Without using HPA-SA, the model produces 74.9% mIoU, whereas it becomes 75.9% with HPA-SA. Here, the global feature map is successively up-sampled and then mapped with the previous level's feature map. After every mapping, the feature is refined by SCB. This process goes on till the feature map has the size of the original input image. This gradual semantic aggregation and refinement improves the quality of the output.

4.4. Model evaluation

As discussed in the introduction, this works specifically focuses on light real-time models that have a very small number of parameters as they are more likely to be suitable for resource-constrained applications, such as mobile and edge devices. However, we also include offline models and large real-time models in order to show how light real-time models can achieve competitive performance whilst being much smaller. Three data sets related to outdoor urban street scenes and one indoor scene data set are used to evaluate the proposed model SFRSeg. We note that some offline (DeepLab [5], HANet [8], DenseASPP [23]) and real-time (DFANet [12], BiseNetV2 [28]) semantic models use variants of ResNet as the backbone and their pre-trained weight on ImageNet [40]. Here, SFRSeg is designed from scratch by employing bottleneck residual blocks of MobileNetV2 [18]. We do not pretrain our proposed model on ImageNet, knowing that this could potentially improve our model's performance further.

4.4.1. Performance on Cityscapes

The proposed model is first trained on Cityscapes training set at full input resolution (1024×2048) for 1000 epochs with a batch size of 4. Tables 8 and 9 display class and category-wise model performance on Cityscapes validation and test sets. The results of the top 5 classes and categories are highlighted in the tables. Due to the uneven class distribution in the whole data set, its performance varies in other classes such as wall, fence and pole. The proposed model produces 70.6% and 71.8% mIoU on Cityscapes test and validation set.

Performance comparison Table 10 illustrates the performance of other existing real-time semantic segmentation models and compares them with the proposed model. While the focus of this study is on light real-time models, offline and large real-time models are also included for completeness. It should be noted that the GFLOPs count can be used to determine model efficiency, but it also depends on the input image resolution and model size. Later in the paper, the impact of input resolution on GFLOPs count and model efficiency is shown.

Table 10 displays the common performance metrics used in semantic segmentation and they are grouped according to the model size. Most of the existing semantic segmentation models reported only test mIoU of Cityscapes data set, and many of their results are not available on the Cityscapes evaluation website. Therefore, other performance measurement metrics of these models could not be reported. The sign '-' in Table 10 indicates missing results, both in the literature and on the Cityscapes evaluation website.

In the offline category, *DeepLabV*3+ [5] and VideoPro [32] have achieved 82.1 + % class mIoU on Cityscapes test set. These models used extra data sets such as ImageNet, coarse images of Cityscapes, and synthetic images for performance enhancement. Moreover the deployment of large backbone such as Xception [11], ResNet101 [41] helps these models achieve excellent segmentation. VideoPro [32] is basically a combination of video prediction and a deep semantic segmentation models in which a video prediction model generates synthetic large data set for semantic segmentation. It also deployed a boundary pixel relaxation pre-processing technique for better semantic representation. Such joint strategies help the model achieve 83.5% mIoU on Cityscapes test set. Other offline models such as PSPNet [6], HANet [8], CENet [9] also accomplishes 80% to 82% test mIoU due to the large ResNet backbone.

In the real-time category, LERNet [9] and ENet [24] are the smallest semantic segmentation models with less than 1 million parameters. However, their test accuracy ranges from 58% to 66.5%. In comparison,moderately large semantic models such as SwiftNet ResNet variant (SwiftNet-RN18) [1], DFANet [12], ICNet [19], and STDC [29] achieved a test mIoU ranging from 69.5% to 76.8% while having 8 to 13 million parameters. Two recent methods, TopFormer [35] and LR-ASPP [42], have a test mIoU close to our proposed SFRSeg, but they have 2 to 3 times the number of parameters, leading to lower inference speed. Other shallower models such as SCMNet [13], ContextNet [14], Fast-SCNN [15], and FANet [16] generate moderate results (64.1% to 68.3% test mIoU) with 1 to 1.2 million parameters. From the above results, it is evident that performance can be enhanced with the increase of model size and an effective design. However, models like SwiftNet [1], DFANet [12],

Table 10
Performance evaluation of different models on Cityscapes validation and test sets.

Type	Model	Param. (M)	FPS	Val. Class mIoU (%)	Val. Cat. mIoU (%)	Test Class mIoU (%)	Test Cat. mIoU (%)	Test Class iIoU (%)	Test Cat. iIoU (%)
Offline models	PSPNet [6]	250.8	0.8	-	-	81.2	91.2	59.6	79.2
	FCN8s [21]	134	2	-	-	65.3	85.7	41.7	70.1
	RefineNet [25]	68.1	-	-	-	73.6	87.9	47.2	70.6
	CENet [9]	66.6	-	77.6	-	81.5	-	-	-
	HANet [8]	65.4	-	80.3	-	80.9	91.2	-	-
	DeepLabV3+ [5]	43.0	-	-	-	82.1	92.0	62.4	81.9
	GANet [31]	>44	-	80.9	-	81.6	91.6	63.6	81.0
	VideoPro [32]	>43	-	81.4	-	83.5	-	-	-
	CDN [30]	>28	-	80.3	-	80.5	91.9	62.5	83.4
Large real-time models	SegNet [10]	29.5	16.7	-	-	57.0	79.1	32.0	61.9
	STDC2 [29]	12.5	188.6	77.0	-	76.8	-	-	-
	S.NetV2-RN [1]	11.8	134.9	70.4	-	75.5	89.8	52.0	77.2
	STDC1 [29]	8.4	250.4	74.5	-	75.3	-	-	-
	DFANet [12]	7.8	100	71.9	-	71.3	-	-	-
	ICNet[19]	6.7	30.3	-	-	69.5	-	-	-
	BiseNetV1 [27]	5.8	105.8	69.0	-	68.4	-	-	-
	BiseNetV2 [28]	5.2	156	73.4	-	72.6	-	-	-
	DFANet-B [12]	4.9	120	68.4	-	67.1	-	-	-
	TopFormer-B [35]	4.84	53.59	65.52	-	64.94	83.18	35.99	63.26
	LR-ASPP [42]	3.28	12.92	69.54	70.89	68.95	87.00	41.41	71.17
Light real-time models	SCMNet [13]	1.2	117	66.5	84.2	68.3	87.2	38.3	69.1
	Fast-SCNN [15]	1.2	266.3	68.6	-	68.0	84.7	37.9	63.5
	FANet [16]	1.1	253	65.9	83.6	64.1	83.1	33.2	61.1
	ContextNet [14]	1.0	136.2	65.9	-	66.1	82.8	36.8	64.3
	LERNet [26]	0.7	100	69.5	-	66.5	-	-	-
	ENet [24]	0.4	76.9	-	-	58.3	80.4	34.4	64.0
	SFRSeg	**1.6**	**194**	**71.8**	**88.6**	**70.6**	**86.3**	**41.3**	**65.8**

- sign means results are not available in literature and Cityscapes evaluation server.

and SDTC [29] are 8 to 10 times larger than all shallow models. To reduce this gap, SwiftNet [1] proposed another pyramid-based mobile variant using MobileNetV2 (MV2) [18] as an encoder. It produced 77.4% mIoU on the Cityscapes validation set while having only 2.7 M parameters. However, SwiftNet-MV2 did not report its performance on the test set, which is considered much more objective.

Compared to the aforementioned models, our proposed model SFRSeg competitively produces 70.6% and 71.8% mIoU on Cityscapes test and validation sets while having only 1.6 million parameters and 37.9 GFLOPs at *full* input resolution. This is a much more balanced trade-off between performance and complexity, placing its at the top among *light* real-time models having less than 2 million parameters. For a fair comparison, we report all performance measurement metrics provided by the Cityscapes evaluation server. For completeness, we have also included in Table 10 the FPS counts measured at 512×1024 input resolution, and the information was extracted from either the literature, the best known public implementation or our own implementation that we can run on our system. As can be seen, SFRSeg can process 194 frames per second at this resolution, being very competitive against other real-time models.

As both inference speed and GFLOPs depend on input resolution, in addition to hardware and topology, we also conduct a separate experiment to measure them at different input resolutions and on different data sets. We report the results in Table 11.

It is a common practice to convert semantic segmentation models implemented in `PyTorch` or `tensorflow` to a deployment model using `TensorRT`. We note that the impact of `TensorRT` optimization generally varies across all models and platforms. Following this approach, we also optimize our `tensorflow` implementation with `TensorRT` and measure its performance. We report GFLOPs and frames-per-second (FPS) counts for this purpose. We also report the deployment model size obtained from `TensorRT` as it is also an indication of the GPU memory requirement. For a fair comparison, we reproduced the work of several existing real-time semantic segmentation models using `tensorflow` based on the literature and publicly available implementations, and measure the FPS under the same system configuration. We also measure the GFLOPs count of each `tensorflow` model at different input resolutions and reported the results in Table 11.

We observe that the deployment size of SFRSeg is similar to other light real-time models (around 10 MB), while other models are 3–5 times larger. This means that the former are more likely to be suitable for devices with less memory or a given device can deploy many more light real-time models simultaneously. SFRSeg also achieves competitive inference speed on three data sets compared to other large real-time models, such as DFANet [12], ICNet [19], BiseNetV2 [28], STDC1 [29], and STDC2 [29]. It is only behind ContextNet [14], Fast-SCNN [15], and FANet [16], but obviously these light real-time models do not achieve the same level of accuracy as SFRSeg.

We also observe that models like DFANet [12], ICNet [19], and STDC1 [29] have low efficiency due to the large number of GFLOPs. When the input resolution increases, the FPS of each model drops, and their GFLOPs increase. For instance, DFANet has 112 FPS on KITTI [36] with 55.3 GFLOPs whereas on Cityscapes [3], it produces 37 FPS with 236 GFLOPs. The FPS and GFLOPs counts may differ from the literature due to different hardware settings and different input resolutions. Among the large real-time models, STDC2, which produces the best test accuracy (76.8%) on Cityscapes, has 36, 26, and 4 FPS on KITTI, CamVid and Cityscapes, respectively. In comparison, the proposed SFRSeg has 209, 170 and 54 FPS, respectively, and its GFLOPs count ranges from 8.9 to 37.9, whereas for STDC2, it varies from 696 to 2970. In terms of number of parameters, the proposed model is at least 5 times smaller than STDC1 and STDC2. It clearly illustrates that the proposed model is more efficient than recent state-of-the-art semantic models whilst offering a very competitive segmentation performance.

Table 11
GFLOPs and FPS at different input resolution. The column Size refers to the model's deployment filesize optimized by TensorRT.

Model	Size (MB)	Params. (million)	KITTI 384 × 1280		CamVid 640 × 896		Cityscapes 1024 × 2048	
			GFLOPs	FPS	GFLOPs	FPS	GFLOPs	FPS
STDC2 [29]	47.2	11.2	696	36	812	27	2970	4
STDC1 [29]	29.9	7.2	684	37	798	27	2920	4
DFANet [12]	40.5	7.8	55.3	112	64.4	141	236	37
ICNet [19]	29.7	6.7	13.7	170	15.9	159	58.4	43
BiseNetV2 [28]	27.3	5.2	31.1	111	35.9	125	133	51
Fast-SCNN [15]	9.6	1.2	3.5	431	4.1	344	14.9	113
SCMNet [13]	10.5	1.2	8.9	113	9.9	98	38.3	27
ContextNet [14]	7.0	1.0	2.31	**465**	2.67	**407**	9.9	**138**
FANet [16]	7.9	1.1	2.7	305	2.7	253	11.4	74
ENet [24]	7.7	0.4	8.7	142	10.2	123	37.3	27
SFRSeg	14.0	1.6	8.9	209	10.3	170	37.9	54

Table 12
Performance evaluation on KITTI test set. Entries with the sign '-' mean the result is not available in the literature.

Type	Model	Pre-train	Class mIoU (%)	Class iIoU (%)	Cat. mIoU (%)	Cat. iIoU (%)
Off line	PAG [7]	City. fine-tune	47.96	17.86	78.11	49.17
	SDNet [33]	Cityscapes	51.14	17.74	79.62	50.45
	VideoPro [32]	Mapilary, Cityscapes	72.8	48.7	88.9	75.3
	SGDepth(Seg.) [34]	Cityscapes	43.1	-	-	-
Real time	**SFRSeg**	Cityscapes	**49.3**	**17.4**	**77.9**	**46.8**

4.4.2. Performance on KITTI

Table 12 displays the performance on the KITTI data set [36]. To ensure better tensor size compatibility with our implementation, we slightly resize images from 375 × 1242 to 384 × 1280 when training SFRSeg. As mentioned previously, we utilize the pre-trained weights on Cityscapes to train the model on KITTI.

To the best of our knowledge, we have listed supervised and unsupervised large semantic segmentation models obtained from the leaderboard of KITTI. Since KITTI is a versatile data set mainly designed for stereo and depth analysis, it is not easy to draw a direct comparison with our proposed model, as the problem settings are different. One example of these methods is [32] which proposes a framework for improving semantic segmentation via video propagation and and label relaxation. Apart from the Cityscapes data set, it also produces synthetic training sets through video propagation and utilizes them for evaluating model performance on different data sets. Due to this joint strategy, it generates excellent results (72.8% test mIoU). Other depth estimation with semantic guidance models include PAG [7], SDNet [33] and SGDepth [34]. The semantic head of PAG, SDNet and SGDepth produce 47.96%, 51.14% and 43.1% test mIoU. However, all these models are comparatively larger than the proposed model due to their deep backbone. In comparison, the proposed SFRSeg produces 49.3% test mIoU with much less parameters and without using any specialized pre or post-processing techniques.

4.4.3. Performance on CamVid

The results on CamVid are presented in Table 13. Similar to the experiment on KITTI, we also resize input images to 640 × 896 for better compatibility during training and use the pre-trained weight on Cityscapes. The only difference here is that domain mapping is required before utilizing the pre-trained weights on Cityscapes. First, Cityscapes' 19 classes need to be mapped to CamVid's 11 classes. Second, the model needs to be trained on 11 classes of Cityscapes. Later, the generated weight on Cityscapes is then utilized to train the model on CamVid. For completeness, we show models belonging to the offline category as a reference. Table 13 shows that VideoPro [32] produces the best test mIoU (81.7%) due to their joint strategy and additional training set generated by the video prediction model. This is not a traditional semantic segmentation model. Their joint mechanism can be used in any semantic segmentation model for performance enhancement. Among other deep traditional scene segmentation models, PSPNet [6] achieves 69.1% mIoU on CamVid test set.

Now, we turn our attention to real-time semantic segmentation models, which is the main focus of this work. In terms of absolute segmentation performance, SwiftNetV2 [1], SegNet [10], BiseNetV2 [28], and STDC1 [29] achieve 73.7%, 71.2%, 76.7% and 73.0% test accuracy respectively. However, these models are still large and have 3 to 18 times more parameters than the proposed SFRSeg, which also competitively produce 74.7% test mIoU at 640 × 896 input resolution. We also generate class-wise IoU on CamVid test set and reported the result in Table 14. We found few studies which have presented CamVid class-wise test performance, and we presented the results in Table 14. It shows that SegNet [10] produces the best results in five classes (sky, building, pavement, tree, and car) and generates a 71.2% mIoU. In comparison, the proposed model SFRSeg achieves a 74.7% mIoU and produces the best results in pole, road, sign, fence and bicyclist classes, while having 18 times less parameters than SegNet. Moreover, the class-wise results also demonstrate that the proposed model performs better than others on tiny and rare objects such as pole, sign, fence and bicyclist.

4.4.4. Performance on Indoor objects

Table 15 displays model's class-wise performance on the Indoor objects data set. It is observed that the proposed model performs exceptionally well on 3 classes: background wall, door, and carpet floor, and achieves good results on movable door handle and fire extinguisher, even though the percentage of pixels belonging to these classes is around $1-2\%$.

However, it does not produce a good result on very rare classes that have less than 1% pixels, including pull door handle, push button, push door handle, and key slot. Overall, the proposed SFRSeg produces a 61.4% mIoU on this Indoor objects data set.

4.4.5. Qualitative results and analysis

In this section, we qualitatively examine the segmentation performance of SFRSeg. Fig. 8 displays the original RGB image, its colored annotation given by the Cityscapes, the segmented output

Table 13
Performance evaluation on the CamVid test set.

Type	Model	Param. (M)	Pre-train	Test class mIoU (%)
Offline models	PSPNet [6]	250.8	ImageNet	69.1
	VideoPro [32]	-	Cityscapes	81.7
Large real-time models	SegNet [10]	29.5	3,433 additional road scenes	71.2
	STDC2 [29]	> 12.5	Cityscapes	73.9
	S.NetV2-RN [1]	12.0	ImageNet	73.7
	STDC1 [29]	> 8.4	Cityscapes	73.0
	DFANet [12]	7.8	ImageNet	64.7
	ICNet [19]	6.7	No pre-training	67.1
	BiseNetV1 [27]	5.8	ImageNet	65.6
	BiseNetV2 [28]	5.2	ImageNet	76.7
Light real-time models	SCMNet [13]	1.2	Cityscapes	71.3
	FANet [16]	1.1	No pre-training	57.8
	LERNet [26]	0.7	Cityscapes	58.2
	ENet [24]	0.4	No pre-training	51.3
	SFRSeg	1.6	Cityscapes	**74.7**

- sign means results are not available in the literature.

Table 14
Class-wise SFRSegperformance on the CamVid test set.

Model	Sky	Buil- ding	Pole	Road	Pav- ment	Tree	Sign	Fence	Car	Pades- trian	Bicy- clist	mIoU
SegNet	**96.1**	**89.6**	32.1	96.4	**93.3**	**83.4**	52.7	53.5	**87.7**	62.2	36.5	71.2
ENet	95.1	74.7	35.4	95.1	86.7	77.8	51.0	51.7	82.4	**67.2**	34.1	68.3
BiseNetV1	91.9	82.2	25.4	93.3	77.3	74.4	42.8	49.7	80.8	53.8	50.0	65.6
SFRSeg	93.8	87.8	**43.5**	**96.5**	85.1	80.8	**55.9**	**66.2**	83.1	59.8	**69.0**	**74.7**

Table 15
Class-wise SFRSeg performance on Indoor objects validation set.

Model	Background wall	Door	Pull door handle	Push button	Movable door handle	Push door handle	Fire extin- guisher	Key slot	Carpet floor	mIoU
SFRSeg	**95.0**	**90.4**	20.3	31.0	69.8	34.8	**82.3**	38.9	**90.4**	61.4

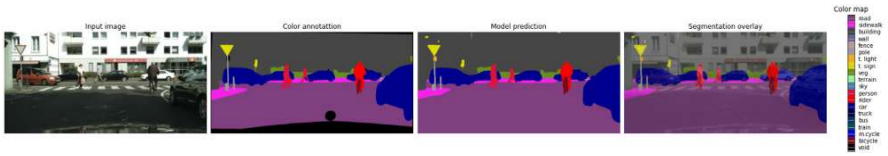

Fig. 8. Color map of Cityscapes data set and model prediction using validation sample.

Fig. 9. The first row displays the original images of the Cityscapes test set and the second row exhibits the corresponding output produced by the proposed model.

of a validation sample generated by SFRSeg, and the segmentation overlay for each objects in the scene. It shows a 20-color map, which also contains the void class by default. Although void class pixels are not included for calculating mIoU on the validation and test sets, the presence of the void class in training influences the model's performance while assigning the class of the neighboring pixels to void. This happens due to the high occurrence of void class within the data set. Hence, the void class is excluded when training the model with all data sets. The segmentation output of test samples can be seen in Fig. 9. Despite the percentage of pixels belonging to motorcycle, rider, truck, train and traffic light class being less than 0.5%, our model correctly identifies all these classes in the scene and assigns right class labels to all the pixels.

Similarly, we also displays the qualitative analysis on KITTI in Figs. 10 and 11. We note that the color mapping of KITTI is exactly same as Cityscapes. The result is obtained from KITTI's evaluation server. Along with the output, the evaluation server also generates an error image for each test sample. In Fig. 11, the first column

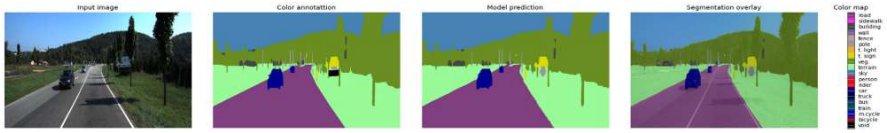

Fig. 10. Color map of KITTI data set and model prediction using validation sample.

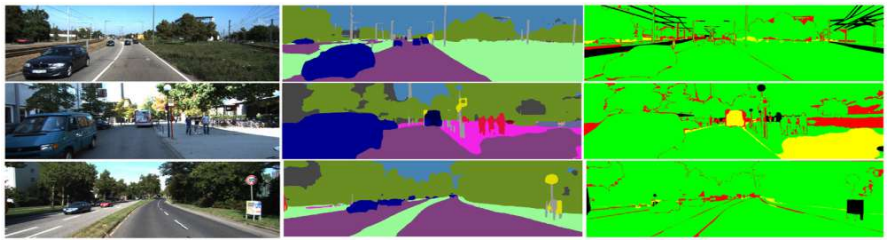

Fig. 11. The first column displays the original images of the KITTI test set, the second column exhibits the corresponding output produced by the proposed model and the third column shows the corresponding error images.

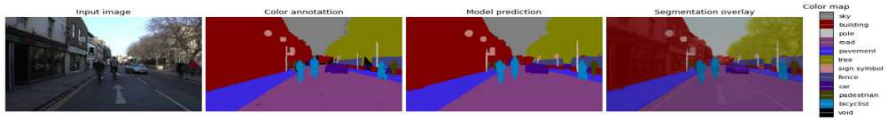

Fig. 12. Color map of CamVid data set and model prediction using validation sample.

Fig. 13. The first row displays the original images of CamVid test set and the second row exhibits the corresponding output produced by the proposed model.

refers to an RGB test sample, the second column shows the colored output produced by the model, and the third column displays the error image. Green pixels in the error image define the correct labelling, yellow pixels define an incorrect class label but correct category, black pixels describe the ground-truth labels not used for evaluation, and red pixels represent a wrong class label and wrong category. It can be clearly observed that the presence of red pixels in the test samples is very little. However, a major section of the second row error image shows the presence of yellow pixels, which still means correct category labelling.

Figs. 12 and 13 display the segmented output generated by the proposed SFRSeg using CamVid validation and test sets. Fig. 12 also exhibits the color codes of all 11 classes of CamVid data set. It uses different color codes than the other two data sets. It can be clearly seen that small objects like pole, sign symbol are correctly labelled by the proposed model and the smooth boundary of different geometric objects in the output demonstrates excellent performance by the proposed model.

Figs. 14 and 15 exhibit the proposed model's qualitative performance on the indoor objects data set [38]. It clearly demonstrates that, along with the urban street scenes, the proposed SFRSeg is equally capable of identifying various objects in indoor scenes. We use different color codes to define the objects in the scene. Fig. 14 shows the color map we use. While analyzing the data set, we observe that in some frames, objects like background wall, and door were annotated incorrectly. This can be seen in the first image of Fig. 15, highlighted by the red rectangle box. In the annotation, the background wall is defined as a void class object and highlighted by the black color code, although the proposed model rightly predicts the class of the background wall object and assigns a gray color code. In the second prediction, some incorrect classifications can be seen in the segmented output. Here, model fails to differentiate between the wooden door and the wooden wall (the yellow box) in the Fig. 15. This could be due to the lack of texture difference between these objects in the scene and also due to the smaller size of the data set.

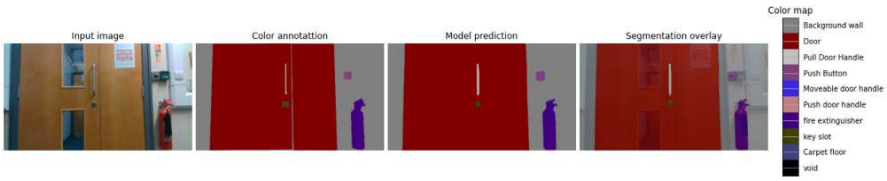

Fig. 14. Color map of Indoor objects data set and model prediction using validation sample.

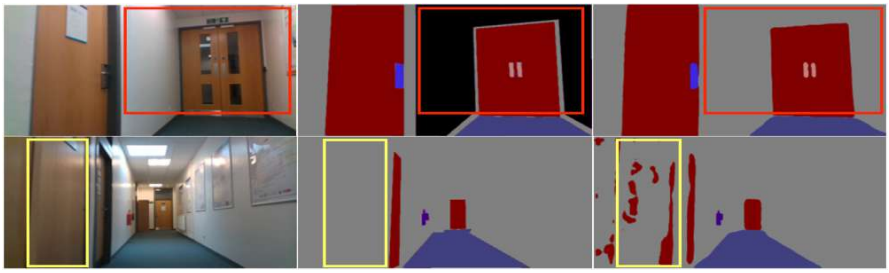

Fig. 15. Output by the proposed model using Indoor objects validation samples. (a) RGB image, (b) Colored ground truth, (c) Prediction. In the second column, there are incorrectly annotated items highlighted by red and yellow boxes. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Overall, the qualitative results on both indoor and outdoor scenes clearly demonstrate that not only does SFRSeg achieve segmentation performance metrics competitive to the state-of-the-arts, it also produces high-quality segmentation outputs which are sufficiently useful for real-time applications.

5. Conclusion

We have introduced the concept of shared multi-branch architecture and proposed a novel semantic segmentation model, named SFRSeg, for resource-constrained mobile devices which produces very competitive results on three popular benchmark data sets among the existing real-time scene parsing models. With only 1.6 M parameters, the proposed model significantly reduces the gap between the performance between real-time and offline semantic segmentation models thanks to key innovations: an effective shared-branch multiple sub-encoders design, a novel context mining module and a semantic aggregating module. The main limitation of our work is that we did not make use of any modern pre-processing and post-processing techniques, such as training on additional synthetic data sets extracted from video propagation models and exploiting additional boundary knowledge of each object in the scene. Similarly, the widely used network pruning technique can also be utilized to optimize the computational cost and improve model's efficiency. Hence, in future we will use these techniques for the performance enhancement of the proposed model. For better scene understanding, we also plan to deploy the proposed model for instance and panoptic segmentation which will help the model distinguish between stuff and things objects in the scene more effectively. Due to the growing demand for real-time semantic applications in various fields such as robotics, autonomous car industry, medical, agriculture, urban planning, civil engineering, the proposed model can be exploited for building crack detection for structural maintenance, medical image processing for early diagnosis, and many more. For reproducibility, the implementation of our model is available at https://github.com/tanmaysingha/SFRSeg.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

We used four publicly available datasets and we have cited all four datasets in our paper.

Acknowledgment

The authors would like to thank the reviewers for valuable comments that significantly improved both technical contents and presentation of this work.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.patcog.2023.109557.

References

[1] M. Oršić, S. Šegvić, Efficient semantic segmentation with pyramidal fusion, Pattern Recognit. 110 (2021) 107611.
[2] S. Lee, T. Son, S. Kwak, FIFO: learning fog-invariant features for foggy scene segmentation, in: Proc. CVPR, 2022, pp. 18911–18921.
[3] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, B. Schiele, The cityscapes dataset for semantic urban scene understanding, in: Proc. CVPR, 2016, pp. 3213–3223.
[4] A. Geiger, P. Lenz, C. Stiller, R. Urtasun, The KITTI vision benchmark suite, 2015. http://www.cvlibs.net/datasets/kitti.
[5] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, H. Adam, Encoder-decoder with atrous separable convolution for semantic image segmentation, in: Proc. ECCV, 2018, pp. 801–818.

[6] H. Zhao, J. Shi, X. Qi, X. Wang, J. Jia, Pyramid scene parsing network, in: Proc. CVPR, 2017, pp. 2881–2890.
[7] S. Kong, C. Fowlkes, Pixel-wise attentional gating for scene parsing, in: Proc. WACV, IEEE, 2019, pp. 1024–1033.
[8] S. Choi, J.T. Kim, J. Choo, Cars can't fly up in the sky: improving urban-scene segmentation via height-driven attention networks, in: Proc. CVPR, 2020, pp. 9373–9383.
[9] Q. Zhou, X. Wu, S. Zhang, B. Kang, Z. Ge, L.J. Latecki, Contextual ensemble network for semantic segmentation, Pattern Recognit. 122 (2022) 108290.
[10] V. Badrinarayanan, A. Kendall, R. Cipolla, SegNet: a deep convolutional encoder-decoder architecture for image segmentation, IEEE TPAMI 39 (12) (2017) 2481–2495.
[11] F. Chollet, Xception: deep learning with depthwise separable convolutions, in: Proc. CVPR, 2017, pp. 1251–1258.
[12] H. Li, P. Xiong, H. Fan, J. Sun, DFANet: deep feature aggregation for real-time semantic segmentation, in: Proc. CVPR, 2019, pp. 9522–9531.
[13] T. Singha, M. Bergemann, D.-S. Pham, A. Krishna, SCMNet: shared context mining network for real-time semantic segmentation, in: Proc. DICTA, IEEE, 2021, pp. 1–8.
[14] R.P. Poudel, U. Bonde, S. Liwicki, C. Zach, ContextNet: exploring context and detail for semantic segmentation in real-time, arXiv preprint arXiv:1805.04554 (2018).
[15] R.P. Poudel, S. Liwicki, R. Cipolla, Fast-SCNN: fast semantic segmentation network, arXiv preprint arXiv:1902.04502 (2019).
[16] T. Singha, D.-S. Pham, A. Krishna, FANet: feature aggregation network for semantic segmentation, in: Proc. DICTA, IEEE, 2020, pp. 1–8.
[17] T. Singha, D.-S. Pham, A. Krishna, J. Dunstan, Efficient segmentation pyramid network, in: Proc. ICONIP, Springer, 2020, pp. 386–393.
[18] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, MobileNetV2: inverted residuals and linear bottlenecks, in: Proc. CVPR, 2018, pp. 4510–4520.
[19] H. Zhao, X. Qi, X. Shen, J. Shi, J. Jia, ICNet for real-time semantic segmentation on high-resolution images, in: Proc. ECCV, 2018, pp. 405–420.
[20] D. Ji, H. Wang, M. Tao, J. Huang, X.-S. Hua, H. Lu, Structural and statistical texture knowledge distillation for semantic segmentation, in: Proc. CVPR, 2022, pp. 16876–16885.
[21] J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, in: Proc. CVPR, 2015, pp. 3431–3440.
[22] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proc. CVPR, 2016, pp. 770–778.
[23] M. Yang, K. Yu, C. Zhang, Z. Li, K. Yang, DenseASPP for semantic segmentation in street scenes, in: Proc. CVPR, IEEE, 2018, pp. 3684–3692.
[24] A. Paszke, A. Chaurasia, S. Kim, E. Culurciello, ENet: a deep neural network architecture for real-time semantic segmentation, arXiv preprint arXiv:1606.02147 (2016).
[25] G. Lin, A. Milan, C. Shen, I. Reid, RefineNet: multi-path refinement networks for high-resolution semantic segmentation, in: Proc. CVPR, 2017, pp. 1925–1934.
[26] J. Wu, Z. Wen, S. Zhao, K. Huang, Video semantic segmentation via feature propagation with holistic attention, Pattern Recognit. 104 (2020) 107268.
[27] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, N. Sang, BiSeNet: bilateral segmentation network for real-time semantic segmentation, in: Proc. ECCV, 2018, pp. 325–341.
[28] C. Yu, C. Gao, J. Wang, G. Yu, C. Shen, N. Sang, BiSeNet V2: bilateral network with guided aggregation for real-time semantic segmentation, Int. J. Comput. Vis. 129 (11) (2021) 3051–3068.
[29] M. Fan, S. Lai, J. Huang, X. Wei, Z. Chai, J. Luo, X. Wei, Rethinking BiSeNet for real-time semantic segmentation, in: Proc. CVPR, 2021, pp. 9716–9725.
[30] J. Fu, J. Liu, Y. Li, Y. Bao, W. Yan, Z. Fang, H. Lu, Contextual deconvolution network for semantic segmentation, Pattern Recognit. 101 (2020) 107152.
[31] P. Zhang, W. Liu, H. Wang, Y. Lei, H. Lu, Deep gated attention networks for large-scale street-level scene segmentation, Pattern Recognit. 88 (2019) 702–714.
[32] Y. Zhu, K. Sapra, F.A. Reda, K.J. Shih, S. Newsam, A. Tao, B. Catanzaro, Improving semantic segmentation via video propagation and label relaxation, in: Proc. CVPR, 2019, pp. 8856–8865.
[33] M. Ochs, A. Kretz, R. Mester, SDNet: semantically guided depth estimation network, in: German Conference on Pattern Recognition, Springer, 2019, pp. 288–302.
[34] M. Klingner, J.-A. Termöhlen, J. Mikolajczyk, T. Fingscheidt, Self-supervised monocular depth estimation: solving the dynamic object problem by semantic guidance, in: Proc. ECCV, Springer, 2020, pp. 582–600.
[35] W. Zhang, Z. Huang, G. Luo, T. Chen, X. Wang, W. Liu, G. Yu, C. Shen, TopFormer: token pyramid transformer for mobile semantic segmentation, in: Proc. CVPR, 2022, pp. 12083–12093.
[36] H. Abu Alhaija, S.K. Mustikovela, L. Mescheder, A. Geiger, C. Rother, Augmented reality meets computer vision: efficient data generation for urban driving scenes, Int. J. Comput. Vis. 126 (9) (2018) 961–972.
[37] G.J. Brostow, J. Fauqueur, R. Cipolla, Semantic object classes in video: a high-definition ground truth database, Pattern Recognit. Lett. 30 (2) (2009) 88–97.
[38] E. Mohamed, K. Sirlantzis, G. Howells, A pixel-wise annotated dataset of small overlooked indoor objects for semantic segmentation applications, Data Brief 40 (2022) 107791.
[39] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, K. He, Accurate, large minibatch SGD: training imagenet in 1 h, arXiv preprint arXiv:1706.02677 (2017).
[40] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, ImageNet: a large-scale hierarchical image database, in: Proc. CVPR, IEEE, 2009, pp. 248–255.
[41] Z. Wu, C. Shen, A. Van Den Hengel, Wider or deeper: revisiting the ResNet model for visual recognition, Pattern Recognit. 90 (2019) 119–133.
[42] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, et al., Searching for MobileNetV3, in: Proc. ICCV, 2019, pp. 1314–1324.

Mr. Tanmay Singha is currently a PhD research scholar in Computer Science at Curtin University, Australia. He has double Master's Degrees- Master of Technology in Information Technology and Master of Computer Applications from University of Calcutta, India. Before joining Curtin University, he served nine years as a lecturer in the department of IT at Royal University of Bhutan, Bhutan. His current research interest focuses on computer vision tasks such as semantic and instance segmentation, object detection, scene graph generation, indoor and outdoor scene analysis, human pose estimation, facial landmark detection and medical image processing using deep neural networks.

Dr. Duc-Son Pham received the PhD degree from Curtin University of Technology in 2005. He is currently a Senior Lecturer with the Discipline of Computing, Curtin University, Perth, Western Australia. His current research interests include sparse learning theory, large-scale data mining, convex optimization, and advanced deep learning with applications to computer vision and image processing. He is a Senior Member of the IEEE. He is a recipient of the Young Author Best Paper Award 2010 for a publication in IEEE Transactions on Signal Processing.

Dr. Aneesh Krishna is currently an Associate Professor with the School of Electrical Engineering, Computing and Mathematical Sciences, Curtin University, Australia. He holds a PhD in computer science from the University of Wollongong, Australia. He was a lecturer in software engineering at the School of Computer Science and Software Engineering, University of Wollongong, Australia (from February 2006 - June 2009). His research interests include AI for software engineering, model-driven development/evolution, requirements engineering, agent systems, formal methods, data mining, computer vision, machine learning, bio-informative and renewable energy systems. He has published more than 130 articles in reputed journals and international conferences. His research is (or has been) funded by the Australian Research Council (ARC), and various Australian government agencies (like NSW State Emergency Service) as well as companies such as Woodside Energy, Amristar Solutions, Autism West Incorporated, BW Solar Australia, Western Australia Dementia Training Center and Andrew Corporation. He serves as an assessor (Ozreader) for the ARC. He has been on the organising committee, served as invited technical program committee member of many conferences and workshops in the areas related to his research.

.9 Publication 9

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 11 March 2023, accepted 21 June 2023, date of publication 27 June 2023, date of current version 6 July 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3289968

RESEARCH ARTICLE

Multi-Encoder Context Aggregation Network for Structured and Unstructured Urban Street Scene Analysis

TANMAY SINGHA, DUC-SON PHAM, (Senior Member, IEEE), AND ANEESH KRISHNA
School of Electrical Engineering, Computing, and Mathematical Sciences, Curtin University, Perth, WA 6102, Australia
Corresponding author: Duc-Son Pham (dspham@ieee.org)

ABSTRACT Developing computationally efficient semantic segmentation models that are suitable for resource-constrained mobile devices is an open challenge in computer vision research. To address this challenge, we propose a novel real-time semantic scene segmentation model called Multi-encoder Context Aggregation Network (MCANet), which offers the best combination of low model complexity and state-of-the-art (SOTA) performance on benchmark datasets. While we follow the multi-encoder approach, our novelty lies in the varying number of scales to capture both global context and local details effectively. We introduce suitable lateral connections between sub-encoders for improved feature refinement. We also optimize the backbone by exploiting the residual block of MobileNet for resource-constrained applications. On the decoder side, the proposed model includes a new Local and Global Context Aggregation (LGCA) module that significantly enhances semantic details in the segmentation output. Finally, we use several known efficient convolution techniques for the classification module to make the model more computationally efficient. We provide a comprehensive evaluation of MCANet on multiple datasets containing structured and unstructured urban street scenes. Among the existing real-time models with less than 3 million parameters, the proposed model is more competitive as it achieves the SOTA performance without ImageNet pre-trained weights on both structured and unstructured environments while being more compact for resource-constrained applications.

INDEX TERMS Semantic segmentation, feature scaling, feature aggregation, deep learning, scene understanding, convolutional neural networks.

I. INTRODUCTION

Scene understanding is a crucial task in many learning systems and has numerous applications, including self-driving vehicles [1], [2], human-computer interaction, virtual reality [3], object detection [4], [5], medical image analysis [6], [7], [8] and online video surveillance [9]. Semantic scene segmentation is a fundamental step towards achieving scene understanding. The goal of semantic segmentation is to recognize and localize different categories in a scene, assigning a class or a label to every pixel. The categories can vary depending on the specific application, as shown in Figure 1.

A semantic segmentation model usually follows an encoder-decoder structure, where the encoder extracts semantic information, and the decoder projects it back to the input space for individual pixel classification. Inspired by the success of Deep Convolutional Neural Networks (DCNNs) in general classification tasks, many offline semantic segmentation models have been developed with deep architectures [10], [11], based on well-known backbone networks, usually ResNet [12], which is suitable for the segmentation task. For instance, DeepLab [11] is an approach that exploits ResNet by removing the striding operation from the last few ResNet blocks. Additionally, by utilizing high dilation rates

The associate editor coordinating the review of this manuscript and approving it for publication was Eduardo Rosa-Molinar.

FIGURE 1. Labelling class to each pixel in semantic segmentation. (a) Original input image, (b) colored annotation of the input image where each pixel of the image has a specific color code.

in the feature scaling module, DeepLab ensures a wider field of view for context inclusion.

Later, the ResNet model pre-trained on ImageNet with dilated convolutions has become a popular choice as a feature extractor for scene segmentation models, including DeepLabV3 [13], PSPNet [14], HANet [15], and OCR [16]. Although these DCNN-based models have shown outstanding performance, many of them are not designed for mobile devices and other resource-constrained applications. Therefore, they cannot achieve satisfactory real-time performance on embedded devices.

In many practical applications, such as mobile and IoT devices [17], [18], the available computing resources are limited. Therefore, deploying large models that can achieve satisfactory real-time performance is prohibitive. This has led to a growing interest in developing lightweight semantic segmentation models [19], [20], [21] for these specific applications. These real-time models aim to reduce the computational cost of existing offline models while still achieving satisfactory segmentation performance.

One major challenge in developing real-time models for mobile devices and resource-constrained applications is the high input resolution with a large field of view, which can significantly increase memory usage. To address this problem, some real-time models, such as BiSeNet [21], ICNet [22], RefineNet [23], and ContextNet [24], have introduced a new encoder design called a *multi-branch* encoder. This design consists of a shallow branch for high-resolution input and a deep branch for low-resolution input. By using a shallow branch with fewer layers, this approach effectively reduces the computational cost while controlling the field of view and maintaining global contextual information through the deep branch. Although several models have achieved computational efficiency, there is still a considerable performance gap between existing offline and real-time semantic scene segmentation models. Developing real-time lightweight models suitable for mobile devices and resource-constrained applications is still an open research question.

In this work, we address the above challenge for mobile devices and other embedded devices with inadequate hardware facilities through a novel architecture. Our solution starts with the observation from the literature on real-time semantic segmentation that the input needs to be processed at multiple scales with larger receptive fields to capture better contextual details for improved scene understanding.

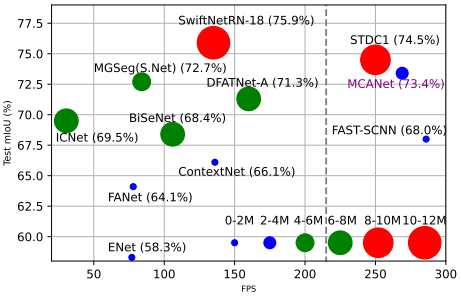

FIGURE 2. Test accuracy vs parameters among real-time models.

We introduce a completely new multi-encoder architecture in this study. Here, the number of stages in each successive sub-encoder is reduced, while the repetition of inverted residual blocks is increased in the successive sub-encoder. Consequently, each sub-encoder becomes deeper than the previous one, allowing for extraction of more semantic information from the scene. Lateral connections are also used at the same stage. After the complete encoding process, we obtain five rich global feature maps at different scales, which are then used for feature fusion at the decoder end. We demonstrate that MCANet enables excellent semantic segmentation performance while keeping the model complexity relatively low. Our design is at least two times smaller than existing real-time scene segmentation models that achieve state-of-the-art results, giving our model a competitive advantage in resource-constrained applications.

We make the following contributions in this research study:

- Introduce a novel backbone architecture, with multiple sub-encoders designed specifically for optimal feature scaling. At the same time, we also reduce the number of stages in each successive sub-encoder to control the number of model parameters.
- Introduce an effective multi-stage module for local and global feature aggregation at the decoder, which combines feature maps at different levels produced by the proposed backbone network.
- Relying on this novel backbone architecture and an efficient multi-stage feature aggregation module, introduce an efficient semantic segmentation model named MCANet that achieves the optimal trade-off between model accuracy and model efficiency for resource-constrained mobile devices. This can be viewed in Figure 2.
- Finally, we provide comprehensive experiments on both structured and unstructured environments with various numbers of classes and demonstrate the model's superior performance in all circumstances among the existing real-time semantic segmentation models having fewer than 3 million (M) parameters.

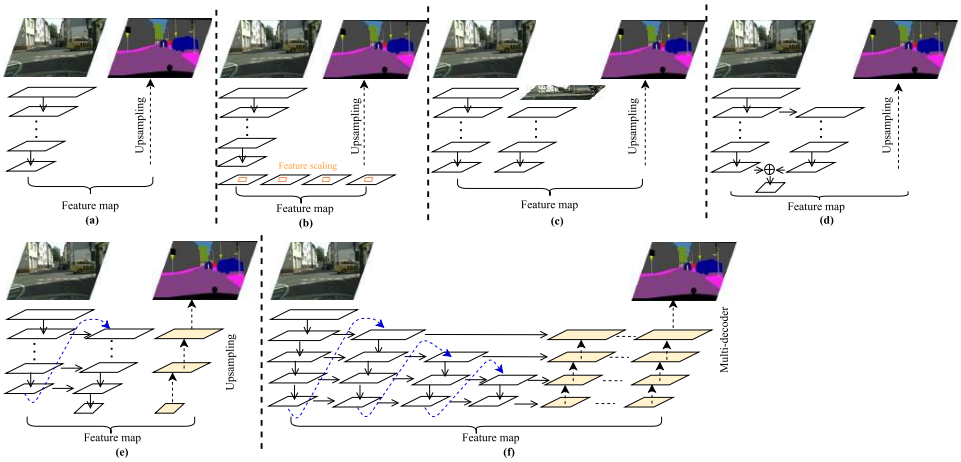

FIGURE 3. Different architectures of semantic segmentation models: (a) One-branch, (b) One-branch with feature scaling technique, (c) Multi-branch, (d) Dual-branch with down-sampling, (e) Feature re-use in sub-encoders with increasing stages, and (f) Feature re-use in multiple sub-encoders with decreasing stages, but increasing depth in each sub-encoder, and feature refinement through multiple decoding paths.

On structured datasets such as CamVid [25], BDD100K [26], and KITTI [27], as well as on unstructured datasets such as IDD-lite, the proposed model produces the state-of-the-art (SOTA) performance among the existing real-time semantic segmentation models. On Cityscapes [28], the proposed model generates a test accuracy of 73.4% without using ImageNet [29] or any pre-trained weights, which is the best performance among the existing real-time semantic models with fewer than 3M parameters. It can be visualized in Figure 2, which plots Cityscapes test mIoU (%) against FPS. The size of the circle in Figure 2 depicts the size of the model. Figure 2 clearly demonstrates the proposed model's superiority in terms of achieving the best balance between model accuracy and model efficiency.

The paper is organized as follows. In Section II, we present related work in semantic segmentation. Section III details our design, and Section IV discusses numerous experiments. Concluding remarks are given in Section V.

II. BACKGROUND

A. ONE-BRANCH DESIGN

As shown in Figure 3(a), a simple encoder-decoder architecture [10] was used in the early days of semantic segmentation, such as FCN [10], DeepLab [11], BiSeNet [21], and UNet [30]. The encoder typically contains a deep neural network to extract contextual details of the scene, and large backbone networks such as ResNet [12], VGG-16 [31], and Xception [32] are common choices. An extension of the one-branch approach is to include feature scaling (see Figure 3(b)), which is known to capture contextual details through a larger receptive field. For instance, PSPNet [14] introduced the Pyramid Pooling Module (PPM), which uses four image pooling branches with different bin sizes. DeepLabV3+ [13] introduced Atrous Spatial Pyramid Pooling (ASPP), which utilizes five dilation branches with different dilation rates. Using ResNet-101 [33] as the backbone, it achieved 82.1% test accuracy on Cityscapes [28]. Similar to feature scaling, a few semantic segmentation models [21], [34], [35], [36] have started to use the attention mechanism to guide the feature learning process using high-level information.

Recently, a new model known as MGSeg [37] has emerged, aiming to improve model efficiency further. It achieves this by utilizing a lightweight backbone (ResNet-18) and incorporating a hybrid feature attention and feature scaling module. It achieves impressive performance, achieving 77.8% test accuracy on Cityscapes. However, it should be noted that the model has 13.3 million parameters and requires 96.5 GFLOPs (Giga Floating Point Operations) at an input resolution of 1024×1024. Although MGSeg has made strides in improving efficiency, the large number of parameters in the whole design can still pose computational challenges, particularly for high-resolution input images. As a result, these models may not be suitable for resource-constrained applications where computational resources are limited.

B. MULTI-BRANCH DESIGN

The multi-branch encoder approach has been introduced recently to address the computational burden of the one-branch approach. Figure 3(c) shows the general

architecture of multi-branch encoders used in ICNet [22], RefineNet [23],ContextNet [24], and SwiftNet [38]. These models typically have a dedicated deep branch that accepts lower resolution input images and produces rich global feature maps. Additionally, several parallel shallow branches are deployed to extract low-level feature maps at higher resolutions. Therefore, a multi-branch encoder can handle higher resolution input images without incurring high computational costs. Despite being more efficient, the open challenge with the multi-branch encoder approach is to close the performance gap with the one-branch deep encoder approach. For instance, recently a new multi-branch semantic segmentation model, called SwiftNet [38], is introduced which has almost 4 times less parameters than DeepLabV3+ [13]. However, its test accuracy on Cityscapes is still 10% lower.

C. DUAL-BRANCH WITH DOWN-SAMPLING TECHNIQUE

This approach, as shown in Figure 3(d), is a deviation from the multi-branch approach. In this design, the encoder network takes a single input and employs a few convolution layers to down-sample the input image. Subsequently, two branches are formed: a deep branch is designed to extract rich global feature maps by exploiting a series of residual blocks whilst a shallow branch is to uproot local features using few convolution layers. Models such as such as BiSeNet [21], Fast-SCNN [39], FANet [40], and ESPNet [41] achieve improved accuracy and efficiency by employing this approach, which requires less data pre-processing.

D. FEATURE REUSE IN SUB-ENCODERS

It is known that deep convolution layers learn more contextual details than the layers at the initial stage, and problems like vanishing gradient can be diminished. Based on this fact, DFANet [42] has introduced the concept of feature reuse in its sub-encoders. Figure 3(e) demonstrates that the global feature of the first sub-encoder is used as input for the second sub-encoder. Before being fed to the next sub-encoder, the global feature is upsampled 2^3 times and passed through a fully connected (FC) attention module for better refinement. Thus, the features from a previous sub-encoder are reused in the next subsequent sub-encoder. Due to feature reuse, DFANet achieves 71.3% test accuracy on the Cityscapes test set while having 7.8 M parameters.

In contrast to the aforementioned designs, we propose a novel approach called the multi-encoder network, which leverages feature map reuse through dynamic sub-encoders and generates refined output through multiple decoding paths. Figure 3(f) provides an overview of the feature reuse in the multi-encoder design. The specifics of our proposed design will be discussed in the following section.

III. PROPOSED METHOD

Figure 4 depicts the end-to-end design of our proposed MCANet. The encoder architecture is based on the concept of reusing feature maps in a multi-encoder design. Specifically, features at lower resolutions contain rich semantic details that require multiple refinements to capture the full context. To achieve this, we employ multiple encoders that effectively utilize these features.

TABLE 1. Bottleneck residual block.

Input	Operator	Output
$h \times w \times c$	1×1 Conv,1/1, Relu	$h \times w \times tc$
$h \times w \times tc$	3×3 DwConv, 3/s, Relu	$h/s \times w/s \times tc$
$h/s \times w/s \times tc$	1×1 Conv,1/1, -	$h/s \times w/s \times c'$

A. MULTI-ENCODER

Figure 4(a) outlines the design of our proposed multi-encoder. In this design, we carefully select individual components to target mobile devices and other resource-constrained applications and optimally construct an encoder network to achieve the best extraction of semantic features.

Empirically, it has been shown that MobileNet [43] bottleneck residual convolution blocks (MBConv) are more efficient for mobile devices than other existing residual blocks [43], [44], [45]. The optimized architecture of these residual blocks and the utilization of depth-wise separable convolution layers in the bottleneck intermediate expansion stage make MBConv much more computationally efficient. For that reason, we decided to use MBConv blocks to build our multi-encoder. The layered architecture of an MBConv block is shown in Table 1. As can be seen, an input feature F_i with spatial dimensions $h \times w$ and channel dimension c is first filtered by a standard convolution layer and produces an output of size $h \times w \times tc$. The number of channels of the input feature is increased by an expansion factor t. The intermediate expansion stage uses a lightweight depth-wise convolution layer that reduces the computational cost by 8 to 9 times compared to a standard convolution layer. Thus, it optimizes the overall number of model parameters and GFLOPs count. Following [43], we utilize MBConv6 blocks to design our backbone. Except for the first residual block, we use an expansion ratio of 6 for other blocks. To better preserve contextual details, we do not apply ReLU non-linearity in the last layer of each MBConv block.

After down-sampling the original input image using a standard convolution layer, we use 11 MBConv blocks of varying expansion ratios to construct the first sub-encoder of our proposed multi-encoder. The literature suggests that using MBConv blocks of different expansion ratios at the initial stage can retain more contextual and spatial details due to its squeeze and excitation architecture [43]. The depth and width of each block are controlled by two tunable hyper-parameters: the width (M_w) and depth (M_d) multipliers. For an input feature map F_i of size $h_i \times w_i \times c_i$, each MBConv block produces an output feature map F_o of size $h_i/s \times w_i/s \times M_w c_i$. Here, h_i, w_i, and c_i denote the height, width, and number of channels of the input feature map, respectively. The stride s is used to control the number of stages in the encoder. Whenever s becomes 2, one new stage is created by down-sampling the input feature map size by half. Thus, we generate all six stages

TABLE 2. Layer architecture of the proposed multi-encoder.

Stage	Input	Operators	Width multiplier (M_w)	Depth multiplier (M_d)	stride	Output
Layer architecture of first sub-encoder						
1	1024×2048×3	Conv, k3×3	-	1	2	512×1024×32
2	512×1024×32	MBConv1, k3×3	0.75	1	2	256×512×24
-	256×512×24	MBConv6, k3×3	1.34	2	1	256×512×32
3	256×512×32	MBConv6, k3×3	1.5	2	2	128×256×48
4	128×256×48	MBConv6, k3×3	1.34	2	2	64×128×64
5	64×128×64	MBConv6, k3×3	1.5	2	2	32×64×96
6	32×64×96	MBConv6, k3×3	1.34	2	2	16×32×128
Layer architecture of second sub-encoder						
4	128×256×48	MBConv6, k3×3	1.34	3	2	64×128×64
5	64×128×64	MBConv6, k3×3	1.5	2	2	32×64×96
6	32×64×96	MBConv6, k3×3	1.34	2	2	16×32×128
Layer architecture of third sub-encoder						
5	64×128×64	MBConv6, k3×3	1.5	3	2	32×64×96
6	32×64×96	MBConv6, k3×3	1.34	2	2	16×32×128
Layer architecture of fourth sub-encoder						
6	32×64×96	MBConv6, k3×3	1.34	3	2	16×32×128

*Feature F7 is created using a MaxPooling operation after the forth sub-encoder.

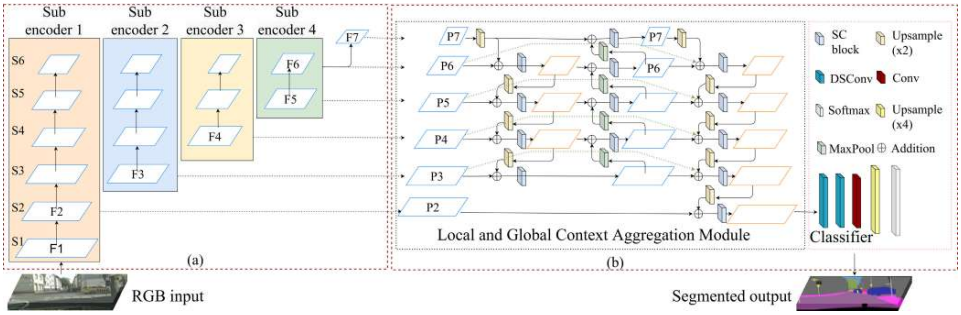

FIGURE 4. Complete pipeline of our proposed MCANet: (a) Multi-encoder design: Feature re-use in sub-encoders. Black dotted lines define the lateral connections from the previous sub-encoder at the same levels and green dotted lines show reusing of global feature map in the next sub-encoder. Feature F7 is produced using a simple pooling operation. (b) Decoder Design: Local and Global Context Aggregation (LGCA) module and Classifier module.

in the first sub-encoder. Hyper-parameter M_d controls the depth of the encoder by controlling the number of repetitions of each MBConv block. To keep our design simple, we repeat each block in the first sub-encoder twice, except the first block. Following [43], we set the range of the width multiplier from 0.75 to 1.5 and generate a maximum of 128 channels for the global feature maps. Models like DeepLab [13], PSPNet [14], and HANet [15] have global features with a maximum of 2048 channels, which contribute to large-scale parameters and GFLOPs. By setting the lower range for the width multiplier, we achieve the best trade-off between model performance and efficiency for mobile devices. Thus, the first sub-encoder generates rich spatial and global features without contributing a large number of parameters and GFLOPs. The complete layer architecture of our proposed multi-encoder is illustrated in Table 2.

Empirical studies have shown that high-level features at lower resolutions contain rich semantic details that are conducive to attention weight learning [13], [37], [42]. Therefore, many models, such as DRANet [35] and DFANet [42], introduce an attention module in the later stages of their encoder networks. The attention mechanism utilizes global pooling to capture global contexts and guides the feature learning process by computing an attention vector. In DFANet, the final global feature of each sub-encoder passes through a fully connected (FC) attention module before being used as an input feature map for the next sub-encoder. It has been demonstrated that deploying the FC module enhances model performance by 4-6%. However, the effective design of our proposed multi-encoder eliminates the need for an attention module on top of each sub-encoder [42], thereby reducing the number of parameters and computational cost.

In DFANet, each sub-encoder has the same layered architecture, but the spatial dimensions of the input feature map for each sub-encoder differ. Consequently, an additional deep stage is created after each sub-encoder. The drawback of this architecture is that the deep features are not optimally reused. For instance, in DFANet, features at the fourth, fifth,

and sixth stages are reused three, two, and one time, respectively. However, high-level features (the sixth stage) should be processed more extensively compared to intermediate and shallow features. Additionally, the uniform layered architecture of each sub-encoder is ineffective in acquiring any additional knowledge while reusing deep feature maps:

$$F^2_{l_3} = \text{Conv}(\text{Upsample8}(F^1_{l_6})), \quad (1)$$

$$F^2_{l_3} = F^1_{l_3} + F^2_{l_3}. \quad (2)$$

In contrast to DFANet, our first sub-encoder consists of all six stages, producing both local and deep global features. The final feature at the sixth stage of the first sub-encoder is upsampled 2^3 times and added to the output of the third stage of the first sub-encoder, as described mathematically in Equations 1 and 2. In the feature maps $F^i_{l_j}$, where i represents the sub-encoder number and j represents the level (l) position, $F^1_{l_6}$ and $F^1_{l_3}$ denote the output of the sixth (l_6) and third (l_3) stages of the first sub-encoder ($i = 1$), respectively. The feature map $F^1_{l_6}$ from the sixth stage of the first sub-encoder is used to produce the feature map $F^2_{l_3}$, which is subsequently used as an input for the second sub-encoder and refined through its fourth, fifth, and sixth stages. Lateral connections are used to reuse features from the last three stages of the first sub-encoder. Similarly, the third and fourth sub-encoders are designed, and their operations are described mathematically in Equations 3, 4, 5, and 6, which define the operations performed before processing feature maps with the third and fourth sub-encoders. 'Upsample8,' 'Upsample4,' and 'Upsample2' refer to scaling up the feature map by 2^3, 2^2, and 2^1 times, respectively:

$$F^3_{l_4} = \text{Conv}(\text{Upsample4}(F^2_{l_6})), \quad (3)$$

$$F^3_{l_4} = F^2_{l_4} + F^3_{l_4}, \quad (4)$$

$$F^4_{l_5} = \text{Conv}(\text{Upsample2}(F^3_{l_6})), \quad (5)$$

$$F^4_{l_5} = F^3_{l_5} + F^4_{l_5}. \quad (6)$$

We show in Table 2 that the layered architecture of each sub-encoder is different. In the first sub-encoder, we have 11 MBConv blocks, whereas in the successive sub-encoders, we have 7, 5, and 3 MBConv blocks, respectively. The reason for reducing the number of MBConv blocks in subsequent encoders is that the repetition of shallow and intermediate stages does not contribute significantly to context assimilation, as these stages contain more spatial details than contextual information. While the repetition of intermediate stages is reduced, reusing deep features in the successive sub-encoder is strategically increased to enhance context assimilation.

Furthermore, Table 2 also illustrates that by increasing the value of the depth multiplier M_d, the repetition of deep MBConv6 blocks is increased in the subsequent sub-encoder. This allows us to increase the depth of the model without creating additional stages. Compared to DFANet, our proposed multi-encoder design reuses deep semantic features more effectively and eliminates the need for FC attention modules, as it scales the feature maps at different levels through a specially designed multi-encoder network.

Figure 4(a) shows that the four sub-encoders produce rich semantic feature maps F_6, F_5, F_4, and F_3. Lateral connections between encoders at the same level are used to address the gradient vanishing problem. We downsample the feature map F_6 by a pooling operation to create an additional feature map F_7. At this stage, the feature map may lose the contextual details of tiny objects in the scene due to the smaller spatial dimensions; however, it retains the context of large objects. To make the model more efficient, we utilize a simple pooling operation to create the feature map F_7 as this operation does not add any additional parameters. The rich features F_7 to F_3 will then be utilized by our decoder network.

B. DECODER NETWORK

Like most semantic segmentation models, our decoder is deployed to produce an output of the same size as the input by employing a series of upsampling techniques. Rich semantic features at different scales from the output of the encoder network need to be fused together at different levels. Similar to feature reusing in the encoder network, fusing features from multiple paths in both directions enhances the ability of object localization in the scene. Figure 4(b) displays the complete architecture of our proposed decoder network. Next, we describe key innovative steps.

1) LOCAL AND GLOBAL CONTEXT AGGREGATION MODULE

To motivate our proposed design, we first make an important observation from the literature [4], [46], [47] that aggregating features at different scales enhances the entire feature hierarchy with accurate object localization in the scene. Therefore, we introduce a novel component called the Local and Global Context Aggregation (LGCA), for this purpose. The blue dotted box in Figure 4(b) displays the complete architecture of LGCA. First, it takes deep and intermediate features (F_3 to F_7) produced by the multi-encoder network described previously. We adopt a channel reduction mechanism in which all feature maps at various stages will be filtered by a standard point-wise convolution layer to generate features with reduced channel $P_{l_i} = \text{Conv}(F_{l_i})$. It is required to provide similar depth of each feature map before being fused with each other. Moreover, by reducing the depth of the deep feature maps, the model complexity is also reduced. Later on, high-level semantic features are propagated downward through a top-down path to boost the semantic representation and improve multi-scale in-variance. Equation 7 shows that before fusing a higher-level feature P_{l_i} with a previous level feature map $P_{l_{i-1}}$, P_{l_i} is bi-linearly upsampled. This top-down path provides the first decoder path which helps in achieving context assimilation in the feature hierarchy. However, the spatial details of local feature maps need to be added with rich semantic details of the global feature maps for better object localization. This necessitates one bottom-up path to send the accurate localization signals from a lower level to

a higher level. Equation 8 shows that a low level feature P_{l_i-1} is down-sampled before it gets added with the next higher-level feature map P_i. The downward arrows define top-down path and the upward arrows signify the bottom-up path in Figure 4(b):

$$P_{l_{i-1}} = P_{l_{i-1}} + \text{Upsample2}(P_{l_i}), \quad (7)$$

$$P_{l_i} = P_{l_i} + \text{MaxPooling}(P_{l_{i-1}}). \quad (8)$$

Every downsampling or upsampling operation typically causes a loss of spatial details. To minimize this loss, we deploy a separable convolution (SC) block after every pooling operation. This block contains a depth-wise separable convolution (DSConv) layer followed by a batch normalization layer (BN). DSConv first filters the feature map along its depth, then deploys a point-wise standard convolution for better refinement. By standardizing the output of the DSConv layer, the BN layer enhances the independent learning ability of every layer of the network. We set the dilation rate to 2 for the DSConv layer in order to achieve a better receptive field while refining the feature maps.

After the bottom-up path, we finally introduce another top-down path for final context engrossment. Similar to feature reuse in the multi-encoder, we aggregate semantic features through multiple channels for better context accumulation and accurate object localization. Some skip connections among the paths of the decoder are introduced to address the degradation problem and help the loss function converge quickly. At the end of the second top-down path, we receive a semantically rich feature map, which is upsampled 2 times to fuse with the coarse local feature map F_2. This completes the pipeline of LGCA.

2) CLASSIFIER

This final module of our proposed decoder network assigns a class label to every pixel based on their contextual details. The literature has shown that adding a few layers in the classifier module supplements model performance [24], [39]. Hence, we employ two depth-wise separable convolution layers, one standard convolution layer, one upsample layer, and one softmax layer. In each DSConv layer, we use a 3×3 filter with a dilation rate of 2 as this provides a better receptive field while refining the feature map. Since the input feature map in the classifier module has 64 channels, the choice of DSConv layers helps reduce the number of parameters and GFLOPs. We implement one standard convolution layer for the finest segmentation, setting the number of channels to be the same as the number of classes in the target dataset. The spatial dimensions of the feature map in the classifier module are one-fourth of the original input. To ensure equal height and width, we utilize a bilinear upsampling layer that upscales the feature map by 2^2 times. Additionally, we employ a Dropout layer to address model overfitting. Finally, the softmax activation function is used to assign a class label to each individual pixel.

IV. EXPERIMENTS

As this work targets resource-constrained mobile devices, so we mainly compare the proposed model's performance with the existing real-time semantic segmentation models having a less than 5 M model parameters.

A. DATASETS

To benchmark our proposed model against others, we conducted extensive experiments on structured and unstructured public datasets. We strictly followed the evaluation protocols of these datasets for training, validation, and testing.

1) STRUCTURED DATASETS

Cityscapes [28] is the most widely used dataset for semantic segmentation. It provides urban street scene images at a resolution of 1024×2048, where objects are classified into 35 classes and grouped into 8 different categories. Following the protocols used in the literature for Cityscapes, we used 19 classes for pixel annotations. The dataset consists of around 5,000 finely annotated images, out of which 2,975 images are used for training, 500 samples are used for validation, and the remaining 1,525 images are used for testing. However, annotations for the test set are not provided with the dataset. To evaluate our model on the test set, we submitted the test results of the proposed model to the Cityscapes online evaluation server, and the results were published on the server.

CamVid [25] is a small structured dataset that provides 267 images for training, 101 for validation, and 233 for testing. Consistent with the evaluation protocols in the literature, we used only 11 fine-tuned classes out of the 32 classes in the dataset. To improve performance on CamVid, we utilized transfer learning by pre-training the model on the Cityscapes dataset with appropriate class mapping between the two datasets.

Similarly to Cityscapes, the BDD100K [26] and KITTI [27] datasets use the same class labeling technique (19 classes) for training and testing. Due to this compatibility in class labeling, we can use Cityscapes pre-trained weights to train the model with BDD100K and KITTI datasets. BDD100K provides a total of 10,000 images, out of which 7,000 images are used for training, 1,000 for validation, and the remaining 2,000 images for testing. Fine-grained annotations are provided only for the training and validation sets. The original input resolution of this dataset is 720×1280 pixels. On the other hand, KITTI is a small urban street scene dataset that provides only 200 training images with fine-tuned annotations and 200 test images without annotations. Each input image has a resolution of 375×1280 pixels. Similar to Cityscapes, KITTI also provides an online evaluation server for the test set. We submitted the test set results of our proposed model to the KITTI evaluation server to obtain the test results.

2) UNSTRUCTURED DATASET

The four datasets mentioned above primarily focus on urban street scenes captured in western countries, such as Europe or the USA, where the road environment is well-structured and there are fewer variations in the objects present. However, such well-defined traffic environments are not representative of the road conditions in Asian countries, such as India. To evaluate the performance of the proposed model in unstructured road conditions, we trained the model using the IDD-lite (Indian Driving Dataset lite version) [48]. This dataset consists of 1,404 urban and rural training images, 204 validation samples, and 404 test samples, each with a resolution of 227 × 320. The dataset divides the entire object space into seven classes: drivable, non-drivable, living things, vehicles, roadside objects, far objects, and sky. We reported the performance of the proposed model on each of these classes.

Furthermore, we also trained the proposed model using IDD part 1 and part 2, which contain approximately 14,027 training samples and 2,036 validation samples. Similar to IDD-lite, we reported the performance of the proposed model on the seven classes of both IDD and Cityscapes validation sets.

B. IMPLEMENTATION DETAILS

All compared models were trained on a server equipped with three Nvidia GeForce TITAN RTX GPUs, each with 24GB of memory. For effective utilization of all GPUs in data-parallel distributed training, we used the `horovod` framework [49]. The software components included CUDA 10.2 for parallel processing, `tensorflow` 2.1.0, and `keras` 2.3.1. We employed the polynomial learning rate strategy, with a base rate of 0.045 and power of 0.9. Using a polynomial scheduler, we found the optimal learning rate at the steepest slope of the training loss vs. learning rate plot for 5 epochs. We used the distributed synchronous stochastic gradient descent (SGD) optimizer, which divides SGD mini-batches over a pool of parallel GPUs to find the best learning rate. Following [50], we also employed a gradual warm-up strategy in the `horovod` distributed framework to overcome optimization challenges, especially in the early stages of the training process.

To improve training, we employed various on-the-fly data augmentation techniques such as resizing, cropping, clipping by value, horizontal and vertical flipping, adjusting brightness, saturation and contrast of the input images to increase the effective size of the training set. We also employed different regularization techniques to address model over-fitting, such as ℓ_2 regularisation for all top layers and a dropout layer with a dropout rate of 0.3.

C. ABLATION STUDY

In this ablation study, we aim to demonstrate the importance of each component of our proposed model in achieving the best segmentation performance. Firstly, we highlight the significance of the multi-encoder design. To do this, we initially trained the proposed model with only the first sub-encoder (as described in Table 2). Subsequently, we progressively added the second, third, and fourth sub-encoders. The validation results reported in Table 3 were obtained after training the model on the Cityscapes training set for 500 epochs at the full input resolution of 1024 × 2048 px. In the results section, we reported the best validation and test mIoU achieved by our proposed model after fine-tuning the model and training it for a large number of epochs. Initially, we did not utilize multiple paths at the decoder side and only employed the first top-down path of the LGCA module to fuse features at different levels. Therefore, the first five rows of Table 3 do not include LGCA. The table clearly demonstrates that with the addition of each sub-encoder, the model's performance noticeably improves and reaches a validation mIoU of 70.7% after 500 epochs.

TABLE 3. Results of ablation study.

Encoder	ASPP	LGCA	Param. (M)	GFLOPs	Val. mIoU (%)
1	-	-	0.76	18.8	60.1
1,2	-	-	1.44	24.0	65.2
1,2,3	-	-	2.11	26.2	69.4
1,2,3,4	-	-	2.68	27.0	70.7
1,2,3,4	✓	-	2.69	27.0	69.9
1,2,3,4	-	✓	2.72	31.2	71.8

The sign '-' means ASPP/LGCA is not used and '✓' means ASPP/LGCA is used in the pipeline.

We also explored the use of ASPP (Atrous Spatial Pyramid Pooling) [13] for feature scaling on top of the fourth encoder. ASPP is known to filter the feature map with various sizes of receptive fields and has the potential to improve segmentation performance. However, we observed a slight drop in performance. This could be attributed to the fact that the model's backbone produces a low-resolution feature map that is 2^6 times smaller than the original input size. Due to this performance reduction, we did not incorporate ASPP in our model design. Furthermore, the different layered architectures of each sub-encoder already provide feature scaling capability, making ASPP unnecessary. The last row of Table 3 demonstrates that with the inclusion of LGCA on top of the multi-encoder network, our proposed model, called **MCANet**, achieves a validation mIoU of 71.8% after 500 epochs, while having only 2.72 million parameters and 31.2 GFLOPs. If we upsample the global feature map by 2^3 times at the decoder end, the GFLOPs count would be reduced to 27.5 at full input resolution. However, this may lead to boundary degeneration effects in the output. Therefore, we perform upsampling of the global feature map at two stages: the first upsampling by 2^1 times occurs inside LGCA, and the second upsampling by 2^2 times occurs inside the classifier module (as shown in Figure 4(b)).

From Table 3, it is evident that the model's performance improves with the addition of each sub-encoder and LGCA

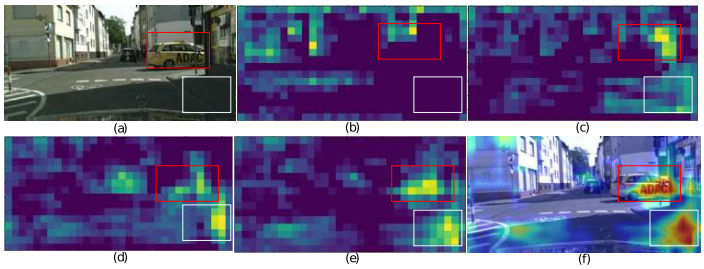

FIGURE 5. Illustration of feature similarities of a group of pixels using score map. Bright and hotter color means more similar feature among the pixels. Weights of intermediate Conv layers are used to produce score map at various levels for an (a) input image. (b) Score map after first sub-encoder, (c) Score map after second sub-encoder, (d) Score map after third sub-encoder, (e) Score map after forth sub-encoder, (f) Final score map overlaying with the input image.

TABLE 4. Class-wise MCANet performance on Cityscapes validation and test sets.

dataset	Road	S.walk	Build.	Wall	Fence	Pole	T.light	T.sign	Veg.	Terrain
Validation set	**98.7**	85.3	**93.8**	51.6	52.8	64.0	69.1	73.9	**94.1**	71.2
	Sky	Person	Rider	Car	Truck	Bus	Train	M.cycle	Bicycle	**mIoU**
	96.2	81.8	61.1	**95.6**	59.2	77.2	65.3	58.5	71.3	**74.8**
dataset	Road	S.walk	Build.	Wall	Fence	Pole	T.light	T.sign	Veg.	Terrain
Test set	**98.3**	84.2	**92.0**	49.5	51.6	62.1	67.8	73.2	**92.6**	70.3
	Sky	Person	Rider	Car	Truck	Bus	Train	M.cycle	Bicycle	**mIoU**
	95.3	81.4	59.2	**94.6**	57.8	75.9	64.2	56.3	69.1	**73.4**

TABLE 5. Category-wise MCANet performance on Cityscapes validation and test sets.

dataset	Flat	Construction	Object	Nature	Sky	Human	Vehicle	mIoU
Validation set	**98.7**	**93.2**	69.7	**93.5**	**92.3**	82.1	**94.6**	**89.2**
Test set	**98.5**	**92.2**	68.4	**95.3**	**91.96**	82.3	**93.9**	**88.9**

module. To visually demonstrate this improvement, we generated score maps (before the softmax function) at various stages using feature maps from different levels, and these score maps are presented in Figure 5. In each score map, pixels with similar features are highlighted using a hotter color. We used feature maps at 16×32 resolution after the first, second, third, fourth sub-encoders, and after LGCA (as depicted in Figure 4). In Figure 5, we highlighted two main sections (car and pedestrian) in the input image using red and white boxes. It clearly illustrates that with the successive addition of each sub-encoder and LGCA module, the model becomes capable of identifying more pixels with similar features.

Therefore, both the quantitative and qualitative studies provide clear evidence of the effectiveness of the multiple sub-encoder design for feature extraction at various levels and the use of LGCA for constructing the segmented output using the extracted features from different levels.

D. MODEL EVALUATION

The proposed model is evaluated on the four urban street scenes datasets. Following the literature and evaluation servers, we present the following metrics: class and category-wise mean Intersection over Union (IoU), mean instance-level Intersection over Union (iIoU), model parameters, GFLOPs and Frame Per Second (EPS). As the proposed backbone is designed from scratch, so we did not use any existing pre-trained weight to train the model with Cityscapes. Moreover, We did not train the proposed backbone with ImageNet [29] dataset like other exiting models.

1) PERFORMANCE ON CITYSCAPES

We trained the proposed model on the Cityscapes dataset for 1000 epochs with a batch size of 4 on each GPU. During the performance evaluation on the validation set, we utilized the training set. However, to improve the accuracy on the test set, we merged both the training and validation sets. Additionally, we included additional coarse images from Cityscapes for a small number of epochs, which resulted in a slight improvement of only 0.4% in test performance. Table 4 presents the class-wise performance of the model on the Cityscapes validation and test sets. It is observed that the proposed model performed exceptionally well on the top 5 classes (including road, building, vegetation, sky, and car), with accuracy above 90% on both the validation and test sets. Overall, MCANet

TABLE 6. Performance evaluation of different models on Cityscapes validation and test set.

Type	Model	Parameter (M)	GFLOPs	Val. Class mIoU(%)	Val. Cat. mIoU (%)	Test Class mIoU(%)	Test Class iIoU(%)	Test Cat. mIoU (%)	Test Cat. iIoU (%)	FPS
Real time	SwiftNetRN-18†‡ [38]	11.8	114	72.2	-	75.9	-	-	-	134.9
	STDC1†‡ [51]	8.4	0.8	74.5	-	75.3	-	-	-	250.4
	DFANet†‡ [42]	7.8	3.4	71.9	-	71.3	-	-	-	160
	ICNet† [22]	6.7	28.3	-	-	69.5	-	-	-	30.5
	BiSeNet†‡ [21]	5.8	14.8	-	-	68.4	-	-	-	105.8
	MGSeg(S.Net)†‡ [37]	4.5	16.2	-	-	72.7	-	-	-	84
	Fast-SCNN* [39]	1.2	14.9	63.3*	82.2*	68.0	37.9	84.7	63.5	285.8
	FANet* [40]	1.1	11.4	65.9*	83.6*	64.1	33.2	83.1	61.1	78
	ContextNet* [24]	1.0	37.5	60.4*	81.5*	66.1	36.8	82.8	64.3	136.2
	ENet [19]	0.4	3.8	-	-	58.3	34.4	80.4	64.0	76.9
	QNet-attention [36]	0.2	19.8	-	-	49.2	-	70.1	-	18.2
Real time	MCANet	2.7	31.2	74.8	89.2	73.4	45.8	88.9	72.8	269

The '†' sign indicates that the test result is not available at the Cityscapes evaluation server. The '‡' sign indicates that the model is pre-trained using the ImageNet dataset. The '*' sign indicates that the existing models mentioned in the table were trained by the study, and their corresponding results are reported in the table.

TABLE 7. GFLOPs and FPS at different input resolution.

Input size	256 × 512		384 × 768		512 × 1024		768 × 1536		1024 × 2048	
Model	GFLOPs	FPS	GFLOPs	FPS	GFLOPs	FPS	GFLOPs	FPS	GFLOPs	FPS
MCANet	2.0	432	4.4	392	7.8	269	17.6	129	31.2	75
FANet	0.7	309	1.6	141	2.9	78	6.4	35	11.4	22
Fast-SCNN	1.2	494	2.6	231	4.6	124	10.4	58	18.4	34
ContextNet	2.4	397	5.3	182	9.4	101	21.1	44	37.5	27

achieved a validation mIoU of 74.8% and a test mIoU of 73.4% on the Cityscapes dataset. The performance on the test set was independently evaluated by the Cityscapes evaluation server, and the results are available on the server.

The 19 classes of Cityscapes are categorized into 7 categories, and the corresponding category-wise mIoU is presented in Table 5. It demonstrates that **MCANet** performed exceptionally well in 5 out of the 7 categories. The average performance in the object and human categories is relatively lower due to the limited occurrence of classes within these categories in the entire dataset. This non-uniform distribution of classes is a common challenge across existing models. Overall, the proposed model achieved an impressive category-wise mIoU of almost 89% on both sets.

Performance comparison To illustrate the effectiveness of our proposed model, we compared its performance with existing real-time semantic segmentation models. It is generally acknowledged in the literature that offline models have a large number of parameters due to their deep network architecture, while real-time models have significantly fewer parameters. Therefore, in Table 6, we did not include the performance of existing offline models to ensure a meaningful comparison. Typically, offline semantic models have over 40 million parameters and achieve mIoU values of 80-84% on the Cityscapes test set. For example, DeepLabV3+ [13] and PSPNet [14] achieve test mIoU values of 82.1% and 81.2%, respectively, with 43 and 250.8 million parameters. In contrast, most existing real-time semantic segmentation models have less than 10 million parameters and achieve test mIoU values of 68-72% on Cityscapes. For instance, STDC1 [51], MGSeg [37], DFANet [42], ICNet [22], and BiSeNet [21] achieve test mIoU values of 75.3%, 72.7%, 71.3%, 69.5%, and 68.4%, respectively. However, these models still have moderately large parameter counts ranging from 4.5-8.4 million. SwiftNet [38], which uses a pre-trained ResNet-18 (RN18) as a backbone, achieves a test mIoU of 75.9% on Cityscapes with 11.8 million parameters. While all these models demonstrate good accuracy, they still have a moderately large number of parameters. On the other hand, models like ENet [19], ContextNet [24], Fast-SCNN [39], and FANet [40] have 0.4-1.2 million parameters and achieve test class mIoU values of 58-68%. These models are more efficient in real-time environments but their performance lags behind moderately large real-time semantic models by 4-6%. Striking a balance between model size and performance, our proposed model, **MCANet**, achieves 74.8% and 73.4% class mIoU on the Cityscapes validation and test sets, respectively, with only 2.7 million parameters. This clearly demonstrates the superior performance of our model on the Cityscapes dataset.

For consistency, we decided to replicate a few existing real-time models based on publicly available implementations on GitHub to ensure a more meaningful comparison. These models are marked with an asterisk '*' sign in the following tables. We trained these models under the same system configurations with the full input resolution. The results obtained from our experiments on the Cityscapes validation set are presented in Table 6 and are marked with an asterisk

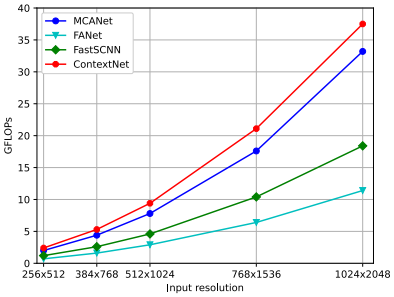

FIGURE 6. The plot Input size vs GLFOPs illustrates that with an increase in input resolution, the GFLOPs count of all models also increases.

'*' sign. Our experimental results for the existing models may differ from the actual literature, but it provides a fair comparison based on the same system settings.

Table 6 provides the class and category-based mIoU results for different existing models on the validation and test sets of Cityscapes. The mean iIoU, as provided by the Cityscapes evaluation server, is also included, along with the model parameters and GFLOPs count. However, some real-time semantic segmentation models did not publish their results on the Cityscapes evaluation server, resulting in unavailable iIoU values denoted by the sign "-". Our result on the Cityscapes test set is available on the benchmark server.

While class mIoU is the primary metric for comparison as it comes from the Cityscapes evaluation server, we also discuss GFLOPs count for completeness. However, it is worth noting that GFLOPs count is not an optimal metric as it depends on the model size and input resolution. The increase in input resolution leads to a somewhat polynomial increase in GFLOPs count, as shown in Figure 6. This has resulted in inconsistent GFLOPs counts being reported in previous work. For example, ENet has 3.8 GFLOPs at a 360 × 360 input resolution with 0.4 million parameters, while STDC1 [51] claims 0.8 GFLOPs at a 224 × 224 input resolution with 8.4 million parameters.

Since we trained three existing models (FANet [40], Fast-SCNN [39], ContextNet [24]) using the Cityscapes dataset under the same system configuration, we measured the GFLOPs of these models at different input resolutions. The results are presented in Figure 6 and Table 7. It can be observed that as the input resolution increases, the GFLOPs count of all models also increases, with ContextNet [24] having the highest GFLOPs count. Our proposed model produces 31.2 and 2.0 GFLOPs at input resolutions of 1024 × 2048 and 256 × 512, respectively.

Another commonly mentioned metric in semantic segmentation evaluation is frames per second (FPS). However, it is evident that FPS is highly dependent on hardware and input resolution. For the sake of completeness, we also discuss this metric here. To ensure a meaningful comparison, we measured the FPS of all four trained models listed in Table 7 under the same system configuration at different input resolutions and presented the experimental results. To measure FPS, we first converted the TensorFlow model to a TensorRT optimized model and then used a single Tesla T4 GPU with 16GB memory. We used a batch size of 4 and averaged the FPS value over 10 iterations. It is evident from the results that Fast-SCNN [39] achieves higher FPS compared to the other models at different input resolutions. We acknowledge that our own measurements may differ from the figures published in the original papers, possibly due to variations in hardware and measurement methods. However, what is more important is the relative performance based on the most intuitive way to measure the overall computation of the entire pipeline.

When compared to our proposed model, all three models listed in Table 7 have approximately 2-3 times fewer parameters. They employ a computationally cheaper method for upsampling the global feature map in the decoder, which results in boundary degradation in the output and overall lower segmentation performance. Our effective upsampling solution, as described earlier, introduces additional computational cost, leading to a lower FPS. However, we believe that such a trade-off is worthwhile for significantly improved segmentation quality. In Table 6, we reported the FPS of our proposed model at an input resolution of 512 × 1024.

2) PERFORMANCE ON CamVid DATASET

We also trained our proposed model along with a few existing semantic segmentation models with CamVid dataset and present the results in Table 8. To ensure tensor size compatibility, we used an input size of 640 × 896 px instead of the full input resolution of 720 × 960 px. Table 8 clearly illustrates that our proposed model **MCANet** achieved the state-of-the-art (SOTA) performance on the CamVid validation and test sets among the existing real-time semantic models. It achieved 81.4% and 80.2% validation and test mIoU, respectively, which is even higher than many existing offline models such as DeepLab [11] and PSPNet [14]. The dual deep model DeepLabV3+ with SDCNetAug [52] achieved the SOTA performance (81.7%) on the CamVid test set due to its large backbone, joint strategies, and large synthetic datasets. In comparison, real-time models such as STDC1 [51] and MGSeg [37] achieved 73.0% and 72.7% test mIoU on the CamVid set, respectively. Literature [37] did not report the performance of the smaller variant of MGSeg (ShuffleNetV2) on the CamVid dataset. Hence, we compared the proposed model's performance with the higher variant of MGSeg (ResNet-18) (refer to Table 8). In terms of size, both of these models (MGSeg (R18) and STDC1) are 3 to 5 times bigger than our proposed model. Despite being a smaller network, our proposed **MCANet** achieved more than 7% test accuracy on the CamVid dataset. Thus, Table 8 shows the superior performance of our proposed model among real-time semantic models.

TABLE 8. Performance evaluation on CamVid validation and test sets.

Model	Input size	Val. class mIoU(%)	Test class mIoU (%)	FPS
DeepLabV3Plus SDCNetAug [52]	720×960	-	81.7	-
PSPNet [14]	-	-	69.1	-
DeepLab [13]	720×960	-	61.6	5
STDC1 [51]	720×960	-	73.0	197.6
MGSeg (R18) [37]	720×960	-	72.7	127
ICNet [22]	720×960	-	67.1	28
FANet* [40]	640×896	73.8	66.9	71
Fast-SCNN* [39]	640×896	73.3	66.7	120
ContextNet* [24]	640×896	69.6	65.8	96
BiSeNet [21]	720×960	-	65.6	-
DFANet [42]	720×960	-	64.7	120
ENet [19]	360×480	-	51.3	-
MCANet	640×896	81.4	80.2	31

Existing Models marked with the '*' sign are trained by the study.

TABLE 9. Performance evaluation on validation set of BDD100K dataset.

Model	Param. (M)	GFLOPs	Class mIoU (%)	FPS
HANet-R101	64.2	2137.8	64.8	-
HANet-MV2 [15]	14.8	142.7	58.9	-
FANet* [40]	1.1	5.4	50.0	40
Fast-SCNN* [39]	1.2	8.6	47.9	70
ContextNet* [24]	1.0	17.5	44.5	56
MCANet	2.7	20.7	58.8	28

Existing Models marked with the '*' sign are trained by the study.

TABLE 10. Performance evaluation on test set of KITTI.

Model	Class mIoU (%)	Class iIoU (%)	Cat. mIoU (%)	Cat. iIoU (%)
DeepLabV3Plus + SDCNetAug [52]	72.8	48.7	88.9	75.3
SGDepth (Seg.) [53]	53.0	24.4	78.7	55.9
SDNet [54]	51.1	17.7	79.6	50.5
PAG [55]	47.9	17.9	78.1	49.2
MCANet	58.5	24.0	83.0	54.1

3) PERFORMANCE ON BDD100K

Table 9 displays models performance on the BDD100K dataset. We trained the model with 768 × 1280 input resolution for better compatibility with tensor dimensions. To improve model performance on BDD100K dataset, we used Cityscapes pre-trained weight. Due to the diverse and complex nature of this data set, not many existing models are trained with this dataset. The only work we could find is [15] which introduced two different variants of HANet- HANet with MobileNetV2 (MV2) as backbone and HANet with ResNet-101 (R101) as backbone, both of which are clearly off-line models. We present both the variants' performance along with the proposed model. HANet R101 variant produces the SOTA result (64.8%) on BDD100K validation set while having as many as 64.2M parameters and 2137.8 GFLOPs. The smaller variant of HANet (MV2) which has 14.8M parameters, generates 58.9% validation mIoU. In comparison, the proposed model generates 58.8% validation mIoU while having 5 to 24 times less parameters than the both variants of HANet. It clearly shows the superior performance of the proposed model on BDD100K dataset. We also trained few existing models (marked by * sign) with BDD100K dataset and presented the results in Table 9. It can be observed that among the real-time semantic models, the proposed model produces the SOTA result on BDD100K validation set.

TABLE 11. Model performance on IDD-lite validation set.

Model	Val mIoU (%)	Param. (M) (%)	GFLOPs	FPS
$DeepLabV3+$ (ResNet50)	64.3	26.7	28.2	470
UNet (ResNet50)	68.6	157.3	50.8	434
Eff-UNet (E.Net B5)	70.7	34.0	5.3	323
Eff-UNet (E.Net B7) [56]	73.8	72.8	10.5	365
MCANet	73.8	2.7	0.7	494

4) PERFORMANCE ON KITTI

We followed the same training protocol as BDD100k [26] to train our proposed model with the KITTI [27] dataset, and the results on the KITTI test set are presented in Table 10. The KITTI dataset is primarily used for stereo, visual odometry, and depth analysis. Therefore, we did not find any existing real-time semantic segmentation models that were specifically trained and tested on the KITTI fine-tune dataset and had their results submitted to the evaluation server. We came across a few works from the KITTI server, such as DeepLabV3Plus+SDCNetAug [52], SGDepth [53], SDNet [54], and PAG [55], which primarily focused on depth analysis. Although these models incorporate a semantic head to enhance the model's performance in addition to the depth analysis decoder head. Among these models, SGDepth [53] achieves relatively better results with a class mIoU of 53.0% on the KITTI test set, followed by SDNet [54] with 51.1%. The current state-of-the-art result on the KITTI test set is achieved by DeepLabV3Plus+SDCNetAug [52], which is a combination of multiple models. It utilizes a joint video prediction model to augment the training sets for robust semantic segmentation, leverages a deep semantic model (DeepLabV3Plus) for feature extraction, and applies a boundary label relaxation technique to reduce noise at the object edges. Due to the collective efforts of multiple models and the presence of large synthetic training sets, this model achieves a class mIoU of 72.8% on the KITTI test set. In comparison, our proposed lightweight single model is significantly smaller and specifically designed for scene parsing. It achieves a class mIoU of 58.5% and a category (Cat.) mIoU of 83.0% on the KITTI test set, setting the state-of-the-art performance in the real-time semantic segmentation category. The results of MCANet were independently generated by the KITTI evaluation server. For tensor dimension compatibility, we used an input resolution of 384×1280 to train the model on the KITTI dataset. All the models listed in Table 10 were pre-trained with the Cityscapes dataset.

TABLE 12. Class-wise model performance on IDD-lite validation set.

Model	Dataset	Drivable	Non-drivable	Living things	Vehicles	Roadside objects	Far Objects	Sky	mIoU
Eff-Unet	IDD-lite	94.9	50.1	62.0	81.3	55.0	77.5	95.6	73.8
MCANet	IDD-lite	94.5	48.1	59.7	81.5	56.8	80.0	96.0	73.8
MCANet	IDD	95.4	50.3	63.7	84.2	58.5	82.0	96.2	75.6
MCANet	Cityscapes	91.4	57.5	54.6	80.7	40.0	88.8	89.6	71.8

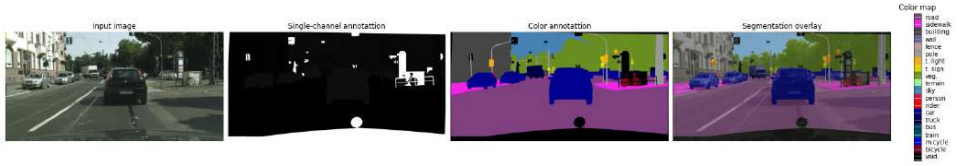

FIGURE 7. Colour mapping of Cityscapes dataset.

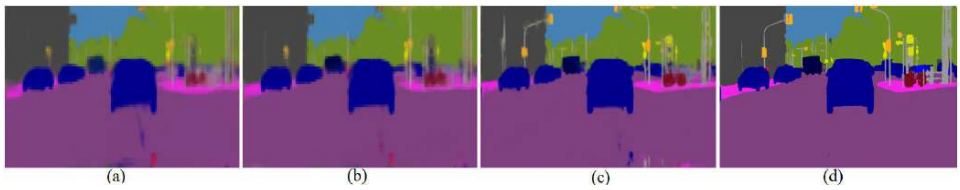

FIGURE 8. Output produced by (a) ContextNet, (b) FANet, (c) FAST-SCNN, (d) MCANet using Cityscapes validation image.

FIGURE 9. Output produced by MCANet using Cityscapes test set samples.

5) PERFORMANCE ON IDD-LITE

he IDD-lite dataset is primarily designed for resource-constrained devices that lack sufficient hardware resources to train models with large input resolutions. To ensure tensor size compatibility, we trained our model with a 256 × 384 input resolution. The performance of different existing models on the IDD-lite validation set is presented in Table 11. Among the existing models, Eff-UNet (E.Net B7) [56] achieves the state-of-the-art performance on the IDD-lite validation set and won the first prize in the IDD-lite segmentation challenge held in 2019. Eff-UNet [56] utilizes a large feature extractor called EfficientNet-B7 (E.Net B7) [57], which has 66M parameters and 37 GFLOPs at a 224 × 224 input resolution. However, despite the IDD-lite dataset targeting resource-constrained embedded devices, the evaluated existing models on this dataset are still too large and computationally inefficient for mobile devices.

Table 11 presents the results of the top-performing existing models on the IDD-lite dataset, including their parameters and GFLOPs at a 128 × 256 input resolution. It is evident that all these existing models have a large number of parameters and GFLOPs, making them impractical to run on resource-constrained embedded devices, especially at higher input resolutions. In contrast, our proposed MCANet is 10 to 58 times smaller than all the listed existing models while achieving a state-of-the-art result (73.8% mIoU)

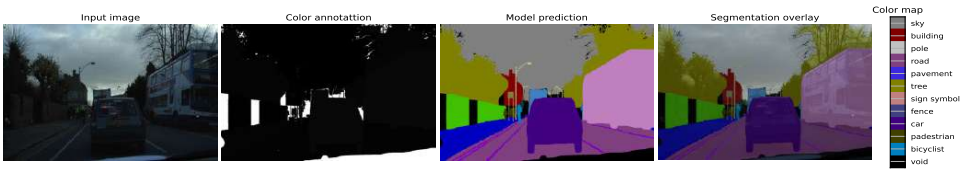

FIGURE 10. Colour mapping of CamVid dataset.

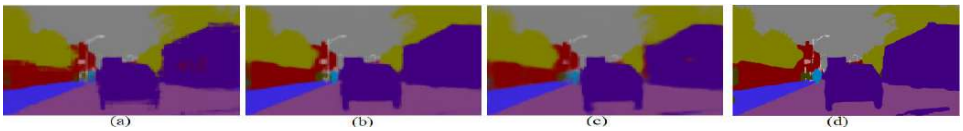

FIGURE 11. Output produced by (a) ContextNet, (b) FAST-SCNN, (c) FANet, (d) MCANet using CamVid validation image.

FIGURE 12. Output produced by MCANet using CamVid test set samples.

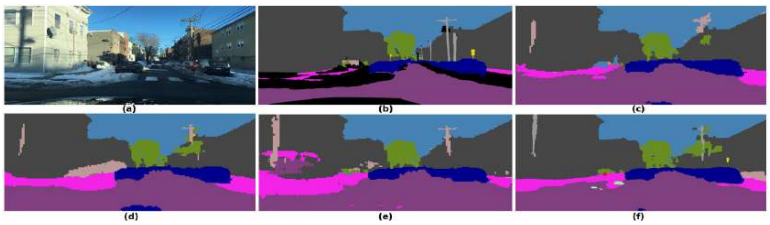

FIGURE 13. Models prediction on BDD100K validation set. (a) RGB input, (b) Coloured annotation, (c) ContextNet, (d) Fast-SCNN, (e) FANet, (f) MCANet.

on the IDD-lite validation set, similar to Eff-UNet (E.Net B7) [56]. Table 11 also provides information on the model parameters, GFLOPs, and FPS count. It is clear that our proposed MCANet is more efficient compared to all the existing models, as it processes a higher number of frames (494) per second while significantly reducing computational usage by reducing the number of parameters and GFLOPs.

Table 12 presents the class-wise mIoU performance of Eff-Unet (E.Net B7) [56] and our proposed MCANet. Additionally, we report the performance of our proposed model on seven classes of the IDD (part 1 and part 2) [58] and Cityscapes [28] datasets. Our model achieves 75.5% and 71.6% mIoU on the IDD and Cityscapes datasets, respectively.

6) QUALITATIVE RESULTS AND ANALYSIS

In this section, we demonstrate the quality of the output produced by our proposed model and compare it with other models. Figure 7 and 10 display the annotation and color map used for the Cityscapes and CamVid datasets, respectively.

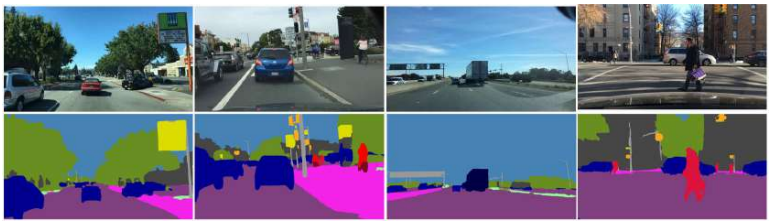

FIGURE 14. Output produced by MCANet using BDD100K test set samples.

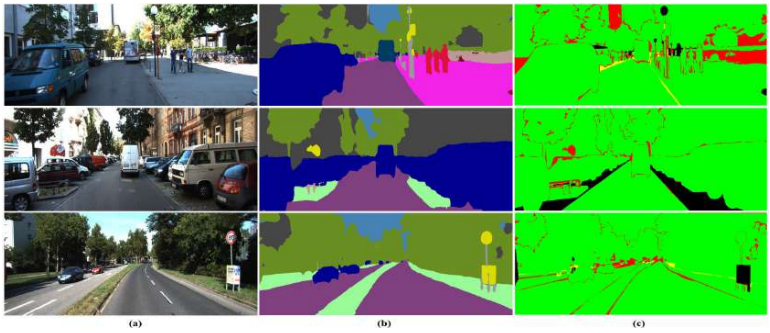

FIGURE 15. Output produced by MCANet using KITTI test set samples.

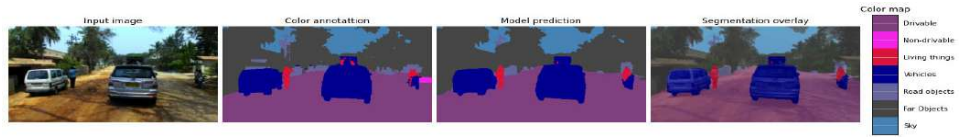

FIGURE 16. Color map of IDD lite dataset and model prediction using validation sample.

FIGURE 17. Models predictions using Cityscapes and IDD samples on 7 classes. (a) Cityscapes input, (b) Cityscapes prediction, (c) IDD input, (d) IDD prediction.

Nineteen color codes are used for Cityscapes, while eleven color codes are used for CamVid. The BDD100K and KITTI datasets follow the same color codes as Cityscapes. The void class is excluded for all datasets.

Figure 8 showcases the corresponding segmented output of the input images depicted in Figure 7. The outputs generated by ContextNet, Fast-SCNN, and FANet exhibit a boundary degradation effect due to the 2^3 times upsampling at the end of the decoder. On the other hand, our proposed model MCANet produces outputs with sharp and clear edges for each object in the scene. The results in Figure 9 demonstrate that our model accurately positions tiny classes such as poles,

traffic lights, and traffic signs in the test samples, without overlooking them amidst the larger classes

Likewise, Figure 11 displays the output produced by different models on selected CamVid images. In contrast to the original annotation of CamVid, we formed some super classes by merging related classes. For instance, we grouped car, truck, bus, and caravan together and formed a single class called "car." Thus, the bus is represented by the same color as the car, as can be observed in Figure 11. Figure 12 displays the output generated by the proposed model using selected test samples from the CamVid dataset. In line with the quantitative results presented in Table 8, Figure 11 also confirms the model's superiority over other models.

Figure 13 shows the segmented output produced by different models using the BDD100K validation set. Classes such as ice and car hood, which are defined by the black color in the colored annotation (Figure 8(b)), are ignored during training of the model. As a result, pixels that belong to ignored classes are assigned the color of the neighboring classes. This does not affect the model's performance, as these pixels are completely disregarded when calculating mIoU. By inspecting all the output, it can be clearly seen that the quality of the output produced by the proposed model is much better than other three models in Figure 13. In order to provide a better view of different scenes that contain tiny objects, we also present the output produced by the proposed model using BDD100K test set in Figure 14. All of these figures clearly demonstrate the excellent performance of MCANet in the field of semantic segmentation.

Figure 15 shows the predictions of the proposed model on KITTI test set samples, as generated by the KITTI evaluation server. Along with the colored predictions, it also provides an error image for each sample. The second column of Figure 15 displays the proposed model's predictions, and the third column shows the corresponding error images. The color red in the error images indicates wrongly classified pixels. It is clear that pixels mostly at the boundaries of each object in the scene are incorrectly classified. However, the proposed model's object identification and overall segmentation demonstrate its excellent performance on the KITTI dataset.

Figure 16 displays the colour map of IDD-lite dataset and the output produced by the proposed model MCANet, using IDD-lite validation sample. Like the other datasets, the quality of the predicted output of the IDD-lite sample is good, and it justifies the quantitative result produced by the proposed model. In Figure 17, we also shows the model's predictions using Cityscapes and IDD samples. Like IDD-lite, seven classes are used.

V. CONCLUSION

To improve the performance of existing models in real-time semantic segmentation for resource-constrained applications and to reduce the performance gap between offline and real-time models, we introduced an efficient multi-encoder network that can handle high-resolution input images and produce competitive semantic segmentation results. The key innovative steps in our design are: a novel multi-encoder network with a dynamic layered structure for better capturing semantic information and sharing information more effectively across different scales; a new local and global context aggregation module for better semantic fusion in the output. Compared to existing real-time semantic segmentation models, our proposed model MCANet produces competitive performance in both structured and unstructured environments and sets a new benchmark on all the tested datasets while having only 2.7 M parameters. The effective design of our proposed multi-encoder fulfills the needs of feature scaling techniques and produces rich feature maps at different scales. By exploiting these feature maps, our proposed decoder assimilates contextual details in multiple paths and produces output with accurate object positioning in the scene. Although the addition of the LGCA module improves the localization of each object in the scene, it also slightly increases the processing time of each frame. Hence, in the future, we will try to optimize the design of the LGCA module to improve the model's FPS without sacrificing the model's performance. We will also exploit the design of the multi-encoder for instance and panoptic segmentation. we make an implementation of our model available at https://github.com/tanmaysingha/MCANet for reproducing the results presented in this work.

REFERENCES

[1] D. Feng, C. Haase-Schütz, L. Rosenbaum, H. Hertlein, C. Gläser, F. Timm, W. Wiesbeck, and K. Dietmayer, "Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 3, pp. 1341–1360, Mar. 2021.

[2] X. Chang, H. Pan, W. Sun, and H. Gao, "YolTrack: Multitask learning based real-time multiobject tracking and segmentation for autonomous vehicles," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 12, pp. 5323–5333, Dec. 2021.

[3] G. Benitez-Garcia, L. Prudente-Tixteco, L. C. Castro-Madrid, R. Toscano-Medina, J. Olivares-Mercado, G. Sanchez-Perez, and L. J. G. Villalba, "Improving real-time hand gesture recognition with semantic segmentation," *Sensors*, vol. 21, no. 2, p. 356, Jan. 2021.

[4] T. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. CVPR*, Jul. 2017, pp. 936–944.

[5] M. K. Noman, S. M. S. Islam, J. Abu-Khalaf, and P. Lavery, "Seagrass detection from underwater digital images using faster R-CNN with NASNet," in *Proc. DICTA*, Nov. 2021, pp. 1–6.

[6] N. Siddique, S. Paheding, C. P. Elkin, and V. Devabhaktuni, "U-Net and its variants for medical image segmentation: A review of theory and applications," *IEEE Access*, vol. 9, pp. 82031–82057, 2021.

[7] M. Z. Khan, M. K. Gajendran, Y. Lee, and M. A. Khan, "Deep neural architectures for medical image semantic segmentation: Review," *IEEE Access*, vol. 9, pp. 83002–83024, 2021.

[8] P. Yin, R. Yuan, Y. Cheng, and Q. Wu, "Deep guidance network for biomedical image segmentation," *IEEE Access*, vol. 8, pp. 116106–116116, 2020.

[9] D. Zeng, X. Chen, M. Zhu, M. Goesele, and A. Kuijper, "Background subtraction with real-time semantic segmentation," *IEEE Access*, vol. 7, pp. 153869–153884, 2019.

[10] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. CVPR*, Jun. 2015, pp. 3431–3440.

[11] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2018.

[12] S. Targ, D. Almeida, and K. Lyman, "ResNet in ResNet: Generalizing residual architectures," 2016, *arXiv:1603.08029*.

[13] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder–decoder with atrous separable convolution for semantic image segmentation," in *Proc. ECCV*, 2018, pp. 801–818.

[14] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proc. CVPR*, Jul. 2017, pp. 6230–6239.

[15] S. Choi, J. T. Kim, and J. Choo, "Cars can't fly up in the sky: Improving urban-scene segmentation via height-driven attention networks," in *Proc. CVPR*, Jun. 2020, pp. 9370–9380.

[16] Y. Yuan, X. Chen, and J. Wang, "Object-contextual representations for semantic segmentation," in *Proc. ECCV*. Cham, Switzerland: Springer, 2020, pp. 173–190.

[17] Y. Deng, "Deep learning on mobile devices: A review," *Proc. SPIE*, vol. 10993, May 2019, Art. no. 109930A.

[18] A. Ignatov, R. Timofte, A. Kulik, S. Yang, K. Wang, F. Baum, M. Wu, L. Xu, and L. Van Gool, "AI benchmark: All about deep learning on smartphones in 2019," in *Proc. ICCVW*, Oct. 2019, pp. 3617–3635.

[19] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "ENet: A deep neural network architecture for real-time semantic segmentation," 2016, *arXiv:1606.02147*.

[20] M. Treml, J. Arjona-Medina, T. Unterthiner, R. Durgesh, F. Friedmann, P. Schuberth, A. Mayr, M. Heusel, M. Hofmarcher, M. Widrich, B. Nessler, and S. Hochreiter, "Speeding up semantic segmentation for autonomous driving," in *Proc. MLITS, NIPS Workshop*, 2016, vol. 2, no. 7, pp. 1–77.

[21] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "BiseNet: Bilateral segmentation network for real-time semantic segmentation," in *Proc. ECCV*, 2018, pp. 325–341.

[22] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, "ICNet for real-time semantic segmentation on high-resolution images," in *Proc. ECCV*, 2018, pp. 405–420.

[23] G. Lin, A. Milan, C. Shen, and I. Reid, "RefineNet: Multi-path refinement networks for high-resolution semantic segmentation," in *Proc. CVPR*, Jul. 2017, pp. 5168–5177.

[24] R. P. K. Poudel, U. Bonde, S. Liwicki, and C. Zach, "ContextNet: Exploring context and detail for semantic segmentation in real-time," 2018, *arXiv:1805.04554*.

[25] G. J. Brostow, J. Fauqueur, and R. Cipolla, "Semantic object classes in video: A high-definition ground truth database," *Pattern Recognit. Lett.*, vol. 30, no. 2, pp. 88–97, Jan. 2009.

[26] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, "BDD100K: A diverse driving dataset for heterogeneous multitask learning," in *Proc. CVPR*, Jun. 2020, pp. 2633–2642.

[27] H. A. Alhaija, S. K. Mustikovela, L. Mescheder, A. Geiger, and C. Rother, "Augmented reality meets computer vision: Efficient data generation for urban driving scenes," *Int. J. Comput. Vis.*, vol. 126, no. 9, pp. 961–972, Sep. 2018.

[28] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. CVPR*, Jun. 2016, pp. 3213–3223.

[29] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. CVPR*, Jun. 2009, pp. 248–255.

[30] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. MICCAI*. Cham, Switzerland: Springer, 2015, pp. 234–241.

[31] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[32] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. CVPR*, Jul. 2017, pp. 1251–1258.

[33] Z. Wu, C. Shen, and A. van den Hengel, "Wider or deeper: Revisiting the ResNet model for visual recognition," *Pattern Recognit.*, vol. 90, pp. 119–133, Jun. 2019.

[34] S. Li, Q. Zhou, J. Liu, J. Wang, Y. Fan, X. Wu, and L. J. Latecki, "DCM: A dense-attention context module for semantic segmentation," in *Proc. IEEE ICIP*, Oct. 2020, pp. 1431–1435.

[35] J. Fu, J. Liu, J. Jiang, Y. Li, Y. Bao, and H. Lu, "Scene segmentation with dual relation-aware attention network," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 6, pp. 2547–2560, Jun. 2021.

[36] J. Cai, Y. Liu, and P. Qin, "Attention based quick network with optical flow estimation for semantic segmentation," *IEEE Access*, vol. 11, pp. 12402–12413, 2023.

[37] J. He, S. Liang, X. Wu, B. Zhao, and L. Zhang, "MGSeg: Multiple granularity-based real-time semantic segmentation network," *IEEE Trans. Image Process.*, vol. 30, pp. 7200–7214, 2021.

[38] M. Oršić and S. Šegvić, "Efficient semantic segmentation with pyramidal fusion," *Pattern Recognit.*, vol. 110, Feb. 2021, Art. no. 107611.

[39] R. P. K. Poudel, S. Liwicki, and R. Cipolla, "Fast-SCNN: Fast semantic segmentation network," 2019, *arXiv:1902.04502*.

[40] T. Singha, D.-S. Pham, and A. Krishna, "FANet: Feature aggregation network for semantic segmentation," in *Proc. DICTA*, 2020, pp. 1–8.

[41] T. Singha, D.-S. Pham, A. Krishna, and J. Dunstan, "Efficient segmentation pyramid network," in *Proc. ICONIP*. Cham, Switzerland: Springer, 2020, pp. 386–393.

[42] H. Li, P. Xiong, H. Fan, and J. Sun, "DFANet: Deep feature aggregation for real-time semantic segmentation," in *Proc. CVPR*, Jun. 2019, pp. 9522–9531.

[43] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. CVPR*, Jun. 2018, pp. 4510–4520.

[44] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, pp. 1–9, Apr. 2017.

[45] A. Howard, M. Sandler, B. Chen, W. Wang, L. Chen, M. Tan, G. Chu, V. Vasudevan, Y. Zhu, R. Pang, H. Adam, and Q. Le, "Searching for MobileNetV3," in *Proc. ICCV*, Oct. 2019, pp. 1314–1324.

[46] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *Proc. CVPR*, Jun. 2018, pp. 8759–8768.

[47] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and efficient object detection," 2019, *arXiv:1911.09070*.

[48] A. Mishra, S. Kumar, T. Kalluri, G. Varma, A. Subramaian, M. Chandraker, and C. Jawahar, "Semantic segmentation datasets for resource constrained training," in *Proc. NCVPRIPG*. Cham, Switzerland: Springer, 2019, pp. 450–459.

[49] A. Sergeev and M. D. Balso, "Horovod: fast and easy distributed deep learning in TensorFlow," *CoRR*, vol. abs/1802.05799, pp. 1–10, Feb. 2018.

[50] P. Goyal, P. Dollár, R. B. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch SGD: Training ImageNet in 1 hour," *CoRR*, vol. abs/1706.02677, pp. 1–12, Jun. 2017.

[51] M. Fan, S. Lai, J. Huang, X. Wei, Z. Chai, J. Luo, and X. Wei, "Rethinking BiSeNet for real-time semantic segmentation," in *Proc. CVPR*, Jun. 2021, pp. 9716–9725.

[52] Y. Zhu, K. Sapra, F. A. Reda, K. J. Shih, S. Newsam, A. Tao, and B. Catanzaro, "Improving semantic segmentation via video propagation and label relaxation," in *Proc. CVPR*, Jun. 2019, pp. 8856–8865.

[53] M. Klingner, J.-A. Termöhlen, J. Mikolajczyk, and T. Fingscheidt, "Self-supervised monocular depth estimation: Solving the dynamic object problem by semantic guidance," in *Proc. ECCV*. Cham, Switzerland: Springer, 2020, pp. 582–600.

[54] M. Ochs, A. Kretz, and R. Mester, "SDNet: Semantically guided depth estimation network," in *Proc. German Conf. Pattern Recognit.* Cham, Switzerland: Springer, 2019, pp. 288–302.

[55] S. Kong and C. Fowlkes, "Pixel-wise attentional gating for scene parsing," in *Proc. WACV*, Jan. 2019, pp. 1024–1033.

[56] B. Baheti, S. Innani, S. Gajre, and S. Talbar, "Eff-UNet: A novel architecture for semantic segmentation in unstructured environment," in *Proc. CVPR Workshops*, Jun. 2020, pp. 1473–1481.

[57] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. ICML*, 2019, pp. 6105–6114.

[58] G. Varma, A. Subramanian, A. Namboodiri, M. Chandraker, and C. V. Jawahar, "IDD: A dataset for exploring problems of autonomous navigation in unconstrained environments," in *Proc. WACV*, Jan. 2019, pp. 1743–1751.

TANMAY SINGHA received the Master of Technology and Master of Computer Applications degrees in information technology from the University of Calcutta, India. He is currently a Ph.D. Research Scholar in computer science with Curtin University, Australia. Before joining Curtin University, he was a Lecturer with the Department of IT, Royal University of Bhutan, Bhutan, for nine years. His current research interests include computer vision tasks, such as semantic and instance segmentation, object detection, scene graph generation, indoor and outdoor scene analysis, human pose estimation, facial landmark detection, and medical image processing using deep neural networks.

DUC-SON PHAM (Senior Member, IEEE) received the Ph.D. degree from the Curtin University of Technology, in 2005. He is currently a Senior Lecturer with the Discipline of Computing, Curtin University, Perth, WA, Australia. His current research interests include sparse learning theory, large-scale data mining, convex optimization, and advanced deep learning with applications to computer vision and image processing. He was a recipient of the Young Author Best Paper Award for a publication in IEEE TRANSACTIONS ON SIGNAL PROCESSING, in 2010.

ANEESH KRISHNA received the Ph.D. degree in computer science from the University of Wollongong, Australia. He was a Lecturer with the School of Computer Science and Software Engineering, University of Wollongong, from February 2006 to June 2009. He is currently a Discipline Lead of computing with the School of Electrical Engineering, Computing and Mathematical Sciences, Curtin University, Australia. He has expertise in large-scale complex software development, model-driven development and evolution, service computing, software quality, requirements engineering, agent systems, formal methods, data mining, computer vision, and machine learning. His research is (or has been) funded by the Australian Research Council (ARC), Australian government agencies, such as the Department of Defence, the Australian Academy of Science, the NSW State Emergency Service, and the Department of Industry, Science, Energy and Resources, as well as various industry partners. He has supervised nine Ph.D. and two M.Phil. students to the successful completion of their degrees. He works closely with industry and undertakes practical; real-world industry-focused research projects. He has worked (and led in most instances) on several successful projects with companies, such as Woodside Energy, Amristar Solutions, Thales, Deloitte, IBM, Immersive Technologies, Gaia Resources, AFG Group, Autism West Inc., BW Solar Australia, Western Australia Dementia Training Center, and Andrew Corporation. He has widely published (with more than 140 articles) in the above areas in top journals and international conferences. He has been on the organizing committee and served as an invited technical program committee member for many conferences and workshops in the areas related to his research.

Statement of contributions

Statement of Contribution of others

All of the written materials submitted as part of this PhD by Publication were conceived and coordinated by Tanmay Singha. Tanmay also undertook the majority of the empirical data collection, analysis and writing for each publication.

Signed detailed statements from all co-authors relating to each publication are provided as appendices at the next page.

Tanmay Singha 30 June 2023

Dr. Aneesh Krishna 30 June 2023
(Supervisor)

Dr. Duc-Son Pham 30 June 2023
(Co-supervisor)

Contribution statements

STATEMENT BY CO-AUTHORS

Publication 1

Singha, T., Pham, DS., Krishna, A., Dunstan, J. (2020). **Efficient Segmentation Pyramid Network**. In: ICONIP 2020. vol 1332. Springer, Cham. https://doi.org/10.1007/978-3-030-63820-7-44

Contributors	Statement of Contribution	% Total Contribution
Tanmay Singha	Conceptualization, Methodology, software, Validation, data analysis, Writing- original draft, Writing - Review and Editing	70%
Dr. Duc-Son Pham	Validation, Resources, Writing - Review and Editing, Supervision	20%
Dr. Aneesh Krishna	Critical review of the article, Editing, Supervision	10%

Tanmay Singha 30 June 2023

I, as a co-author endorse that this level of contribution by the candidate indicated above is appropriate.

Dr. Duc-Son Pham 30 June 2023

I, as a co-author endorse that this level of contribution by the candidate indicated above is appropriate.

Dr. Aneesh Krishna 30 June 2023

STATEMENT BY CO-AUTHORS

Publication 2

T. Singha, D. -S. Pham and A. Krishna, "**FANet: Feature Aggregation Network for Semantic Segmentation**,". In: Proc. DICTA, Melbourne, Australia, 2020, pp. 1-8, doi: 10.1109/DICTA51227.2020.9363370.

Contributors	Statement of Contribution	% Total Contribution
Tanmay Singha	Conceptualization, Methodology, software, Validation, data analysis, Writing- original draft, Writing - Review and Editing	70%
Dr. Duc-Son Pham	Validation, Resources, Writing - Review and Editing, Supervision	20%
Dr. Aneesh Krishna	Critical review of the article, Editing, Supervision	10%

Tanmay Singha 30 June 2023

I, as a co-author endorse that this level of contribution by the candidate indicated above is appropriate.

Dr. Duc-Son Pham 30 June 2023

I, as a co-author endorse that this level of contribution by the candidate indicated above is appropriate.

Dr. Aneesh Krishna 30 June 2023

STATEMENT BY CO-AUTHORS

Publication 3

Singha, T., Pham, DS., Krishna, A., Gedeon, T. (2021). "**A Lightweight Multi-scale Feature Fusion Network for Real-Time Semantic Segmentation**". In: Proc. ICONIP 2021. vol 13109. Springer, Cham. https://doi.org/10.1007/978-3-030-92270-2-17.

Contributors	Statement of Contribution	% Total Contribution
Tanmay Singha	Conceptualization, Methodology, software, Validation, data analysis, Writing- original draft, Writing - Review and Editing	70%
Dr. Duc-Son Pham	Validation, Resources, Writing - Review and Editing, Supervision	20%
Dr. Aneesh Krishna	Critical review of the article, Editing, Supervision	10%

Tanmay Singha 30 June 2023

I, as a co-author endorse that this level of contribution by the candidate indicated above is appropriate.

Dr. Duc-Son Pham 30 June 2023

I, as a co-author endorse that this level of contribution by the candidate indicated above is appropriate.

Dr. Aneesh Krishna 30 June 2023

STATEMENT BY CO-AUTHORS

Publication 4

T. Singha, M. Bergemann, D. -S. Pham and A. Krishna, "**SCMNet: Shared Context Mining Network for Real-time Semantic Segmentation**," In: Proc. DICTA, Gold Coast, Australia, 2021, pp. 1-8, doi: 10.1109/DICTA52665.2021.9647401.

Contributors	Statement of Contribution	% Total Contribution
Tanmay Singha	Conceptualization, Methodology, software, Validation, data analysis, Writing- original draft, Writing - Review and Editing	60%
Moritz Bergemann	software, Validation, Writing - Review and Editing	15%
Dr. Duc-Son Pham	Resources, Writing - Review and Editing, Supervision	15%
Dr. Aneesh Krishna	Critical review of the article, Editing, Supervision	10%

Tanmay Singha 30 June 2023

I, as a co-author endorse that this level of contribution by the candidate indicated above is appropriate.

Dr. Duc-Son Pham 30 June 2023

I, as a co-author endorse that this level of contribution by the candidate indicated above is appropriate.

Dr. Aneesh Krishna 30 June 2023

STATEMENT BY CO-AUTHORS

Publication 5

Singha, Tanmay, Pham, Duc-Son, and Krishna, Aneesh. "**Urban Street Scene Analysis Using Lightweight Multi-level Multi-path Feature Aggregation Network**". Multiagent and Grid Systems, vol. 17, no. 3, pp. 249-271, 2021. DOI: 10.3233/MGS-210353.

Contributors	Statement of Contribution	% Total Contribution
Tanmay Singha	Conceptualization, Methodology, software, Validation, data analysis, Writing- original draft, Writing - Review and Editing	70%
Dr. Duc-Son Pham	Validation, Resources, Writing - Review and Editing, Supervision	20%
Dr. Aneesh Krishna	Critical review of the article, Editing, Supervision	10%

Tanmay Singha 30 June 2023

I, as a co-author endorse that this level of contribution by the candidate indicated above is appropriate.

Dr. Duc-Son Pham 30 June 2023

I, as a co-author endorse that this level of contribution by the candidate indicated above is appropriate.

Dr. Aneesh Krishna 30 June 2023

STATEMENT BY CO-AUTHORS

Publication 6

T. Singha, M. Bergemann, D. -S. Pham and A. Krishna, "**SC-CrackSeg: A Real-time Shared Feature Pyramid Network for Crack Detection and Segmentation**," In: Proc. DICTA, Sydney, Australia, 2022, pp. 1-8, doi: 10.1109/DICTA56598.2022.10034629.

Contributors	Statement of Contribution	% Total Contribution
Tanmay Singha	Conceptualization, Methodology, software, Validation, data analysis, Writing- original draft, Writing - Review and Editing	60%
Moritz Bergemann	software, Validation, Writing - Review and Editing	15%
Dr. Duc-Son Pham	Resources, Writing - Review and Editing, Supervision	15%
Dr. Aneesh Krishna	Critical review of the article, Editing, Supervision	10%

Tanmay Singha 30 June 2023

I, as a co-author endorse that this level of contribution by the candidate indicated above is appropriate.

Moritz Bergemann 30 June 2023

I, as a co-author endorse that this level of contribution by the candidate indicated above is appropriate.

Dr. Duc-Son Pham 30 June 2023

I, as a co-author endorse that this level of contribution by the candidate indicated above is appropriate.

Dr. Aneesh Krishna 30 June 2023

STATEMENT BY CO-AUTHORS

Publication 7

T. Singha, D. -S. Pham and A. Krishna, "**SDBNet: Lightweight Real-time Semantic Segmentation Using Short-term Dense Bottleneck**," In: Proc. DICTA, Sydney, Australia, 2022, pp. 1-8, doi: 10.1109/DICTA56598.2022.10034634.

Contributors	Statement of Contribution	% Total Contribution
Tanmay Singha	Conceptualization, Methodology, software, Validation, data analysis, Writing- original draft, Writing - Review and Editing	70%
Dr. Duc-Son Pham	Validation, Resources, Writing - Review and Editing, Supervision	20%
Dr. Aneesh Krishna	Critical review of the article, Editing, Supervision	10%

Tanmay Singha 30 June 2023

I, as a co-author endorse that this level of contribution by the candidate indicated above is appropriate.

Dr. Duc-Son Pham 30 June 2023

I, as a co-author endorse that this level of contribution by the candidate indicated above is appropriate.

Dr. Aneesh Krishna 30 June 2023

STATEMENT BY CO-AUTHORS

Publication 8

Tanmay Singha, Duc-Son Pham, Aneesh Krishna, "**A real-time semantic segmentation model using iteratively shared features in multiple sub-encoders**", Pattern Recognition, Volume 140, 2023, 109557, ISSN 0031-3203, https://doi.org/10.1016/j.patcog.2023.109557.

Contributors	Statement of Contribution	% Total Contribution
Tanmay Singha	Conceptualization, Methodology, software, Validation, data analysis, Writing- original draft, Writing - Review and Editing	70%
Dr. Duc-Son Pham	Validation, Resources, Writing - Review and Editing, Supervision	20%
Dr. Aneesh Krishna	Critical review of the article, Editing, Supervision	10%

Tanmay Singha 30 June 2023

I, as a co-author endorse that this level of contribution by the candidate indicated above is appropriate.

Dr. Duc-Son Pham 30 June 2023

I, as a co-author endorse that this level of contribution by the candidate indicated above is appropriate.

Dr. Aneesh Krishna 30 June 2023

STATEMENT BY CO-AUTHORS

Publication 9

T. Singha, D. -S. Pham and A. Krishna, "**Multi-encoder Context Aggregation Network for Structured and Unstructured Urban Street Scene Analysis**," in IEEE Access, doi: 10.1109/ACCESS.2023.3289968.

Contributors	Statement of Contribution	% Total Contribution
Tanmay Singha	Conceptualization, Methodology, software, Validation, data analysis, Writing- original draft, Writing - Review and Editing	70%
Dr. Duc-Son Pham	Validation, Resources, Writing - Review and Editing, Supervision	20%
Dr. Aneesh Krishna	Critical review of the article, Editing, Supervision	10%

Tanmay Singha 30 June 2023

I, as a co-author endorse that this level of contribution by the candidate indicated above is appropriate.

Dr. Duc-Son Pham 30 June 2023

I, as a co-author endorse that this level of contribution by the candidate indicated above is appropriate.

Dr. Aneesh Krishna 30 June 2023

Attribution statement

Attribution Statement

Publications and titles	% Total contributions				
	Co-author 1 (Dr. Duc-Son Pham)	Co-author 2 (Dr. Aneesh Krishna)	Co-author 3 (Mr. Moritz Bergemann)	Co-author 4 (Prof. Tom Gedeon)	Co-author 5 (Mr. Joel Dunstan)
Publication 1: Efficient Segmentation Pyramid Network	20%	10%			5%
Publication 2: FANet: Feature Aggregation Network for Semantic Segmentation	20%	10%			
Publication 3: A Lightweight Multi-scale Feature Fusion Network for Real-Time Semantic Segmentation	20%	10%		5%	
Publication 4: SCMNet: Shared Context Mining Network for Real-time Semantic Segmentation	15%	10%	15%		
Publication 5: Urban Street Scene Analysis Using Lightweight Multi-level Multi-path Feature Aggregation Network	20%	10%			
Publication 6: SC-CrackSeg: A Real-time Shared Feature Pyramid Network for Crack Detection and Segmentation	15%	10%	15%		
Publication 7: SDBNet: Lightweight Real-time Semantic Segmentation Using Short-term Dense Bottleneck	20%	10%			
Publication 8: A real-time semantic segmentation model using iteratively shared features in multiple sub-encoders	20%	10%			
Publication 9: Multi-encoder Context Aggregation Network for Structured and Unstructured Urban Street Scene Analysis	20%	10%			
	Attribution statement				
Co-authors Name	Contributions	Co-authors Acknowledgement: I acknowledge that these represent my contribution to the above research output.			

Co-author 1 (Dr. Duc-Son Pham)	Validation, Resources, Writing - Review and Editing, Supervision	Signature:	7/7/23
Co-author2 (Dr. Aneesh Krishna)	Critical review of the article, Editing, Supervision	Signature:	7/7/23
Co-author 3 (Mr. Moritz Bergemann)	Software, Validation, Writing - Review and Editing	Signature:	7/7/23
Co-author 4 (Prof. Tom Gedeon)	Critical review of the article	Signature:	
Co-author 5 (Mr. Joel Dunstan)	Software	Signature:	7/7/23

Copyright release for published materials

RightsLink

Help Live Chat Tanmay Singha

Efficient Segmentation Pyramid Network

Author: Tanmay Singha, Duc-Son Pham, Aneesh Krishna et al
Publication: Springer eBook
Publisher: Springer Nature
Date: Jan 1, 2020

Order Completed

Thank you for your order.

This Agreement between Tanmay Singha ("You") and Springer Nature ("Springer Nature") consists of your order details and the terms and conditions provided by Springer Nature and Copyright Clearance Center.

License number	Reference confirmation email for license number
License date	Jun, 09 2023

Licensed Content

Licensed Content Publisher	Springer Nature
Licensed Content Publication	Springer eBook
Licensed Content Title	Efficient Segmentation Pyramid Network
Licensed Content Author	Tanmay Singha, Duc-Son Pham, Aneesh Krishna et al
Licensed Content Date	Jan 1, 2020

Order Details

Type of Use	Thesis/Dissertation
Requestor type	academic/university or research institute
Format	print and electronic
Portion	full article/chapter
Will you be translating?	no
Circulation/distribution	1 - 29
Author of this Springer Nature content	yes

About Your Work

Title	PhD thesis: Scene understanding using Deep Neural Networks
Institution name	Curtin University
Expected presentation date	Aug 2023

Additional Data

Requestor Location

Requestor Location	Mr. Tanmay Singha 1 Kyle Avenue Perth, Western Australia 6102 Australia Attn: Mr. Tanmay Singha

Tax Details

Billing Information

Billing Type	Invoice
Billing address	Mr. Tanmay Singha 1 Kyle Avenue Perth, Australia 6102 Attn: Mr. Tanmay Singha

Price

Total	0.00 AUD

Total: 0.00 AUD

CLOSE WINDOW

FANet: Feature Aggregation Network for Semantic Segmentation

Conference Proceedings: 2020 Digital Image Computing: Techniques and Applications (DICTA)

Author: Tanmay Singha

Publisher: IEEE

Date: 29 November 2020

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

A Lightweight Multi-scale Feature Fusion Network for Real-Time Semantic Segmentation

Author: Tanmay Singha, Duc-Son Pham, Aneesh Krishna et al
Publication: Springer eBook
Publisher: Springer Nature
Date: Jan 1, 2021

Order Completed

Thank you for your order.

This Agreement between Tanmay Singha ("You") and Springer Nature ("Springer Nature") consists of your order details and the terms and conditions provided by Springer Nature and Copyright Clearance Center.

License number	Reference confirmation email for license number
License date	Jun, 09 2023

Licensed Content

Licensed Content Publisher	Springer Nature
Licensed Content Publication	Springer eBook
Licensed Content Title	A Lightweight Multi-scale Feature Fusion Network for Real-Time Semantic Segmentation
Licensed Content Author	Tanmay Singha, Duc-Son Pham, Aneesh Krishna et al
Licensed Content Date	Jan 1, 2021

Order Details

Type of Use	Thesis/Dissertation
Requestor type	academic/university or research institute
Format	print and electronic
Portion	full article/chapter
Will you be translating?	no
Circulation/distribution	1 - 29
Author of this Springer Nature content	yes

About Your Work

Title	PhD thesis: Scene understanding using Deep Neural Networks
Institution name	Curtin University
Expected presentation date	Aug 2023

Additional Data

Requestor Location

Mr. Tanmay Singha
1 Kyle Avenue

Requestor Location

Perth, Western Australia
6102
Australia
Attn: Mr. Tanmay Singha

Tax Details

Billing Information

Billing Type: Invoice

Billing address

Mr. Tanmay Singha
1 Kyle Avenue

Perth, Australia 6102
Attn: Mr. Tanmay Singha

Price

Total 0.00 AUD

Total: 0.00 AUD

CLOSE WINDOW

RE: Rights&Permissions Contact Formulier

Carry Koolbergen <C.Koolbergen@iospress.nl>

Fri 2/06/2023 8:15 PM

To:Tanmay Singha <tanmay.singha@postgrad.curtin.edu.au>

DOI: 10.3233/MGS-210353
Citation: Multiagent and Grid Systems, vol. 17, no. 3, pp. 249-271, 2021

Dear Tanmay Singha,

We hereby grant you permission to reproduce the below mentioned material in **print and electronic format** at no charge subject to the following conditions:

1. If any part of the material to be used (for example, figures) has appeared in our publication with credit or acknowledgement to another source, permission must also be sought from that source. If such permission is not obtained then that material may not be included in your publication/copies.

2. Suitable acknowledgement to the source must be made, either as a footnote or in a reference list at the end of your publication, as follows:

   "Reprinted from Publication title, Vol number, Author(s), Title of article, Pages No., Copyright (Year), with permission from IOS Press".
   "The publication is available at IOS Press through http://dx.doi.org/[insert DOI]"

3. This permission is granted for non-exclusive world **English** rights only. For other languages please reapply separately for each one required.

4. Reproduction of this material is confined to the purpose for which permission is hereby given.

Yours sincerely

Carry Koolbergen (Mrs.)
Contracts, Rights & Permissions Coordinator
Not in the office on Wednesdays

IOS Press | Nieuwe Hemweg 6B, 1013 BG Amsterdam, The Netherlands
Tel.: +31 (0)20 688 3355/ +31 (0) 687 0022 | c.koolbergen@iospress.nl | www.iospress.nl
Twitter: @IOSPress_STM | Facebook: publisheriospress
View the latest IOS Press newsletter here

From: tanmay.singha@postgrad.curtin.edu.au [mailto:tanmay.singha@postgrad.curtin.edu.au]
Sent: vrijdag 2 juni 2023 08:06
To: Carry Koolbergen <C.Koolbergen@iospress.nl>
Subject: Rights&Permissions Contact Formulier

Submitted on Fri, 06/02/2023 - 08:05

Submitted by: Anonymous

Submitted values are:

Name
Tanmay Singha

Email address

Department
Rights & Permissions

Personal Details
Tanmay Singha
Curtin University
1 Kyle Avenue
Perth. 6102
Australia
.

Title
PhD Thesis

Published by
Curtin University espace

Article
Urban street scene analysis using lightweight multi-level multi-path feature aggregation network

Author(s)
Singha, Tanmay* | Pham, Duc-Son | Krishna, Aneesh

Journal or Book Title
Multiagent and Grid Systems

Volume / Issue Nr:
17 / 3

Pages
249-271

DOI
10.3233/MGS-210353

What would you like to use from the IOS Press article?
Portions or extracts

For what purpose?
Inclusion in a thesis or dissertation

SC-CrackSeg: A Real-time Shared Feature Pyramid Network for Crack Detection and Segmentation

Conference Proceedings:
2022 International Conference on Digital Image Computing: Techniques and Applications (DICTA)

Author: Tanmay Singha

Publisher: IEEE

Date: 30 November 2022

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University MicroFilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

SDBNet: Lightweight Real-time Semantic Segmentation Using Short-term Dense Bottleneck

Conference Proceedings:
2022 International Conference on Digital Image Computing: Techniques and Applications (DICTA)

Author: Tanmay Singha

Publisher: IEEE

Date: 30 November 2022

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

Re: Obtain permission request - Website (1358300) [230602-006822]

Rights and Permissions (ELS) <Permissions@elsevier.com>
Mon 5/06/2023 6:12 PM
To:Tanmay Singha <tanmay.singha@postgrad.curtin.edu.au>

Dear Tanmay Singha,

Thank you so much for contacting us.

Please note that, as one of the authors of this article, you retain the right to reuse it in your thesis/dissertation. You do not require formal permission to do so. You are permitted to post this Elsevier article online if it is embedded within your thesis subject to proper acknowledgment;

Our preferred acknowledgement wording will be :

Example: "This article/chapter was published in Publication title, Vol number, Author(s), Title of article, Page Nos, Copyright Elsevier (or appropriate Society name) (Year)."

All the best for your thesis submission!

Kind regards,

Kaveri Thakuria

Senior Copyrights Coordinator

ELSEVIER | HCM - Health Content Management

Visit Elsevier Permissions

From: Administrator
Date: Friday, June 02, 2023 04:38 AM GMT

Dear Tanmay Singha,

Thank you for contacting the Permissions Granting Team.

We acknowledge the receipt of your request and we aim to respond within seven business days. Your unique reference number is 230602-006822.

Please avoid changing the subject line of this email when replying to prevent a delay with your query.

Regards,

Permission Granting Team

From: Tanmay Singha
Date: Friday, June 02, 2023 04:38 AM GMT

Submission ID: 1358300
Date: 02 Jun 2023 5:38am

Name: Mr. Tanmay Singha
Institute/company: Curtin University
Address: Kent street
Post/Zip Code: 6102
City: Perth
State/Territory: Western Australia
Country: Australia
Telephone:
Email: 1

Type of Publication: Website

Title: A real-time semantic segmentation model using iteratively shared features in multiple sub-encoders
Website URL: http://espace.curtin.edu.au

I would like to use: Full article / chapter

I am the author of the Elsevier material: Yes
Involvement: I am the lead author of this paper and want to reuse the paper in my PhD thesis

In what format will you use the material: Print and Electronic
Translation: No

Proposed use: Reuse in a thesis/dissertation

Material can be extracted: No

Additional Comments / Information:

Creative Commons Attribution License (CCBY)

Multi-encoder Context Aggregation Network for Structured and Unstructured Urban Street Scene Analysis

TANMAY SINGHA,DUC-SON PHAM,ANEESH KRISHNA

IEEE Access

By clicking the checkbox at the bottom of this page you, as the author or representative of the author, confirm that your work is licensed to IEEE under the Creative Commons Attribution 4.0(CCBY 4.0). As explained by the Creative Commons web site, this license states that IEEE is free to share, copy, distribute and transmit your work under the following conditions:

- Attribution - Users must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse the users or their use of the work).

With the understanding that:

- **Waiver** - Any of the above conditions can be waived if users get permission from the copyright holder.
- **Public Domain** - Where the work or any of its elements is in the public domain under applicable law, that status is in no way affected by the license.
- **Other Rights** - In no way are any of the following rights affected by the license:
  - A user's fair dealing or fair use rights, or other applicable copyright exceptions and limitations;
  - The author's moral rights;
  - Rights other persons may have either in the work itself or in how the work is used, such as publicity or privacy rights.

For any reuse or distribution, users must make clear to others the license terms of this work.

Upon clicking on the checkbox below, you will not only confirm that your submission is under the CCBY license but you will also be taken to IEEE's Terms of Use, which will require your signature.

[X] I confirm the submitted work is licensed to IEEE under the Creative Commons Attribution 4.0 United States (CCBY 4.0)

TERMS AND CONDITIONS OF AN AUTHOR'S USE OF THE CREATIVE COMMONS ATTRIBUTION LICENSE (CCBY)

1. Creative Commons Licensing

To grow the commons of free knowledge and free culture, all users are required to grant broad permissions to the general public to re-distribute and re-use their contributions freely. Therefore, for any text, figures, or other work in any medium you hold the copyright to, by submitting it, you agree to license it under the Creative Commons Attribution 4.0 Unported License.

2. Attribution

As an author, you agree to be attributed in any of the following fashions: a) through a hyperlink (where possible) or URL to the article or articles you contributed to, b) through a hyperlink (where possible) or URL to an alternative, stable online copy which is freely accessible, which conforms with the license, and which provides credit to the authors in a manner equivalent to the credit given on this website, or c) through a list of all authors.

3. Terms of Publication

A. By submitting your work to IEEE, you agree to comply with the IEEE Publication Services and Products Board Operations Manual (the "Operations Manual"), including, but not limited to, the specific provisions referenced herein(except to the extent any provision of the Operations Manual requires assignment of copyright in your work to IEEE).

B. Submission to this IEEE journal does not guarantee publication. By submitting your work to this journal you, as author, recognize that your

work may be rejected for any reason. All submissions shall be reviewed by the Editor in accordance with section 8.2.2 of the Operations Manual.

C. Should your paper be rejected IEEE will not exercise any of the rights granted to it under the Creative Commons Attribution 4.0 Unported License.

D. IEEE takes intellectual property protection seriously and is opposed to plagiarism in any fashion. Accordingly, you consent to having your work submitted to a plagiarism detection tool and to be bound by IEEE policies concerning plagiarism and author misconduct.

E. IEEE distributes its technical publications throughout the world and wants to ensure that the material submitted to its publications is properly available to the readership of those publications. You must ensure that your work meets the requirements as stated in section 8.2.1 of the Operations Manual, including provisions covering originality, authorship, author responsibilities and author misconduct. More information on IEEE's publishing policies may be found at https://www.ieee.org/publications/rights/author-rights-responsibilities.html.

F. You warrant that your work, including and any accompanying materials, is original and that you are the author of the work. To the extent your work incorporates text passages, figures, data or other material from the works of others, you represent and warrant that you have obtained all third party permissions and consents to grant the rights herein and have provided copies of such permissions and consents to IEEE. As stated in section 8.2.1B12 of the Operations Manual: "It is the responsibility of the authors, not the IEEE, to determine whether disclosure of their material requires the prior consent of other parties and, if so, to obtain it."

G. You are advised of Operations Manual section 8.1.1B: "Statements and opinions given in work published by the IEEE are the expression of the authors."

H. You agree that publication of a notice of violation as a corrective action for a confirmed case of plagiarism, as described in Section 8.2.4 of the IEEE PSPB Publications Operations Manual, does not violate any of your moral rights.

I. You agree to indemnify and hold IEEE and its parents, subsidiaries, affiliates, officers, employees, agents, partners and licensors harmless from any claim or demand, including reasonable attorneys' fees, due to or arising out of: (1) content you submit, post, transmit or otherwise make available through IEEE's publishing program; (2) your use of this IEEE journal; (3) your violation of these Terms of Use; or (4) your violation of any rights of another party.

BY TYPING IN YOUR FULL NAME BELOW AND CLICKING THE SUBMIT BUTTON, YOU CERTIFY THAT SUCH ACTION CONSTITUTES YOUR ELECTRONIC SIGNATURE TO THIS FORM IN ACCORDANCE WITH UNITED STATES LAW, WHICH AUTHORIZES ELECTRONIC SIGNATURE BY AUTHENTICATED REQUEST FROM A USER OVER THE INTERNET AS A VALID SUBSTITUTE FOR A WRITTEN SIGNATURE.

[redacted] 25-06-2023

Signature Date

Questions about the submission of the form or manuscript must be sent to the publication's editor. Please direct all questions about IEEE copyright policy to:
IEEE Intellectual Property Rights Office, copyrights@ieee.org, +1-732-562-3966