School of Electrical Engineering, Computing and Mathematical Sciences

Department of Electrical and Computer Engineering

Vulnerable Road Users Detection using Convolutional Deep Feedforward Network

Lau Mian Mian

0000-0002-8940-0347

This thesis is presented for the degree of

Doctor of Philosophy

of

Curtin University

May 2021

Student's 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.

The thesis contains no material which has been accepted for the award of any other degree or diploma in any university.

Name: Mian Mian Lau

Position: PhD Candidate

Date: 30th April 2021

Supervisor's Declaration

I hereby declare that I have read this thesis and in my opinion this thesis is sufficient in terms of scope and quality for the award of the degree of Doctor of Philosophy (By Research) in Engineering

Name: Dr King Hang Lim
Position: Associate Professor
Date: 12th April 2021

(On behalf of the thesis committee)
Thesis Chair: A/Prof. Dr. Huo Chong Ling
Principal Supervisor: A/Prof. Dr. Zhuquan Zang
Co-supervisor: A/Prof. Dr. King Hann Lim

Abstract

Vulnerable road users (VRUs) detection is the core process in the field of video surveillance and autonomous vehicle. However, many practical challenges are encountered in designing a single computer vision model to enable autonomous vehicle to capture and analyse the variation of VRUs appearance and moving traffic scenes. To design a better VRUs detection model, this research explores and investigates a series of structure analysis and development on deep neural networks (DNNs), which are currently popular models in many real-world applications due to its learning capability and generalization. This research studies the use of activation functions that can broadly be classified into three types, i.e. saturated, unsaturated, and adaptive activation functions. As a result, parametric rectified linear unit (PReLU) is one type of adaptive activation functions achieving the lowest test misclassification rate of 1.6% using MNIST dataset as compared to other activation functions. Subsequently, an adaptive parameter is added to the hyperbolic tangent function to overcome the vanishing gradient issue and reduce the training iterations in the network optimization. In addition, a new framework of VRUs detection model known as convolutional deep feedforward network (C-DFN) is proposed. This proposed model performs a better accuracy than deep feedforward network (DFN), deep belief network (DBN), and convolutional deep belief network (C-DBN) using MNIST dataset, INRIA pedestrian dataset and Daimler pedestrian dataset. C-DFN incorporated with PReLU activation function can further reduce an average of 9.22% from the misclassification rate. The convolutional layer in C-DFN acted as a trainable feature extractor can improve the network performance significantly and reduce 14% of the trainable parameters in DFN. For the VRUs detection system, C-DFN is also modified into stacked C-DFN holistic VRUs detection model and stacked C-DFN part-based C-DFN. As a result, part-based stacked C-DFN model archives the lowest miss detection rate and less false positive per image (FPPI) among all the detection models. Part-based model has further reduced 0.95% from the 3 stacked C-DFN's miss detection rate due to the use of upper and lower body parts as the cue in pedestrian detection. Therefore, the part-based model can detect the medium scale VRUs even if the image is at low-resolution. As in future work, larger pedestrian datasets can be deployed to train C-DFN model to reduce its miss detection rate and FPPI. Moreover, it can be further enhanced in a real-time detection system through code optimization.

Acknowledgement

First, I would like to express my greatest gratitude to Curtin University and Curtin Sarawak Research Institute (CSRI) for their funding support. I would like to express the deepest appreciation to my thesis community especially my co-supervisor Assoc. Prof. Dr. Lim King Hann for his patient, guidance, encouragement, constant inspiration throughout this research and teaching me how to become a qualified researcher. Special thanks to Dr. Alex Goh Kwang Leng for his support and assist during my experimental work. Finally, I sincerely thank my family and friends for their endless support and encouragement.

Table of Contents

List of Tables

List of Figures

List of Publications

1. M. M. Lau and K. H. Lim, “Convolutional and Fully Connected Layer in DFN.” *Journal of Information Science & Engineering* Vol. 36. No.5, 2020, pp. 1069-1078 – **(The content is demonstrated in Chapter 4.)**
2. M. M. Lau and K. H. Lim, "Convolutional and fully connected layer in DFN," in *2019 7th International conference on smart computing & communication (ICSCC 2019),* 2019, pp. 1-4 – **(The content is demonstrated in Chapter 4.)**
3. M. M. Lau, J.T.S. Phang and K. H. Lim, "Convolutional deep feedforward network for image classification," in *2019 7th International conference on smart computing & communication (ICSCC 2019),* 2019, pp. 1-4 --**(The content is demonstrated in Chapter 4.)**
4. M. M. Lau and K. H. Lim, "Review of Adaptive Activation Function in Deep Neural Network," in *2018 IEEE EMBS conference on biomedical engineering and sciences (IECBES)*, 2018, pp. 686-690. -- **(The content is demonstrated in Chapter 3.)**
5. M. M. Lau and K. H. Lim, "Investigation of activation functions in deep belief network," in *2017 2nd International Conference on Control and Robotics Engineering (ICCRE)*, 2017, pp. 201-206. -- **(The content is demonstrated in Chapter 3.)**
6. M. M. Lau, K. H. Lim, and A. A. Gopalai, "Malaysia traffic sign recognition with convolutional neural network," in *IEEE international conference on digital signal processing*, Singapore, 2015, pp. 1006-1010. -- **(The content is demonstrated in Chapter 3.)**

List of Abbreviation

ACF	Aggregate channel feature
ADAS	Advanced driver assistance systems
ANN	Artificial neural network
APL	Adaptive piecewise linear activation function
BB	Bounding box
BBdt	Detected Bounding Box
BBgt	Ground truth Bounding Box
BBig	Ignored Bounding Box
C-DBN	Convolutional deep belief network
C-DFN	Convolutional deep neural network
CL	Convolutional layer
CNN	Convolutional neural network
Co-HoG	Co-occurrence histogram of oriented gradient
COV	Covariance features
CP	Convolutional layer and pooling layer
CPFL	Single layer of convolution, pooling, and fully connected layer
CPU	Central processing unit
CSS	Colour-self-similarity
CtF	Course to fine
DBN	Deep belief network
DCNN	Deep convolutional neural network
DFN	Deep feedforward network
DNN	Deep neural network
DPM	Deformable part-based model
ECF	effective comparison feature
ELU	Exponential linear units
exp	Exponential
FCN	Fully connected network
FL	Fully connected layer
FPPI	False positive per image
GPU	graphical processing unit
HoG	Histogram of oriented gradient
ICA	Independent component analysis
LBP	Local Binary Patterns
LDCF	locally decorrelated channel feature
LFOV	large field of view deep network
LReLU	Leaky rectified linear unit
LUV	Colour channel
MFD	Multi-feature descriptor
MI-SVM	Multiple instant support vector machine
MOCO	Multi-order contextual co-occurrence
MSAF	Multistate activation function

MSAFsymm	Symmetrical multistate activation function
NCSS	Colour self-similarity in Neighbour
NiN	network in network
PASCAL	Pattern Analysis, Statistical Modelling, and Computational Learning
pAUC	Partial area under the ROC curve
PCA-CFF	Principle component analysis filter based convolutional channel features
PL	Pooling layer
PReLU	Parametric rectified linear unit
RBM	restricted Boltzmann machine
RCNN	Region based convolutional neural network
ReLU	Rectified linear unit
RNN	Recurrent neural network
ROI	Region of interest
SIFT	Scale-invariant feature transform
SRBM	switchable restricted Boltzmann machine
SVM	Support vector machine
Tanh	Hyperbolic tangent
VRUs	Vulnerable road users

List of Symbols

a	amplitude
b	bias
c	ReLU constant
d	input image
E	error function
f	activation function
$f(x)$	function output
i	input layer
j	hidden layer
k	convolution kernel
l	ith layer
M	input map
m	number of columns for subsampling
n	number of rows for subsampling
q	row of convolution map
r	column of convolution map
$r1$	multi-state constant
s	slope
u	output before activation function
up	up-sampling local gradients
v	input data
w	weights
x	ith data
X	top left corner coordinate
y	layer output
Y	top left corner coordinate
z	maxpooling output
α	ELU constant
δ	local gradients

Chapter 1 Introduction

1.1 Project Overview

Road traffic injuries are one of the major global public health issues associated with the loss of approximately 1.35 million lives yearly on the road [1]. In 2018, road accidents are reported as the eighth leading cause of death toll across all age groups globally. With less concern on traffic rule legislation and road safety improvement, road accidents are predicted to become the seventh leading cause of death toll by 2030. As referring to world statistics displayed in Fig. 1.1, there are 54% of the road fatalities involving vulnerable road users (VRUs) such as pedestrians, motorcyclists, and cyclists. Especially in the South-East Asia region, the fatalities of VRUs have made up to 59% of the total road fatalities. The traffic death rate in Malaysia has been ranked as the third highest rate globally, which is beyond the countries with a larger population such as China and India [2].

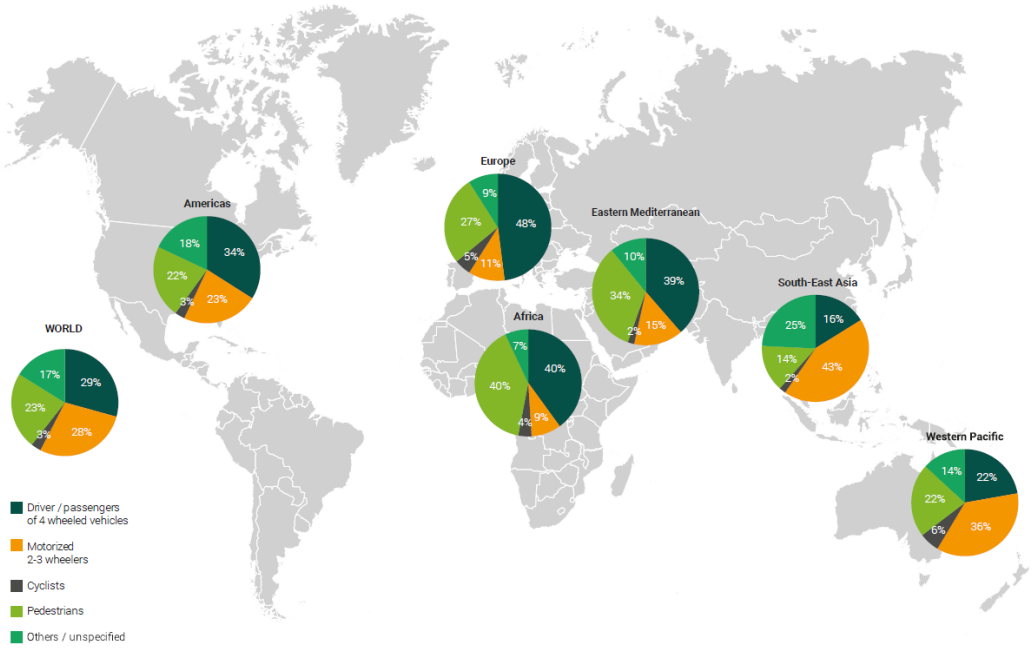


Figure 1.1: *World statistic of road traffic deaths in 2018 has reported that the death toll largely involves vulnerable road users (VRUs) 26% pedestrian and cyclists, 28% motorcyclists, 29% car occupants, and 17% were unidentified road users.* [1]

To improve road safety features in a vehicle, the emerging technology of Advanced Driver Assistance Systems (ADAS) and self-driving cars has drawn increasing attention of its potential to minimize the fatalities rate on the road. Due to the massive traffic loads and road structures, the implementation of driverless cars in

the South-East Asia region may incur huge challenges on the current road condition and environment. To gradually move towards the direction of driverless cars, an intensive research and development on ADAS is vastly focused to create more intelligent on-board processing system that aims to increase the pre-crash awareness to assist drivers on the road.

ADAS is designed in such a way that it can provide car-ahead road information, execute counteractive measure and assist a driver in making quick decisions to avoid collision [3]. Various existing ADAS in the current market is summarized in Table 1.1 with the detail description of its functionality. The integration of ADAS with on-board sensors can lead to self-driving cars as illustrated in Fig. 1.2. Optical sensors, or it may usually refer to colour cameras, is one of the major on-board sensors to detect objects on the road. The advantages of optical sensors are rich in visual information, easy configuration for video acquisition, and wide range capturing ability. As compared to other sensors such as radar, laser scanner, and infrared sensor, cameras can provide valuable information such as textual, colour, and shape at lower cost with the advancement of computer vision algorithms [4]. Therefore, visual-based ADAS has obtained great potential and realistic values to be installed on a vehicle in the low- and middle-income countries [5].

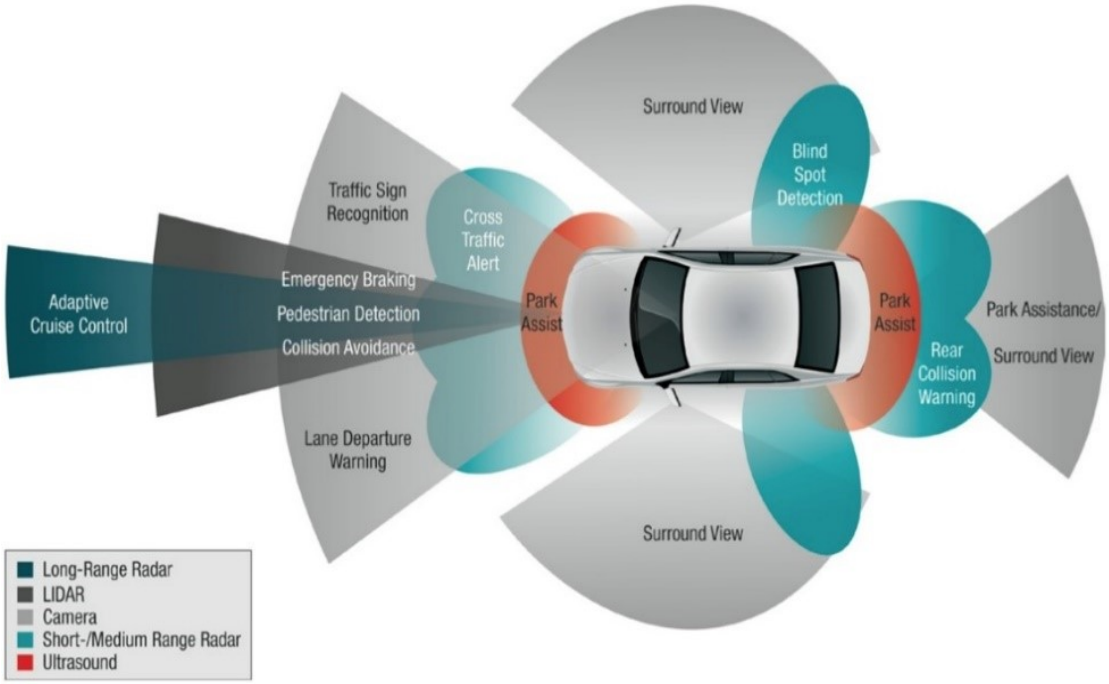


Figure 1.2: *Advanced driver assistance systems. [6]*

Table 1.1: *Advanced Driver Assistance Systems (ADAS) [7].*

ADAS	Functions	Sensors
Traffic sign recognition	To detect road sign and remind driver the current speed limit	Camera
Emergency brake assist	To activate an automatic braking system as soon as the vehicle within a dangerous distance from a preceding vehicle	LIDAR
Adaptive cruise control	To follow the flow of traffic ahead of the vehicle, allowing drivers to arrive more relax and safely.	Long-range radar
Blind spot detection	To warn the driver when there is a vehicle in the blind spot of the side view mirror	Short/medium range radar
Rear cross traffic alert	To avoid collision with objects behind the vehicle when reversing out of a parking space	Short/medium range radar
Intelligent headlamp control	To change the high and low light of the vehicle by considering the oncoming vehicle and travelling ahead vehicle into account	Camera
Lane departure warning	To alert the driver with an acoustical or haptic warning before his vehicle is about to leave the lane	Camera
Road departure protection	To apply the brake and steer the vehicle back on the road to avoid off-road accidents	Camera
Mirror replacement	To replace the side view mirror and rear view mirror into cameras to improve safety, efficiency and comfort.	Camera
GPS navigation	To provide vocal direction and live traffic data	Gyroscope, accelerometer
Hill descent control	To slow down the vehicle by activating the braking system to slow down the vehicle and facilitate safe travel down a steep hill	Pressure sensor, accelerometer
Driver drowsiness detection	To monitor the driver's eyes and face for a sign of drowsiness and send an alert the driver when drowsiness is detected	Camera
Pedestrian detection	To detect pedestrian on the road and avoid the collision	Camera, LIDAR

VRUs detection using cameras is increasingly important to reduce fatalities on the road. An overview of on-board VRUs detection system is showed in Fig. 1.3. First, an on-board camera captures the road scene as an input for a micro-computer processing unit. The processing unit converts a road image into a matrix form as the input for a computer vision algorithm known as a deep neural network (DNN). DNN is an artificial neural network (ANN) that contains more than two hidden layers in its architecture [8]. DNN mimics human brain recognition function in terms of a mathematical expression to solve complex real-world applications [9]. In VRUs detection, DNN extracts the pertinent features automatically from the raw images and classify them into VRUs or non-VRUs. If VRUs are detected, the micro-computer delivers an output in terms of a warning signal or sound through the audio-visual system in the vehicle. Automated control action will be taken over by the system if necessary. Therefore, this research focuses on the structure investigation and analysis of a new architecture of DNN to detect VRUs. A simple multilayer fully connected neural network (FCN) is developed as the baseline of this research to investigate the problems of the network needed to be improved. However, designing a VRUs detection system possesses many challenging issues, which are highlighted in the problem statement section.

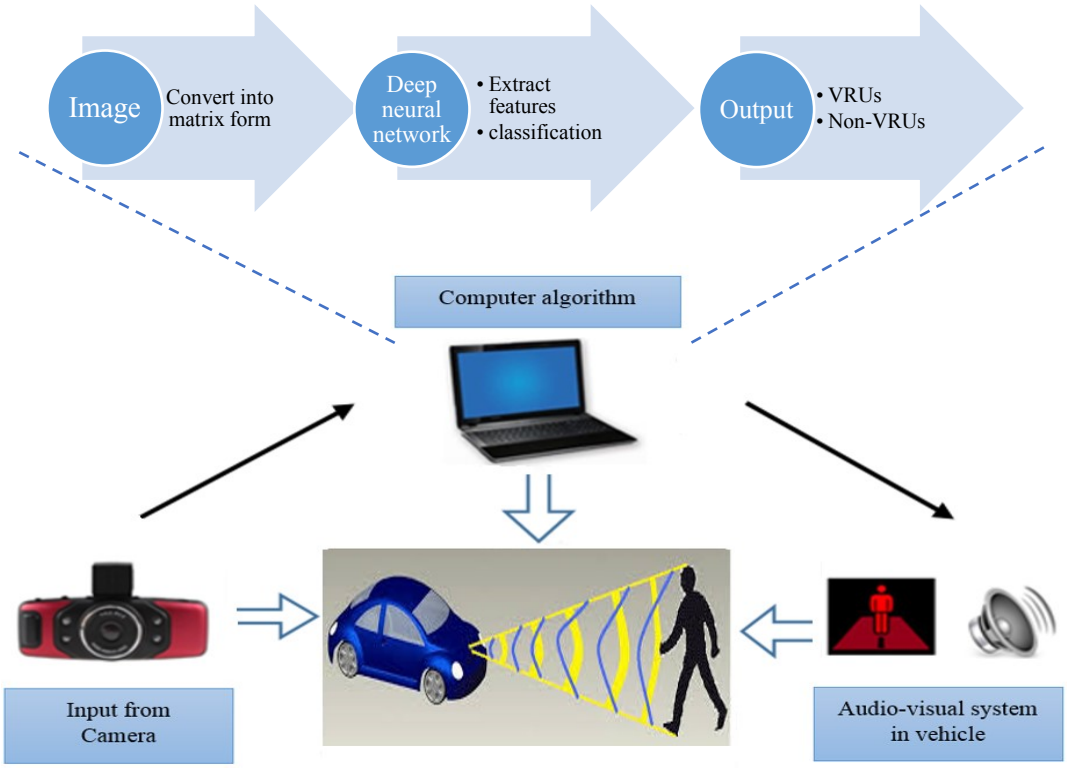


Figure 1.3: *Block diagram of the on-board vision system.*

1.2 Problem Statement

In recent years, VRUs detection is a key technology in a wide variety of applications such as video surveillance, robotics, and automotive safety. It uses a standalone camera associated with on-board computer vision algorithm to detect VRUs from a possible collision on the road ahead. The main technical challenge in the VRUs detection system is the rapid change of visual information obtained from a moving camera. Fast changing unconstrained outdoor conditions has increased the complexity of designing a VRUs detection system.

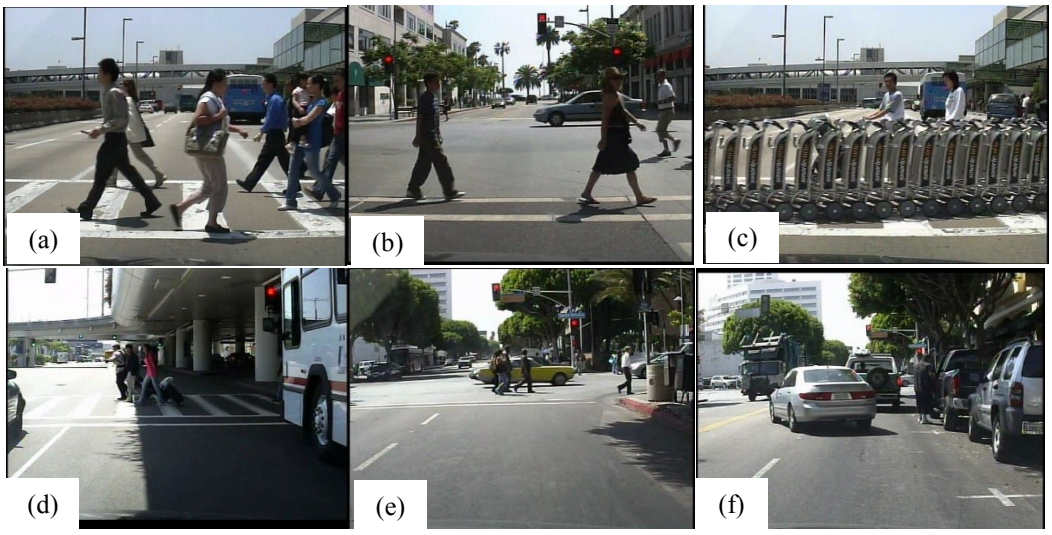


Figure 1.4: *VRUs in different road scene.* [10]*(a) Pedestrians carried different objects (b) Pedestrians with different walking poses (c) Partially occluded pedestrians (d) Lighting condition of the environment (e) Complex outdoor environment (f) Colour of the pedestrian and vehicle are the same.*

The challenges to design a VRUs detection pipeline are summarized as follows:

1. High variability in appearance among VRUs increases the difficulties of VRUs detection as shown in Fig. 1.4 (a) and (b). They exhibit high variability due to their changing poses, personal fashion, different objects carrying and different individual size.
2. VRUs can be appeared on the road at every possible corner in the road scene due to their size and flexibility as compared to civic structures by the roadside. They may suddenly emerge behind a vehicle or next to a building without a further indication. Therefore, Fig. 1.4(c) shows VRUs partially or fully occluded by obstacles such as vehicles, signboards, buildings and other roadside objects, which are challenging to be detected using DNN algorithm.

3. Variation of weather conditions such as cloudy day, raining day or sunny day incorporated with various human activities increases the difficulty of VRUs detection as shown in Fig. 1.4(d). The uncontrolled outdoor conditions such as shadows, may affect the images clarity and hence it decreases the accuracy of VRUs detection system.
4. Fast responding time for a real-time VRUs processing pipeline is required to quickly identify and localize a human object before a collision. Implementation of VRUs processing pipeline using DNN algorithm possesses great challenges in hardware processing and algorithm design. The design of DNN in detecting pedestrian must be light-weight and simple structure for real-time applications.
5. The development of VRUs detection system is dependent on the visual information recorded from a moving camera in a vehicle. Hence, the moving camera with rapid change of background scenes increases the complexity of VRUs detection and background segmentation. Fig. 1.4(e) shows the complex outdoor environment with pedestrians camouflaged in the background. Fig. 1.4(f) shows the similar appearance of a pedestrian standing next to a vehicle. Consequently, it is difficult to spot the pedestrian acquiring the same colour density as the vehicle or background.

In short, VRUs detection system requires an efficient and high flexibility of computer algorithm that can extract pertinent information from raw images to deal with high variability of VRUs appearance and complex outdoor conditions. Fig. 1.5 illustrated the current typical VRUs detection algorithm pipeline which relies on isolated feature extraction method to enhance the performance. However, developing feature extraction method is task-oriented and statistical-driven approach, which is not flexible for large variance of object samples. On the other hand, the use of DNN can learn the features from the data using learnable parameters. Hence, DNN is proposed in VRUs detection system as one of efficient algorithms to retrieve multi variants of objects using trainable features. Several objectives and contributions of this research are described in the next sections.

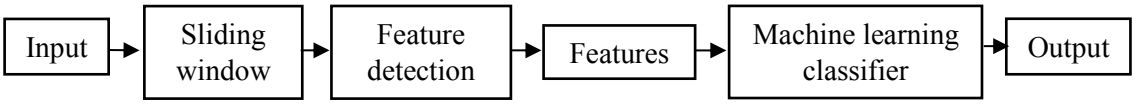


Figure 1.5: *Current VRUs detection algorithm*

1.3 Objective

The main goal of this research is to design a computer vision pipeline for vulnerable road user detection at arbitrary road scenarios using DNN algorithm. It can be further divided into three sub-objectives as follows,

1. To develop a deep fully connected network model as a baseline and investigate the relationship between the activation function and network performance.
2. To design a new DNN framework for single pipeline in VRUs detection.
3. To validate the detection performance of the new model using benchmark pedestrian datasets.

1.4 Contribution

The main contributions of this research are documented in detail in every chapters of the thesis. In chapter 3, a specific randomization technique is found to generalize the training process of DNN. This randomization technique is then applied in all classification experiments as the baseline in this research. Besides that, two types of deep neural network are implemented for the study of activation functions. Fig. 1.6 shows the component of activation functions used in the neurons of deep neural network. Deep belief network (DBN) is implemented to evaluate the saturated type of activation functions. Deep feedforward network (DFN) is implemented to evaluate unsaturated activation function and adaptive activation function. The main difference between these DNNs is that the requirement of extra pre-training process in DBN. The pre-training algorithm is used to eliminate the vanishing gradient problem occurred in saturated type of activation functions.

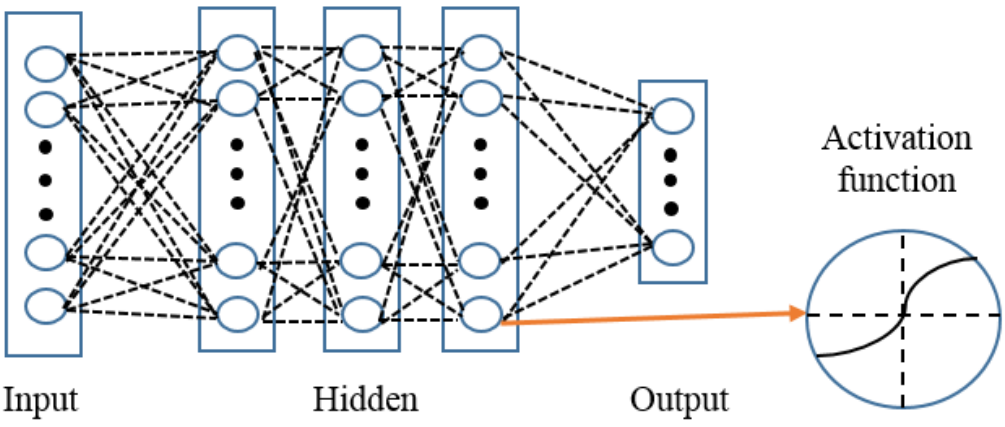


Figure 1.6: *Activation function in deep neural network.*

In chapter 4, a new framework of deep neural network (DNN) known as convolution deep feedforward network (C-DFN) is proposed to improve the detection rate. As demonstrated in Fig. 1.7, C-DFN uses a convolutional layer as a trainable feature extractor and four layers of standard DFN as a classifier. In addition, the performance of C-DFN is further investigated and compared with another network architectures such as DFN, DBN, and C-DBN. The overall test results show that the C-DFN outperformed the other types of networks. Besides, C-DFN with trainable activation function can gain better generalization and flexibility to solve the given classification tasks.

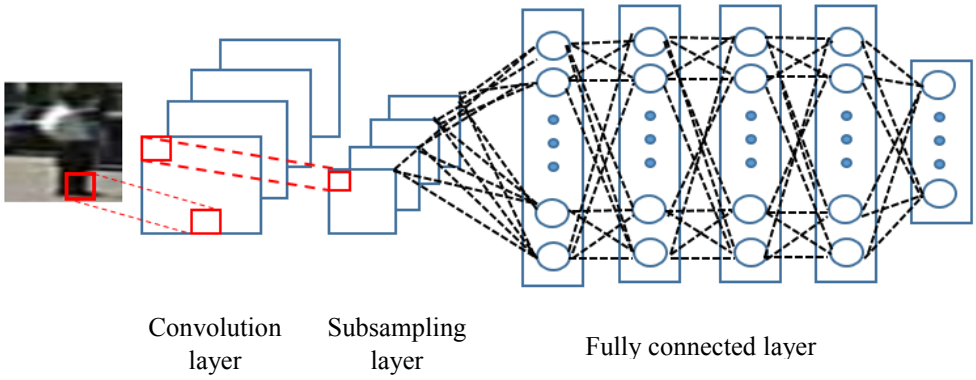


Figure 1.7: *C-DFN structure.*

In chapter 5, the novel learning algorithm known as C-DFN is implemented for VRUs detection task as shown in Fig. 1.8. C-DFN is further modified into C-DFN holistic VRUs detection, stacked C-DFN holistic VRUs detection, and stacked C-DFN part-based VRUs detection. These new frameworks are evaluated using benchmark Caltech pedestrian dataset. The experimental results show that the stacked C-DFN part-based VRUs detection system can detect medium scale VRUs even if the image resolution is low. Part-based C-DFN detects lower and upper part of a body to cast a vote in DNN to determine a human region when the human is occluded in the scene.

Figure 1.8: *Proposed new VRUs detection algorithm.*

1.5 Significance

Four significant contributions of the research are documented and highlighted in this thesis. First, the importance of dataset randomization with related to the training

process of DNN is explored to efficiently optimize the learning parameters in the network. The findings on this randomization techniques of input data can make a significant performance improvement to the network training. The randomness of the input data generalizes the DNN to learn meaningful features from the raw data through shuffling the input data. Based on the experimental result and findings, the randomization shall be applied to the overall training set and subsequently, this shuffled data is segmented into multiple mini batches of dataset. During the training phase, these mini batches are iteratively inserted to the network for parameter optimization. With every mini batch not having an equal number of input samples for each class, this randomization technique can generalize the training process in the stochastic manner.

Besides that, there is a need to include pre-training process if using saturated activation functions such as hyperbolic tangent function to eliminate the vanishing gradient problem during the network training process. The problem of vanishing gradient may cause the slowness of training process due to a small gradient element being backpropagated in the layers to update weights. In addition, this pre-training process is an extra mechanism if we can use only fine tuning process at one goal to update the weights. As a result, this problem can be mitigated by adding two trainable parameters into the hyperbolic tangent function and this trainable function is known as an adaptive tanh activation function. With the modification, the training phase can be completed without pre-training process and the vanishing gradient issue can be eliminated due to the adaptive capability of the activation functions.

To improve the feature learning in the neural network, a new framework known as convolutional deep feedforward network (C-DFN) is proposed to improve network performance. C-DFN uses a convolutional layer as a trainable feature extractor and four layers of standard DFN is connected as a classifier. The overall test results show that the C-DFN can outperform other variants of network architectures such as DFN, DBN, and C-DBN. Besides that, the C-DFN incorporated with trainable activation function can gain better generalization and flexibility to solve the given classification tasks.

Lastly, a single pipeline of VRUs detection system is proposed to deal with the VRUs variability and complex background. C-DFN utilizes the trainable feature

mechanism of DNN to reduce the process of VRUs detection system. By contrast, traditional VRUs detection system requires to do image processing on the input image and then utilizes hand-engineered feature detector to extract the important features in the image. C-DFN has a simple architecture as compared to the other types of DNN architecture. It can be easily to train and to be modified into a specific task-oriented applications. For VRUs detection application, C-DFN is modified into stacked C-DFN holistic VRUs detection model and stacked C-DFN part-based VRUs detection model. Stacked C-DFN does not required to retrain the whole architecture, therefore, it simplifies the process of training as compared to a complex DNN architecture.

1.6 Thesis Layout

This thesis provides a detailed study and development on deep neural networks for VRUs detection and evaluation. The structure of the thesis can be outlined into five chapters as follows:

Chapter 2: Literature Review

The previous work and state-of-the-art in vulnerable road user detection by using the monocular camera is discussed. It is further divided into three sections: holistic approach, part-based approach and deep learning approach. The research gap of the existing approaches is discussed at the end of the chapter.

Chapter 3: Activation functions in a Deep Feedfoward Network

The existing activation functions such as sigmoid, hyperbolic tangent (tanh), multistate activation function (MSAF), rectified linear unit (ReLU), leaky ReLU (ReLU), PReLU, and exponential unit (ELU) are implemented into the DFN. The relationship between the activation function and network performance is analysed and discussed in the experimental section.

Chapter 4: Convolutional Deep Feedforward Network

A detail description for developing convolutional layer (CL), max-pooling operation (PL), fully connected layer (FL) and their backpropagation training process are described in this chapter. Experiments are carried out to analyse and compare the misclassification rate of CL and FL. Experimental results based on MNIST dataset, INRIA person dataset and Daimler pedestrian classification dataset will be discussed.

Chapter 5: Vulnerable Road Users Detection

The proposed framework C-DFN is implemented into VRUs detection task. Detection dataset pre-processing, network training and testing process are discussed in detail in this chapter. C-DFN is further modified into holistic C-DFN, holistic stacked C-DFN and part-based C-DFN. The network performance is evaluated using the benchmark Caltech pedestrian dataset.

Chapter 6: Conclusion and Future Work

A summary of every chapter is drawn in this chapter to highlight the research contributions and significance of analysing results. The advantages and limitation of the proposed VRUs detection model are pinpointed with future improvement.

Chapter 2 Literature Review

2.1 Introduction

The research of vulnerable road users (VRUs) detection system has received increasing attention in recent years due to increasing demands in practical application, such as a smart surveillance system and on-board driving assistance system. A comprehensive review of the state-of-the-art in pedestrian detection system covers from the year 2009 until 2018 as reviewed in this chapter. The literature review mainly focuses on the computer vision algorithms related to using an optical camera. The general flow of detecting VRUs using a computer algorithm is shown in Fig. 2.1. Majority of the computer algorithm for VRUs detection systems use this processing pipeline order: (a) foreground segmentation, (b) classification (c) VRUs detection (d) tracking. Foreground segmentation is used to extract the region of interest (ROI) from the raw images captured by a camera. It extracts the possible candidates from the raw images and removes the background regions (e.g. sky, roads, and buildings).

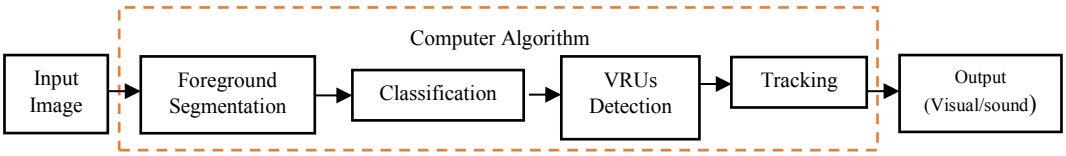


Figure 2.1: *VRUs detection process in computer algorithm.*

Sliding window method [11-13] is the simplest ROI selection procedure that can scan input image horizontally and vertically using bounding box shifting by a stride in classification. However, this method has two main disadvantages: 1) It slows down the classifier processing speed due to a large number of candidates are detected and 2) there are many irrelevant background regions such as sky and building in the scenes are included which would increase the potential number of false positive. Therefore, other methods are proposed to do the foreground segmentation such as those based on inter-frame motion and optical flow or based on the colour and gradient information to do ROI selection.

After that, the generated candidates are fed into a classifier to classify them as VRUs or non-VRUs. Feature extraction methods are applied to reduce the input dimension before input to the machine learning classifier. Histogram of Oriented Gradient (HoG) and Haar wavelet are the prevalent methods for feature extraction.

There are two types of classifier i.e. shallow learning classifier and deep learning classifier. Shallow learning classifiers such as Support Vector Machine (SVM) and AdaBoost consist only one hidden layer in its architecture. Shallow learning classifier can offer good classification result with simple architecture model [14]. However, it relies heavily on good feature extraction method to extract important information on the raw data [15]. Deep learning classifiers such as Convolutional neural network and deep belief network comprises of more than one hidden layer in its architecture. Deep learning classifier can learn to extract important information from the raw data automatically throughout the training process of the classifier [15].

Next, a tracking method is subsequently used to avoid false detection over time and predict the pedestrians moving direction. Kalman filter [14] and Particle filter [16] are the prevalent methods for tracking VRUs over time to deal with a high variety of multiple postures. If VRUs is detected, the computer system will send an output in terms of a warning signal or sound through the audio-visual system in the vehicle to alert the driver. This chapter begins with the general overview of VRUs detection process in a computer algorithm and followed by a brief introduction on benchmark datasets and the detection performance evaluation method. The state-of-the-art of VRUs detection system is then divided into three general approaches, i.e. holistic approach in section 2.2, part-based approach in section 2.3, and deep learning approach in section 2.4. After that, the research gap of the research will be discussed, and a summary will be concluded in the end of the chapter.

2.2 Benchmark Dataset

Public benchmark dataset for pedestrian has been published for performance evaluation over the years. These benchmark datasets are used for the researcher to do a comparison to other state-of-the-art proposals. Therefore, the new proposed algorithm can compare its performance with the benchmark databases using the same evaluation criteria. Four VRUs benchmark datasets i.e. INRIA [17], Daimler [18], Caltech [10], and TUD-Brussels [19] were discussed along with the evaluation metrics for pedestrian detection.

2.2.1 INRIA Person Dataset

INRIA [17] is the oldest benchmark pedestrian dataset among the others and has comparatively fewer images. The dataset is divided into two formats: a) original

images with corresponding annotation files, and b) normalized images which further divided into three image size: 64×128, 96×160 and 70×134 pixels. The positive training set contained 1208 image and the test set contained 566. The negative training set contained 1218 images and the negative test set 453 images. They included indoors, outdoor, city, mountain, and beach scenes. Some images also focused on cars, bicycles, and motorbikes as shown in Fig.2.2. This dataset is commonly used for training the classifier due to it high-quality annotations of pedestrians in a diverse scene.

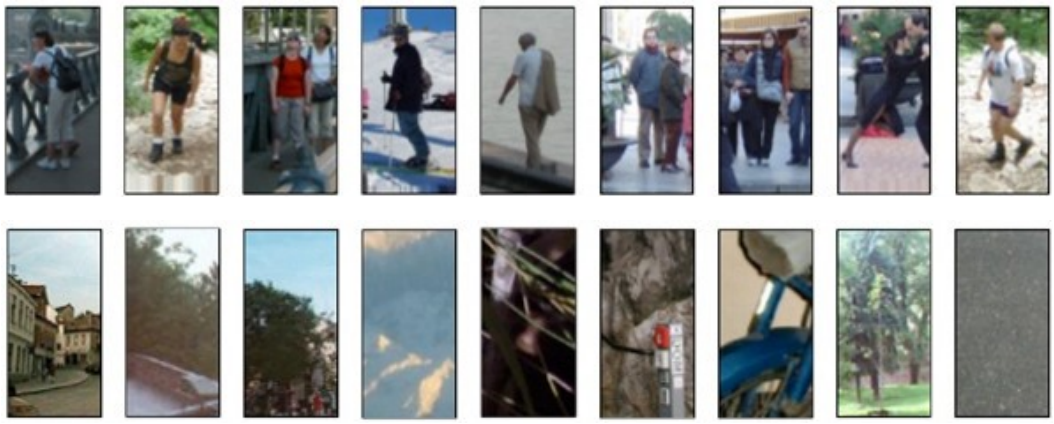


Figure 2.2: *INRIA's positive and negative samples. [17]*

2.2.2 Daimler Pedestrian Dataset

Daimler pedestrian classification dataset [20] is a pedestrian dataset provided by University of Amsterdam. The pedestrian data are observed from the on-board vehicle camera in mono, stereo and multi-cue. This dataset consisted of two parts, i.e. a) the base dataset and, b) additional non-pedestrian images. Base dataset contained 4000 pedestrian and 5000 non-pedestrian images which are sampled out from video. These images are subsequently scaled to the common size with a resolution of 18×36 pixels. From these image samples, pedestrian and non-pedestrian images were cropped and labelled manually using rectangular box. Video images were recorded at various (day) times and locations with no constraints on pedestrian pose or clothing, except that pedestrians were standing in upright position and were fully visible. As for non-pedestrian images, patterns representative of not containing pedestrians within a pedestrian classification steps were retrieved from known video images that have not to contain any pedestrians. On the other hand, an additional collection of 1200 images

not containing any pedestrians were included in the second part of the dataset. Fig.2.3 showed the sample images from Daimler dataset.

Figure 2.3: *Daimler pedestrian classification dataset positive and negative samples. [20]*

2.2.3 Caltech Pedestrian Detection Dataset

Caltech pedestrian dataset [10] is a benchmark pedestrian data provided by California Institute of Technology. It consists of approximately 10 hours of road scene video taken from an on-board camera in a vehicle driving through regular traffic in an urban environment. The video resolution is 640×480 in 30Hz capturing rate. It consists of six sets of training sample and each set of training sample contained of 6 to 13 one-minute long videos along with all annotation information. On the other hand, it also consists of five sets of testing data and each testing set contained 6 to 13 one-minute long videos along with annotation information. Fig.2.4 showed the sample images from Caltech dataset.

Figure 2.4: *Example image in the Caltech pedestrian dataset. [10]*

2.2.4 TUD-Brussels

This dataset is divided into training set and test sets which named as TUD-MotionPairs and TUD-Brussels respectively [19]. TUD-MotionPairs was recorded from a handheld camera at a resolution 720×675 pixels in a crowded area as shown in Fig. 2.5. It consists of 1092 positive images, 1776 annotated VRUs and 192 negative images. TUD-Brussels was recorded from a moving car in the city of Brussels. It contains 508 images with a resolution 640×480 and 1326 annotated VRUs. Table 2.1 showed a summary of the benchmark dataset.

Table 2.1: *Comparison of pedestrian detection dataset.*

Dataset	Publication year	Imaging setup	Colour image	Training		Testing	
				Positive image	Negative image	Positive image	Negative image
INRIA	2005	photo	√	1208	1218	566	453
TUD-Brussels	2009	video	√	1092	192	-	508
Diamler-DB	2009	video	×	4k	5k	-	21.8k
Caltech	2009	video	√	61k	67k	56k	65k

Figure 2.5: *Positive samples and optical flow field of TUD-MotionPairs. [19]*

2.2.5 Evaluation Metrics

As referring to the benchmark datasets, most of the evaluation methodology are developed based on Pattern Analysis, Statistical Modelling, and Computational Learning (PASCAL) criteria [21]. This PASCAL criteria is a standard image dataset

evaluation to analysis the accuracy of detectors to the ground truth comparison. A detection system takes a single frame of video as input and returns a bounding box (BB) and a score for each detection.

$$a_o \doteq \frac{area\left(BB_{dt} \cap BB_{gt}\right)}{area\left(BB_{dt} \cup BB_{gt}\right)} > 0.5 \tag{2.1}$$

where a detected Bounding Box (BB_{dt}) and ground truth Bounding Box (BB_{gt}). Pedestrian is detected if the interaction area and union area of the two BB exceeds 50%. There are four types of BBs that set to ignore as follows:

1) The height of BB is under 20 pixels high

2) BBs truncated by image boundaries

3) BBs labelled as "Person?"

4) BBs labelled as "People".

BB_{dt} falls in these regions will not affect detection system performance evaluation.

$$a_o = \frac{area\left(BB_{dt} \cap BB_{ig}\right)}{area(BB_{dt})} \tag{2.2}$$

However, BB_{dt} that matches to BB_{ig} will not be counted as true positives and BB_{dt} that is not matched to BB_{ig} will not be counted as false negatives. To further evaluate detectors performance, miss rate against false positives per image (FPPI) (log-log plots) is computed and plotted to compare the performance of detectors. The notation used for this evaluation can be displayed as follows,

1) BB_{dt} – bounding box output by a computer algorithm
2) BB_{gt} – bounding box provided by dataset
3) BB_{ig} – bounding box that set to be ignored during the performance evaluation.
4) True positive – BB_{dt} matched with BB_{gt}
5) False positive – BB_{dt} unmatched with BB_{gt}
6) False negative– BB_{gt} matched with BB_{dt}

The miss rate calculation as shown in Eq. (2.3) provided a measure of how well the detection system correctly rejects false positives.

$$Miss\ rate = \frac{False\ negative}{True\ positive\ + False\ negative} \tag{2.3}$$

Subsequently, detection rate is a measure of the percentage of true targets that was detected in the system as demonstrated in Eq. (2.4).

$$Detection\ rate = \frac{True\ positive}{True\ positive\ + False\ negative} \tag{2.4}$$

The false positive per image (FPPI) measured the count of the number of the bounding box output with the unmatched ground truth data as follows,

$$false\ positive\ per\ image = the\ number\ of\ \mathrm{BB}_{dt}\ \text{unmatched with}\ \mathrm{BB}_{gt} \tag{2.5}$$

2.3 Holistic Approach

Holistic approach detects VRUs by extracting whole candidates using a feature detection method as shown in Fig. 2.6. The extracted features were verified using a classifier to detect possible candidates. Feature detection method such as Histogram of oriented gradient (HoG) feature [17], Haar-like features [22], and SIFT [23] had been widely used to extract the overall shape of VRUs from the raw images captured by the camera before input the information into a classifier. Maji et al. [24] improved HoG VRUs detector by introducing a multiscale HoG and used intersection kernel SVMs for VRUs detection. Sabzmeydani and Mori [13] introduced the shapelet feature to extract mid-level shape features to enhance the detection system. Other than using gradient orientation of the image pixels, other information provided by the raw images such as colour [25], texture [26], and motion [27] were utilized to detect VRUs. Wojek and Schiele [28] showed that a combination of multiple features was able to improve the detection rate. Viola and Jones [22] were the first to combine appearance and motion information to detect VRUs. Lee et al. [29] combined HoG feature extraction method and motion detection to classify a detected motion as a VRUs or non-VRUs.

Schwartz et al. [30] utilized shape, texture and colour information and introduced a feature extraction method known as Partial Least Squares to reduce the dimensionality of the feature space. Hariyono and Jo [31] introduced an optical flow method to segment out moving objects from a static background. After that, HoG and

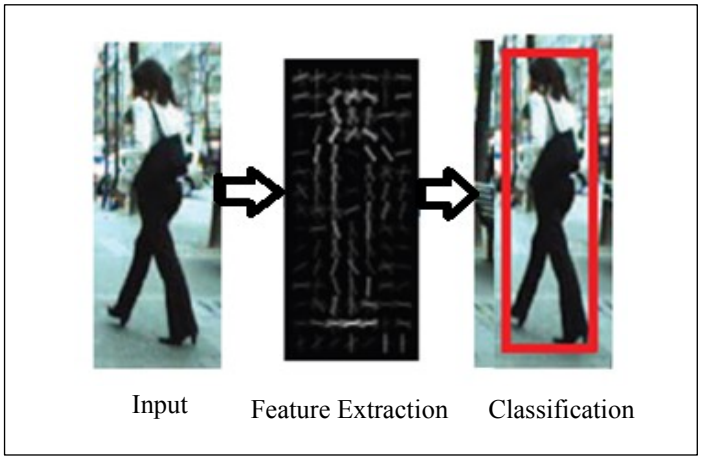


Figure 2.6: *General idea of holistic approach.*

SVM were used to extract features and perform classification. Liu et al. [32] introduced a new type of feature detection method known as multi-feature descriptor (MFD). They combined Optimal HoG, Local Binary Patterns (LBP) and Colour self-similarity in neighbour (NCSS) features to form MFD. MFD detects VRUs by utilizing gradient, texture and colour information. In addition, MFD with cascaded classifiers i.e. Adaboost rejecters and SVM classifier improved the speed and accuracy of the detection performance in INRIA dataset.

Dollar et al. [25] improved the computation of local sum in feature detection methods by using an image processing method known as integral image. Dollar et al. [33] re-implemented detector [25] with a focus on low-level features optimization. They shorten the computational time around 5-10 times of the original detector [25] using multiscale gradient histogram approximation. Nam et al. [34] improved VRUs detection performance by introducing macro feature layout selection to improve the localization of the VRUs. Dollar et al. [35] introduced a new type of feature extraction method known as feature mining to discover pertinent features automatically. Dollar et al. [36] had made a great contribution by introducing a fast and accurate feature detector known as the aggregate channel feature (ACF). They improved the accuracy of the detector by computing feature at octave-spaced scale intervals instead of approximate features on a finely sampled pyramid. However, this method is only applicable to static images.

Benenson et al. [37] investigated different factors affected the performance of integral channel features detectors family. They improved integral channel features by combining HoG, colour features, multi scales technique and global normalization

named as *Roerei* detector. It successfully outperformed other detectors including non-linear SVMs, deformable part-based model and also deep learning architectures. Zhang et al. [38] investigated on the newly developed feature extraction method such as SquaresChntrs [37], InformedFilters [39], locally decorrelated channel feature (LDCF) [40] to identify the effect of selecting different filter and its detection performance. They introduced a modified version of InformedFilters named checkerboards and combined with SDt [27] named it as "all-in-one". By adding optical flow on the "all-in-one" method, they successfully reduced the miss detection rate to 17.1% on Caltech pedestrian dataset.

Zhang et al. [41] found that the precision of current detectors could be improved by a centre-surround contrast features for VRUs detection which motivated by human vision mechanism. This work achieved 15.90% of miss detection rate on INRIA dataset and 34.96% of miss detection rate on Caltech dataset. Ding and Xiao [42] designed a multiscale context feature by combining the local window with neighbourhood windows. They introduced contextual boost classifier to exploit the response in the local neighbourhood across multiple scales. They reduced FPPI in Caltech dataset from 50% to 48%. Costea and Nedevschi [43] introduced a detector which does not require image resizing and using only one classifier for all sliding window scales. They used a high-level feature known as Word Channels inspired from codebook with the advantage of scale independent behaviour of the normalized feature.

Other than creating new feature detection method, another interesting method was using multiple classifiers concatenate together to build a faster and more accurate detector. Viola and Jones [22] introduced "Integral Image" for fast computation and cascaded AdaBoost to improve the detector accuracy. Dollar et al. [44] implemented detector [33] and introduced crosstalk cascade which allows neighbour detector to communicate and shorten the computational time. Leithy et al. [45] cascaded Haar based classifier, Co-occurrence HoG (Co-HoG), and SVM classifier to improve the speed and accuracy detection system for static images. Co-HoG is able to extract features from multi-scale shape information. However, it required high computational time due to its high dimensional pattern descriptor. Therefore, they overcome this problem by using independent component analysis (ICA). The detector was evaluated in INRIA and Daimler Chrysler benchmark dataset. The experimental result showed

Table 2.2: *Holistic approach summary*

VRUs detector	Miss rate %	Feature types	Dataset
VJ [22]	94.73	Haar	INRIA
Shapelet [13]	91.37	Gradients	INRIA
PoseInv [11]	86.32	HoG	INRIA
FtrMine [35]	74.42	HoG+Colour	INRIA
HikSvm[46]	73.39	HoG	INRIA
HoG [17]	68.46	HoG	INRIA
MultiFtr [28]	68.26	HoG+Haar	INRIA
Pls [47]	62.10	HoG +Colour frequency	INRIA
MLS [34]	61.03	HoG	INRIA
MultiFtr+CSS [48]	60.89	HoGF+CSS	TUD
pAUCBoost [49]	59.66	HoG+COV	INRIA
FPDW [33]	57.40	HoG+LUV	INRIA
ChnFtrs [25]	56.34	HoG+LUV	INRIA
Crosstalk [44]	53.88	HoG+LUV	INRIA
ACF [36]	51.36	HoG+LUV	INRIA
RandForest [50]	51.17	HoG+LBP	INRIA + Caltech
MultiFtr+Motion	50.88	Many+Flow	TUD
SquaresChnFtrs [37]	50.17	HoG+LUV	INRIA
MultiResC [51]	48.45	HoG	Caltech
Roerei [37]	48.35	HoG+LUV	INRIA
ACF-Caltech [36]	44.22	HoG+LUV	Caltech
MultiResC+2Ped	43.42	HoG	Caltech
WordChannels [43]	42.30	Many	Caltech
ACF+SDt [27]	37.34	ACF+Flow	Caltech
SquaresChnFtrs [37]	34.81	HoG+LUV	Caltech
InformedHaar [39]	34.60	HoG+LUV	Caltech
ACF-Caltech+ [36]	29.76	ACF	Caltech
SpatialPooling [52]	29.24	pAUC	Caltech
Spatial-Pooling+ [52]	21.89	pAUC	Caltech

in INRIA and Daimler Chrysler benchmark dataset. The experimental result showed that the detector can reduce 0.001% computational time if compared to using a Co-HoG pattern descriptor alone.

Wang et at. [53] cascaded AdaBoost detector and random vector functional-link net to reduce the false positive per frame rate in VRUs detection. Cascade classifier combined different classifier advantages to improve classification results. Ke et al. [54] enhanced ACF by introducing principal component analysis filter based convolutional channel features (PCA-CFF). It cascaded with Adaboost classifier and reduced miss detection rate to 14.24% on INRIA person dataset. Guo et al. [55] adopted a two-stage holistic detection system comprising of cascaded AdaBoost to generate the region of interest from input images in the first stage. In the second stage,

SVM classifier was used to determine the presence of VRUs in the region of interest. The experimental results showed that better performance was achieved with a two-stage classification than a single classifier approach. Table 2.2 summarized the holistic approaches discussed in this section. The drawbacks of using a holistic approach in single or cascaded classifiers have highly relied on feature extraction method and the degraded performance due to complex background clutters and occlusion problems.

2.4 Part-based Approach

The part-based approach is a method used to overcome the complex background clutter and occlusion problems in the VRUs detection system. The general idea of a part-based approach is shown in Fig. 2.7. This approach treats VRUs as an assembly of a different body part. They run detection on individual body parts and then combine the detection results to achieve classification on the whole VRUs figure. It is segmented into several body parts based on standard body ratio: 16% for head and neck, 34% for the torso, and 50% for legs [56]. Each part of the candidate is delivered to an individual classifier to verify the candidate is a VRUs or not. There are two ways to make a decision based on the output of each classifier: (1) Majority vote, (2) regression output classification. Majority vote method uses the binary output to represent the detection result for each part of the body [57]. A VRU is detected if at least two out of three classifiers label the parts as VRUs [57]. Regression output classification uses one more classifier to label the parts for VRUs detection [17].

Deformable part-based models (DPM) utilize the fact that the overall figure of the pedestrian might change, but each part of the VRUs (e.g. arm, leg) does not change significantly. DPM optimized the body part model holistically from the root (head) to terminals (hands and legs). DPM is often had better performance than simpler holistic models such as rigid templates using HoG feature detector [17]. Zhang et al. [58] handled the occlusion problem by learning spatial co-occurrence relations using a hierarchical co-occurrence model. They used latent SVM to generate visibility statuses and used the random forest as a classifier. They improved 5% of miss rate on three benchmark dataset, i.e. INRIA [17], ETH [59], Caltech dataset [10].

Bar-Hillel et al. [60] extend further feature mining method [35] to detect partial occlusion VRUs. They introduced SVM predictive feature selection method to

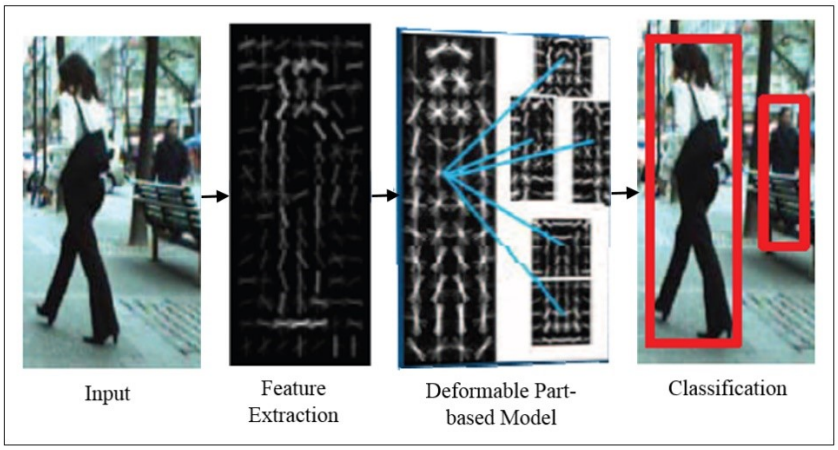


Figure 2.7: *General idea of part-based approach.* [61]

used the HoG to extract the shape boundaries of the VRUs and the shapes were segmented by probabilistic hierarchical part-template tree model. Kernel SVMs were used as classifiers to classify VRUs and non-VRUs. Wang et al. [26] combined HoG and LBP to tackle the partial occlusion problem. They used HoG feature response to find the occluded part of the image. Then, the occlusion likelihood map is segmented by mean shift-approach. Part-detectors were applied to the visible regions to achieve the final classification on the current window. Marin et al. [50] combined part-based model and a fast execution time classifier known as random forest classifier to detect VRUs. The part evaluation was replaced by the decision tree so it could be adapted to the different VRU viewpoint and body poses.

Felzenszwalb et al. [12] solved the detecting and localizing VRUs by using a multiscale deformable part model. They introduced latent SVM to utilize the latent information for VRUs detection task. Ouyang and Wang [62] utilized a deformable part-based model (DPM) to extract the unique visual patterns illustrated in multiple nearby pedestrians effectively. This method decreased the miss detection rate on the benchmark Caltech dataset from 48% to 43%. Ouyang et al. [63] used Bayesian theorem to create a new probabilistic framework and a mixture of DPM to design a two pedestrians detector. This research reduced the misdetection rate from 37% to 32% on the benchmark Caltech dataset. Mittal et al. [14] utilized DPM to detect heavy occluded VRUs and reduced miss detection rate to 27.2%. The miss detection rate was further reduced to 17.7% by using a Kalman filter for tracking the candidates on benchmark Caltech dataset.

Li et al. [64] introduced a part-based detection approach by using HOG-SVM to create a hierarchical grammar model with the AND-OR graphical structure to represent the composition of pedestrians. The experimental results showed that this detection system could achieve stable detection performance in a complex traffic environment. However, this is only applicable with high-resolution images. Yan et al. [65] also utilized DPM to jointly encode the similarity and the differences among the pedestrians with different resolutions. They tackled the problem of lower resolution that usually caused the significant drop of the VRUs detection system's performance.

Pedersoli et al. [66] introduced a hierarchical multi-resolution part-based model. They computed the HoG features at a different resolution to obtain a pyramid of HoGs. Then, they extended Coarse to fine (CtF) feature detection method into a deformable part-based model. Ctf feature detection refers to scanning the detected region in every layers of the pyramid of HoGs. This localization evaluation method saved computational time from scanning the whole image. Mathias et al. [67] improved integral channel feature (ChnFtrs) detector by using spatially biasing feature selection and 16 occlusions specific classifier to handle occlusion problem. ChnFtrs is a feature detection method uses integral image to extract important information from the image. Park et al. [51] introduced a multiresolution part-based model. They added ground plane estimation as a latent variable to be estimated for each frame of a video. It reduced the detection rate from 43% to 40%.

Zhang et al. [68] designed a multimodal and multichannel Haar-like features to form a statistical pedestrian part-based model. The miss detection rate obtained by this research was 14.43% on INRIA dataset and 34.60% on Caltech dataset. Chen et al. [69] extended Contextual Boost [42] detector by introducing Multi-Order Contextual co-occurrence (MOCO) to the model high-level context using one classifier. They reduced the miss rate from 48% to 46% in Caltech dataset. Zhang et al. [39] designed a pool of rectangular templates that are tailored to this shape model and used Haar–like features to describe local differences. This detector avoids exhaustive scan on the image thus reduces the dimensionality of the input data for classification. They reduced the Caltech dataset miss rate to 34.60% and 14.43% on INRIA dataset.

Chen et al. [70] used SIFT to obtain feature matrix from a sliding window. They used multiple mapping vectors to project the original feature matrix into different

Table 2.3 : *Part-based approach summary*

VRUs detector	Miss rate %	Feature types	Dataset
LatSVM-v1 [12]	79.78	HoG	Pascal
HoGLBP [26]	67.77	HoG+LBP	INRIA
AFS+Geo [71]	66.76	Custom (AFS)	INRIA
AFS [71]	65.38	Custom (AFS)	INRIA
LatSVM-V2 [72]	63.26	HoG	INRIA
FeatSynth [60]	60.16	Custom	INRIA
Zhang et al. [58]	60.0	HoG	Caltech
RandForest−HOGLBP−CGP [50]	51.0	HoG-LBP	Caltech
Franken [67]	48.68	HoG+LUV	INRIA
MF+Motion+2Ped [73]	46.44	Many+Flow	INRIA
MOCO [69]	45.53	HoG+LBP	Caltech
MT-DPM [65]	40.54	HoG	Caltech
MT-DPM+Context [65]	37.64	HoG	Caltech
Zhang et al. [68]	34.60	Haar-like features	Caltech
ACF+SDt+2Ped [63]	32.0	HoG	Caltech
MultiresC [51]	28.9	HoG	Caltech

mask spaces and used MI-SVM to perform classification. Enzweiler et al. [74] introduced a mixture-of-experts framework to classify partially occluded VRUs. They trained a set of component-based expert classifiers on features derived from the intensity, depth, and motion. Occlusion problem was solved by computing expert weights based on visibility estimation from depth and motion information. The expert weights were the trained weights from different component-based classifiers.

As a conclusion, DPM could detect pedestrians by finding each part of the VRUs model individually and combining them. However, it requires high-resolution images to achieve satisfactory detection performance. The part-based approaches summarized in Table 2.3 demonstrated the classification task using shallow architecture such as SVM and AdaBoost. The limitation of shallow architecture is that it does not have good data generalization in all dataset or nature scenes in the real-world applications. Therefore, more research activities were explored towards the feature extraction methods to improve the detection performance. In the following section, the combination of the part-based model and deep learning approach are discussed.

2.5 Deep Learning Approach

Deep learning approach as shown in Fig. 2.8 combined the part-based approach that performs well in detecting occluded pedestrians and a multi-layer classifier (i.e.

deep learning model) to detect VRUs. Ouyang and Wang [75] utilized a probabilistic detection method together with DPM. Ouyang et al. [76] introduced a mutual visibility deep model that jointly estimated the overlapping pedestrian. They verified the visibility of a part multiple times at different layers of the deep model and improve the accuracy of the detector. Deep model learned the visibility relationship between the pedestrians to detect the overlapped pedestrian. This mutual visibility deep model reduced miss detection rate to 48% in Caltech dataset.

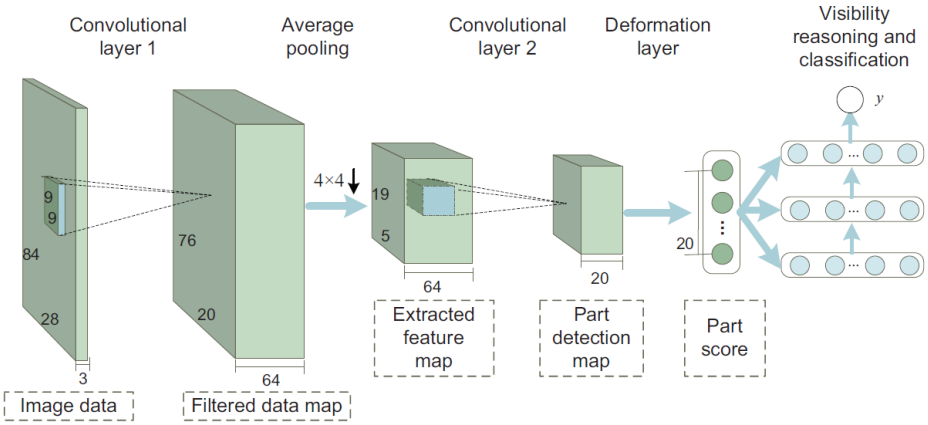


Figure 2.8: *Example of deep learning approach.*

Zeng et al. [77] utilized an unsupervised pre-training method to avoid the overfitting problem. This method reduced the miss detection rate to 45% on Caltech dataset. Ouyang et. al. [61] used a deep convolutional neural network to jointly learn the pedestrian detection processing stage and to establish an automatically mutual interaction between all stages. This method reduced the miss detection rate to 39% on Caltech dataset. Sermanet et al. [78] used convolutional sparse coding to obtain multistage features instead of using hand-engineered features detection method. Luo et al. [79] enhanced convolutional neural network (CNN) by introducing a switchable restricted Boltzmann machine (SRBM) to model the mixture of body parts and hierarchical feature representation. It reduced the miss detection rate to 37.87% on Caltech dataset.

Angelova et al. [80] modified Krizhevshy et al. object detector [81] to form a new framework known as a large field of view deep network (LFOV). They cascaded their deep network with a CNN to speed up the run time to 280 ms per image on the graphical processing unit (GPU). This research reduced the miss detection rate to 35.85% on Caltech dataset. Wang and Zhang [82] combined a modified integral channel

features with a CNN and introduced a boost-like structure in the CNN to improve the detection structure stability and convergence speed. This research achieved 35.51% of miss detection rate on Caltech dataset. Benenson et al. [83] combined SquaresChnFtrs, DCT, SDt, and 2Ped detector and renamed it as Katamari-v1 reduced Caltech dataset miss rate to 22.49%.

Hosang et al. [84] showed the effectiveness of the CNN framework in VRUs detection. They analysed small and big CNNs, their architectural choices, parameters, and the influence of different training data, including pretraining on surrogate tasks through a wide range of experiments. Girshick et at. [85] used an unsupervised method based on convolutional sparse coding to pretrain CNN for VRUs detection. Tian et al. [86] introduced a deep model that consists of extensive part detectors. However, VRUs detector based on CNN is known to be very slow. They need multiple GPUs run in parallel to shorten the computational time.

Tian et al. [87] introduced task-assistance CNN that jointly learned semantic tasks using multiple benchmark dataset. They utilized ACF [36] to extract hard negative samples and pruning the candidates' window in the training stage. It reduced the miss detection rate to 20.86% on Caltech dataset. Liu et al. [85] introduced a spatially weighted max-pooling in the CNN to model high-level semantic features. This work reduced the miss detection rate to 12.82% on INRIA dataset, and 30.17% on TUD-Brussels dataset. Tome et al. [88] evaluated deep CNN (DCNN) i.e. AlexNet and GoogleLeNet and combined with different region proposal method i.e. HoG, ACF, LDCF, and selective search. LDCF with DCNN reduced miss rate to 19.9% with higher computational time. Therefore, they introduced a combination of ACF with DCNN (DeepPed) which is capable to archive real-time using a supercomputer with 192-core NVIDIA Kepler graphical processing unit (GPU), NVIDIA quad-core ARM Cortes A15 central processing unit (CPU). However, they did not show the miss rate of this detector.

Jung and Hong [89] used the AdaBoost algorithm i.e. ACF and ECF to extract proposal samples and using DCNN model VGG-16 network for VRUs detection. They introduced a guiding network which uses two fully connected networks to classify the ROI selected from DCNN. They reduced the miss rate to 7.95 % on Caltech dataset. Li et al. [90] combined Edge Boxes feature detector and Region-based CNN (RCNN)

to extract the region of interest (ROI) from the input image. They used a linear SVM as a classifier to identify the candidates in ROI. They reduced the miss rate to 23% on INRIA dataset. Jiang et al. [91] introduced share features across a group of CNNs corresponding to VRUs models of different scales. They detect VRUs using multiple scales on one layer of an image pyramid. They found that if using four detectors the miss rate on Caltech dataset is increased and the computational time can be increased when the number of the detectors is increased. Deep learning approach combined with the part-based model improved greatly on the VRUs detection problem. However, the existing works are largely dependent on the image processing and feature extraction method. The detection approaches are summarized in Table 2.4 and the research gap of the current VRUs detection approaches is discussed in the following section.

Table 2.4: *Deep learning approach summary*

VRUs detector	Miss rate %	Feature types	Dataset
ConvNet [90]	77.20	Pixels	INRIA
DBN-Isol [75]	53.14	HoG	Caltech
DBN-Mut [76]	48.22	HoG	Caltech
MultiSDP [77]	45.39	HoG + CSS	Caltech
JointDeep [61]	39.32	Colour + gradient	Caltech
SDN [79]	37.87	Colour + Sober edge detector	Caltech
BDL [82]	35.51	HOG+CSS	Caltech
AlexNet [84]	32.38	SquaresChnFtrs	Caltech
LDCF [40]	24.80	Custom (Modified ACF)	Caltech
Katamari-v1 [83]	22.49	HoG+Flow	Caltech
TA-CNN [87]	20.86	ACF	Caltech
DeepPed[88]	19.90	LDCF	Caltech
Checkerboards [38]	18.47	Custom	Caltech
Jung and Hong [89]	17.97	ACF	Caltech

2.6 Research Gap

The afore-mentioned VRUs detection approaches are still far from better accuracy with a simple processing pipeline. Most of the detectors are still rely on hand-engineered features methods such as shape features, colour features and gradient features. VRUs detector based on shape feature is not applicable due to the variability of the shape of VRUs. It requires a large number of features that manually labelled at the training stage to cover all the possible poses and shape of VRUs. This is highly time-consuming and may be computationally intractable for real-world application. Majority of the VRUs detectors were relied on gradient information obtained using

HoG. The capability of the gradient information to represent richer image patterns is limited. In addition, such hand-engineered features are usually optimized only for certain tasks. Thus, they are hard to be adaptive to different tasks at hand. While low-level features can be designed manually, higher-level features that can capture richer information are difficult to engineer. Therefore, more attention should be the focus on building higher-level features which are more adaptive and can provide complementary information to the manually designed features.

Nowadays, researchers implemented deep architecture neural network into VRUs detection system to build higher-level features to improve the performance of VRUs detection system. However, most of the existing deep learning approach for VRUs detection still rely on hand-engineered feature detector to detect VRUs as shown in table 2.4. The main motivation of deep architecture neural network is to extract the pertinent information from the raw images captured by the camera. In addition, it learns the low-level features and high-level features automatically which is known as trainable feature mechanism. However, as shown in Table 2.2, it showed that the majority of the detection systems were using a hand-engineered feature detection method rather than a trainable feature mechanism to extract the pertinent information from the raw images.

In a nutshell, VRUs detection has limitations on its detection pipeline using trainable feature mechanism. This research is aimed to develop a new framework of DNN that is able to detect the VRUs without using hand-engineered feature detection method. The advantages of using a trainable feature mechanism are better generalization in VRUs detection and high reliability of adaption towards wide range of the real-world application involving natural images and visual scenes. Therefore, a new framework of deep architecture neural network shall be proposed to handle various kinds of real-time road scenarios while retain good VRUs detection accuracy.

2.7 Summary

A comprehensive review of the existing VRUs detection approaches had been done in this chapter. It is divided into three categories: holistic approach, part-based approach, and deep learning approach. Holistic approach detects VRUs using a feature detector to search for a whole pedestrian in the raw input images. After that, the possible candidates are then put into the classifier to determine whether it is a VRUs

or not. The drawback of this approach is that it does not detect the VRUs when the VRUs is occluded. Part-based approach solved this problem by dividing the possible candidates into several body parts for classification. However, the part-based approach would not generalize well due to the limitation of shallow architecture neural network that uses as a classifier to detect VRUs. Therefore, a deep learning approach is introduced. Deep learning approach provides better generalization via deep architecture neural network with the combination of part-based approach. Deep architecture neural network can learn features from raw pixels; therefore, it is more robust to noise and copes well with the natural scene in real-world application. However, most of the deep learning approach in VRUs detection system were still using hard-engineered features such as HOG instead of using trainable feature mechanism to detect VRUs in the raw images. Therefore, this research is aimed to propose a new framework of deep architecture neural network to detect VRUs without using image pre-processing method and feature extraction method. In the next chapter, a simple deep fully connected network is developed as a baseline to understand the deep learning classifier behaviour. The relationship between the activation functions and the network performance is investigated and discussed in detail.

Chapter 3 Activation Functions in a Deep Feedforward Network

3.1 Introduction

Deep neural network (DNN) is a computer algorithm that mimics our human brain neural network to solve many real-world applications. DNN refers to more than one hidden layer artificial neural network (ANN). A multiple layer DNN can learn to detect important features automatically. Thus, DNN is robust to solve the complicated task such as object recognition [92], speech recognition [93] and voice analysis. DNN can be further divided into three categories: discriminative, generative, and hybrid architectures. Discriminative type is referred to a supervised learning algorithm which learned a statistical model for estimating an output if there is an existence of input image. Generative type of architecture is referred to un-supervised learning algorithm which learned a joint probability of data distribution. Hybrid type is the combination of generative and discriminative architectures. The goal of hybrid architecture is discriminative, but it is assisted with the outcomes of generative architectures.

The training process of the DNN is challenging because of the vanishing gradient problem and the saturation problem of the activation function in the DNN [94]. Vanishing gradient problem causes DNN learns ineffectively when the gradient value is backpropagating through a hidden layer becomes smaller. This problem is mainly because of the saturation problem of the activation function in DNN. This problem occurs when the neurons reach near the "near-horizontal" part of the curve on either side, the error gradient becomes very small or "vanished" [94]. Therefore, the network refuses to learn further or is drastically slow during the training of the network. Various research is therefore developed to deal with the problem of vanishing gradient. Hinton [95] introduced a hybrid DNN known as Deep Belief Network (DBN) which using a greedy layer-wise pre-training algorithm to solve the vanishing gradient problem in DNN. This unsupervised pre-training algorithm successfully overcomes the vanishing gradient problem by initializing the weights parameter using a pre-training process.

Beside the weight's parameter initialization, the activation function plays an important role in the DNN training process. Activation functions can be categorized into three groups, i.e. saturated activation function, unsaturated activation function and

adaptive activation function. Saturated activation function was the first type of decision boundaries used in DNN. A Sigmoid function is the default activation function used in ANN due to its bounded differentiable characteristic with the positive-valued output. The main disadvantage of the sigmoid function is the output value is not zero-centered and vanishing gradient problem. Cai et al. [96] introduced multistate activation function (MSAF) that had additional states on its activation function. Xu et al. [97] presented leaky hyperbolic tangent (tanh) activation function by adjusting the slope in negative region to solve the vanishing gradient problem.

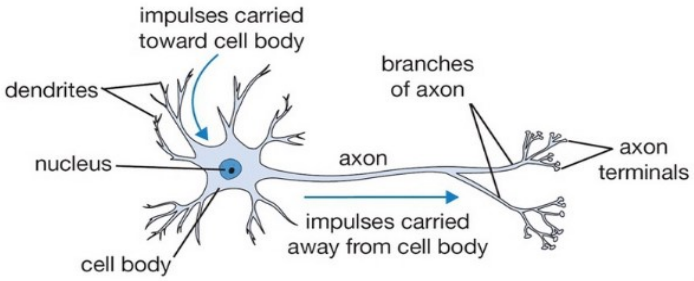
Unsaturated activation functions for instance rectified linear unit (ReLU) [98] does not have saturation problem and vanishing gradient problem during the DNN training process. In addition, it has no exponential terms in the activation function thus, it requires less computational time. The main drawback of ReLU is when the data falls into the negative region of ReLU, the DNN's node cannot be re-activated during the training process [99]. Thus, it slows down the learning process of the DNN. Leaky ReLU (LReLU) is proposed to solve this problem by assigning a constant variable in the negative region of the ReLU [99]. Adaptive activation function such as parametric ReLU (PReLU) proposed by He et al. [8] adaptively changed the variable of the negative region in the ReLU according to the training algorithm. Jin et al. [100] introduced an adaptive piecewise linear activation function (APL) that used the piecewise linear functions to estimate any function during the training process. Goodfellow et al. [101] introduced maxout activation function to improve dropout regularization approach. The maxout activation function added the trainable parameters in its function to achieve better recognition rate.

Nair and Hinton [98] proposed an unsaturated activation function known as a ReLU to improve DBN performance and solved the vanishing gradient problem without using the pre-training algorithm. Unsaturated activation function also provides sparsity in the hidden unit of the network. Unsaturated activation function like ReLU deactivated some of the neurons in the hidden layer of the network to reduce excessive neurons, thus reduces the chance of overfitting on the training data. This breakthrough has enabled the era of deep learning in artificial intelligence society to solve many real-world applications. Due to ReLU simplicity and effectiveness, it has become the default activation function in DNN.

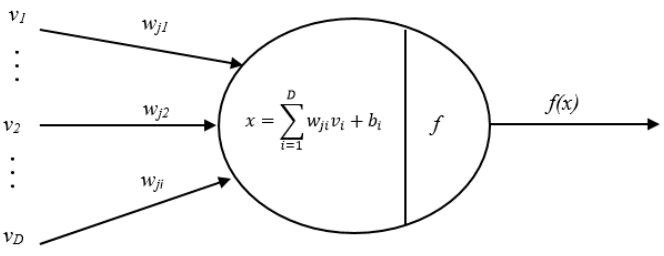
This chapter compares and analyses the existing activation functions such as sigmoid, hyperbolic tangent, multistate activation function, ReLU, LReLU, PReLU, and exponential unit by implementing them into the deep feedforward network (DFN). This chapter is organized as follows: section 3.2 introduces different types of activation functions and the proposed activation function. In the next section, a series of randomization techniques are investigated which is required for the experiment setup in section 3.4. In section 3.5 the detail evaluations of activation functions are performed using MNIST dataset. Finally, a conclusion is drawn in the end of the chapter.

3.2 Activation Functions

A neural network is made up of a number of processing elements known as neurons shown in Fig. 3.1(a). Each neuron received inputs from external world or from the outputs of other neurons. The interconnections between the neurons are called synapses. The synapses have a processing value called weights. The dendrites attached with the neurons' cell body act as input/output channels all input from other neurons activities through dendrites. If the internal electric potential of the cell raises, the neuron delivers a signal through axon to other neurons.



(a)



Input Artificial neuron Output

(b)

Figure 3.1: *(a) Biological neuron (b) Mathematical model of artificial neuron.*

The mathematical model of the biological neuron is presented in Fig. 3.1(b). Each artificial neuron has an activation function to determine the firing rate of the neuron to activate. Moreover, it provides non-linearity property to the ANN and bounded the input data to a finite value [9]. In Eq. (3.1) the input data $\boldsymbol{v}$ is multiplied with weights $\boldsymbol{w}$ parameter and summed up with an additional bias, $\boldsymbol{b}$. An activation function f is used to decide whether that specific artificial neuron is been activated.

$$y_i(\boldsymbol{x}) = f\left(\sum_{i=1}^{D} w_{ji} v_i + b_i\right) \tag{3.1}$$

Activation function can be divided into two categories, i.e. saturated activation function and unsaturated activation function. Saturated activation function was common in use for decision boundaries since the ANN was introduced. There are two important features of the activation function in a DNN: (1) Activation function limits the amplitude range of the output signal to a finite value, (2) It provides non-linearity property to estimate any given task. Various types of activation function such as sigmoid, hyperbolic tangent, multistate activation function, rectified linear unit, and exponential unit is described in detail in the following section.

3.2.1 Logistic function

The logistic function is the most common activation function used in an ANN. It has a similar representation to a biological neuron's firing rate [9]. Sigmoid function, f_1 and hyperbolic tangent (tanh), f_2 are the two types of logistic activation function variants. Logistic function is a continuous function that demonstrates a balance between

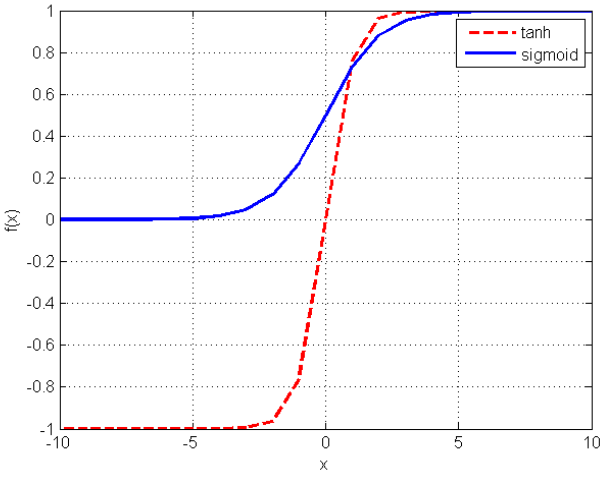


Figure 3.2: *Logistic function.*

linear and nonlinear behavior as shown in Fig. 3.2. The sigmoid function assumes a continuous range of values from 0 to 1. The sigmoid function, f_1 is expressed as follows:

$$f_1(x) = \frac{1}{(1+\exp(-x))}, \tag{3.2}$$

where $x = \sum_{i=1}^{D} w_{ji} v_i + b_i$. Differentiating Eq. (3.2) with respect to $\boldsymbol{x}$ denotes,

$$f_1'(x_j) = \frac{\exp(-x)}{[1+\exp(-x)]^2}. \tag{3.3}$$

Substitute $y_j(\boldsymbol{x}) = f_1(x)$ to eliminate the exponential term denotes,

$$f_1'(x_j) = y_j(\boldsymbol{x})(1 - y_j(\boldsymbol{x})) \tag{3.4}$$

Recently, the sigmoid function has fallen out of favour as the activation function in the neural network due to two major drawbacks found in this activation function [9]:

a) Sigmoid function squishes the input data into a small input region between 0 and 1. Therefore, the derivative shown in Eq. (3.3) and Eq. (3.4) has a very small gradient which towards zero gradient. The gradient is not able to update the weights parameter during the DNN training process. This vanishing gradient problem slows down the learning process of the DNN and degrades the DNN recognition performance. But, this issue can be reduced by using better initialization of weights parameter.

b) Sigmoid function is not zero-centred as shown in Fig. 3.2. This may cause the undesired zig-zag dynamics in the gradient updates for the weights parameter. This issue can be reduced by using a mini batch training scheme. Tanh shown in Fig. 3.2 scales sigmoid function to zero-centred. It converted a real-valued input, x to the range between -1 to 1. Tanh activation function, f_2 in Eq. (3.5) is the zero-centred version of the sigmoid activation function. The tanh function is defined as follows:

$$f_2(x) = \frac{\exp(2x)-1}{\exp(2x)+1} \tag{3.5}$$

The derivative of Eq. (3.5) for backpropagation training process is defined as follows:

$$f_2'(x) = [1 - f(x)][1 + f(x)] \tag{3.6}$$

3.2.2 Multistate activation function

MSAF [96] adding multiple state using a constant, r_1 and combined multiple logistic functions together as shown in Fig. 3.3. Two types of MSAF were introduced,

i.e. N-order MSAF and the symmetrical MSAF. This type of activation function allowed DNN to capture more important features on the input data v to improve classification rate. The 2-order MSAF demonstrated in Eq. (3.7) consists of state 0, state 1 and additional state 2. The symmetrical MSAF is a zero-centred version of MSAF which consists of three states i.e. states -1, state 0 and state 1 as illustrated Eq. (3.8).

$$f_3(x) = \frac{1}{1+\exp(-x-r_1)} + \frac{1}{1+\exp(-x)} \tag{3.7}$$

$$f_4(x) = -1 + \frac{1}{1+\exp(-x-r_1)} + \frac{1}{1+\exp(-x)} \tag{3.8}$$

The derivative of Eq. (3.7) and Eq. (3.8) for backpropagation training process is defined as follows:

$$f_3'(x) = f_4'(x) = \frac{e^{r_1-x}}{(e^{r_1-x}+1)^2} + \frac{e^{-x-r_1}}{(e^{-x-r_1}+1)^2} \tag{3.9}$$

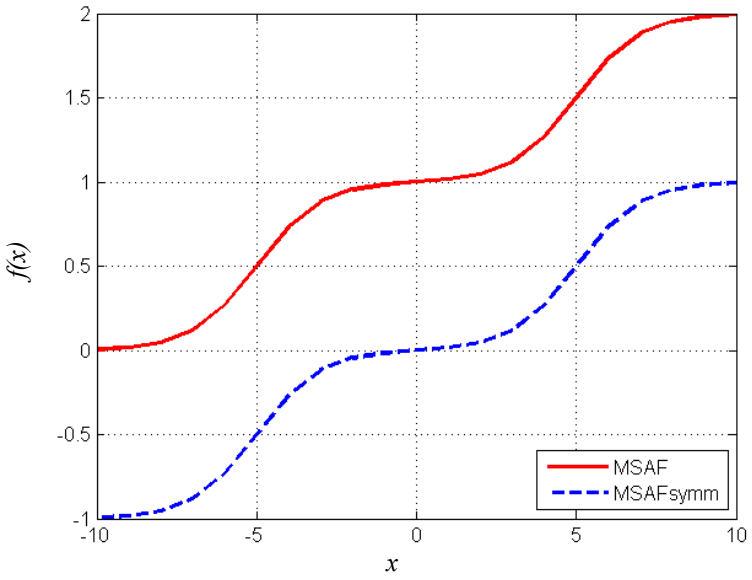


Figure 3.3: *Multistate activation function*

3.2.3 Rectifier linear unit

ReLU [98] as shown in Fig. 3.4 is widely used in DNN due to its simplicity in backpropagation that could reduce the computational cost in the training process. The ReLU equation is expressed in Eq. (3.10) with its derivations as shown in Eq. (3.11).

$$f_5(x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases} \tag{3.10}$$

$$f_5'(x) = \begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases} \tag{3.11}$$

However, the drawback of ReLU is that the value of activation function might be out of bound during the backpropagation training. Thus, the learning rate becomes the crucial variable from the training process. In addition, the activation node with smaller learning rate causes the data falls into zero region that could be dead node and cannot be re-activated in the training process. LReLU [99] is proposed to tackle the ReLU issues by adding a small constant value (c), for example $c = 0.1$ in the negative region of the ReLU activation function as shown in Eq. (3.12). Another version of LReLU known as PReLU [102] trains the LReLU constant variable adaptively throughout the training process.

$$f_6(x) = \begin{cases} x, & x > 0 \\ cx, & x \leq 0 \end{cases} \tag{3.12}$$

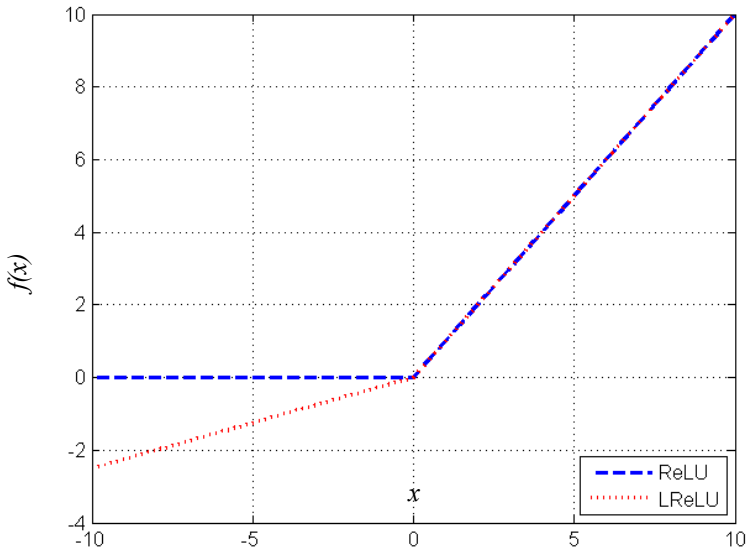


Figure 3.4: *Variation of Rectified linear unit*.

In PReLU, there is an additional trainable parameter © inserted in the negative segment of activation function. The derivative of Eq. (3.12) is defined as follows:

$$f_6'(x) = \begin{cases} x, & x > 0 \\ c, & x \leq 0 \end{cases} \tag{3.13}$$

The derivative of the trainable parameter c is defined as follows:

$$\frac{\partial f}{\partial c} = \begin{cases} 0, & x > 0 \\ x, & x \leq 0 \end{cases} \tag{3.14}$$

The derivatives shown in Eq. (3.13) and Eq. (3.14) will be used in backpropagation training process.

3.2.4 Exponential linear unit

Exponential linear units (ELU) [103] as shown in Fig. 3.5 is proposed to solve the bias shift issue during the training process. During the training process, the bias shift issue is defined as the change of a neuron's mean value. Bias shift issue always causes oscillations and impeded learning in the backpropagation training process. ELU has a similar characteristic as the batch normalization. ELU possesses negative values to push mean unit activations closer to 0 with lower computation complexity. Mean shift towards 0 shortens the network training process by carrying the gradient closer (decrease the gap) to the unit natural gradient. As a result, the bias shift effect can be reduced. ELU activation function has the advantage of noise robustness over the LReLU activation function. ELU has a clear saturation plateau in its negative regime, allowing them to learn more robust and stable representation in the deep neural network. Therefore, ELU network performs better than ReLU network that is trained using batch normalization. Eq. (3.15) demonstrates the activation function of ELU with a controlling parameter (α) as follows,

$$f_7(\alpha, x) = \begin{cases} \alpha(e^x - 1), & x < 0 \\ x, & x \geq 0 \end{cases} \tag{3.15}$$

The derivative of Eq. (3.15) can be defined as follows,

$$f_7'(\alpha, x) = \begin{cases} f(\alpha, x) + \alpha, & x < 0 \\ 1, & x \geq 0 \end{cases} \tag{3.16}$$

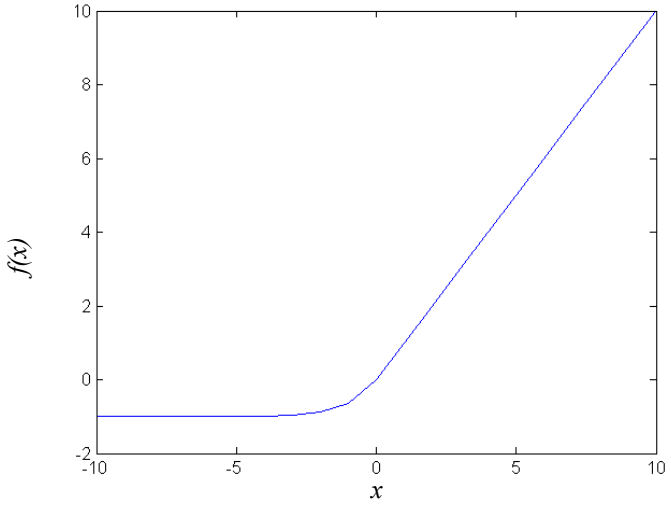


Figure 3.5: *Exponential linear unit (ELU).*

3.2.5 Activation function modification

Adaptive tangent activation function is introduced to solve the saturation problem occurred in tanh activation function as shown in Fig. 3.6. Two trainable variables slope, *s* and amplitude, *a* are added to the tanh activation function. The trainable variables are adjusted accordingly in the training process using gradient descent method. Equation (3.17) illustrated the adaptive tanh activation function.

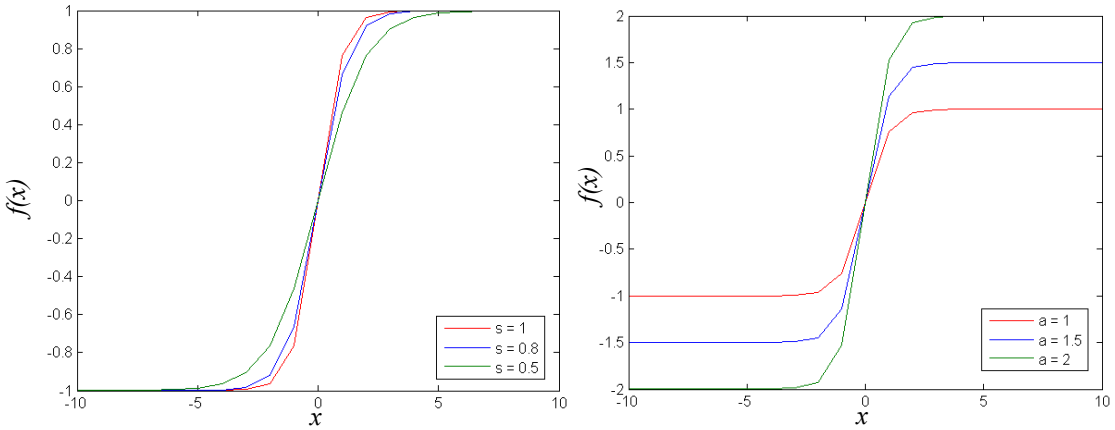


Figure 3.6: *Adaptive tanh activation function.*

$$f_8(x) = \text{atanh}(sx) \tag{3.17}$$

where *a* and *b* are the trainable parameter to rescale and bias. The derivative of the activation function is defined as follows:

$$\begin{aligned} f_8'(x) &= as\ sech^2(sx) \\ &= as\ (1 - tanh^2(sx)) \\ &= \frac{s}{a}[a - y_j(x)][a + y_j(x)] \end{aligned} \tag{3.18}$$

Derivative of trainable parameter *a* and *s,*

$$\frac{\partial f}{\partial a} = \tanh(sx) \tag{3.19}$$

$$\frac{\partial f}{\partial s} = \frac{x}{a}[a - y_j(x)][a + y_j(x)] \tag{3.20}$$

The derivative equations are demonstrated in the backpropagation algorithm (fine-tune) in appendix A.

3.3 Data randomization techniques

Randomness is a very important key point in training deep architecture neural network. Randomization of the data helps deep neural network (DNN) to learn well in

data generalization by shuffling the input samples. Therefore, DNN can adapt well in real life scenario i.e. DNN still can perform well regardless the change of background, orientation or position of the object. There are many types of randomization technique can be found in the literature. For example, adding random noise into the input data improves the classification rate of the DNN.

Dropout neural network [104] and DropConnect [105] are the regularization methods randomly removed visible and hidden units from the DNN during the training process. It is an effective way to reduce overfitting by preventing co-adaptions between the units. Stochastic backpropagation algorithm utilizes the randomness to provide "exploration energy" for finding better optimization solutions during gradient descent. Three types of dataset randomization were introduced to have the network generalization in the following subsections.

3.3.1 Randomization Technique #1

Randomization technique #1 as shown in Fig. 3.7, *K* of training images was divided into mini batches with the size of 100. Each mini batch was randomized with one fixed random sequence. The pseudocode for randomization technique #1 is illustrated as follows:

Pseudocode #1:

```
COMPUTE non-repeated random sequence;
READ total number of minibatch, R
READ width of minibatch
    FOR t = 1 to R
        CALL one minibatch
        REARRANGE according to the random sequence
        SAVE minibatch
    ENDFOR;
```

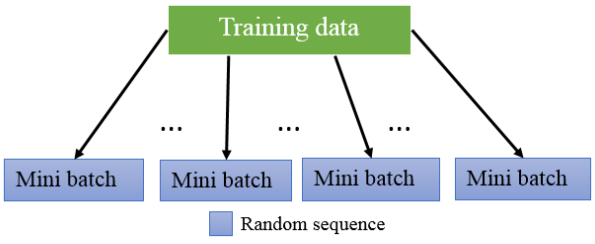


Figure 3.7: Randomization technique #1

3.3.2 Randomization Technique #2

Randomization technique #2 is illustrated in Fig. 3.8. *K* of training images was divided into mini batches with the size of 100. Each mini batch was randomized with *R* number of random sequences, where *R* is the number of mini batch. The pseudocode for randomization technique #2 is illustrated as follows:

Pseudocode #2:

READ total number of minibatch, *R*

READ width of minibatch

FOR tt = 1 to *R*

COMPUTE *R* number of non-repeated random sequence;

CALL one minibatch

REARRANGE according to the random sequence

SAVE minibatch

ENDFOR;

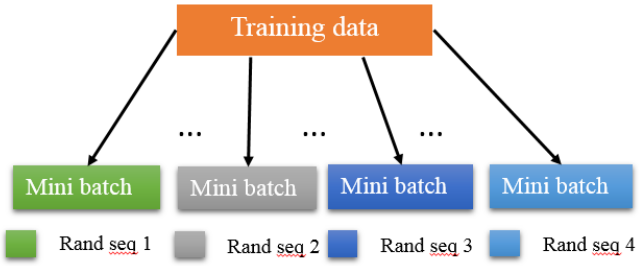


Figure 3.8: Randomization technique #2

3.3.3 Randomization Technique #3

Randomization technique #3 is illustrated in Fig. 3.9. A random sequence with the size of 100 is generated. *K* of training images were randomized first before divided into mini batches. Therefore, each mini batch did not have an equal number of samples for each class. The pseudocode for randomization technique #3 is illustrated as follows:

Pseudocode #3:

READ total number of input data, *R1*

COMPUTE *R1* number of non-repeated random sequence

REARRANGE input data according to the random sequence

CALCULATE total number of minibatch can be divided; minibatch width size = 100;

COMPUTE divide the input data into minibatches

SAVE minibatches

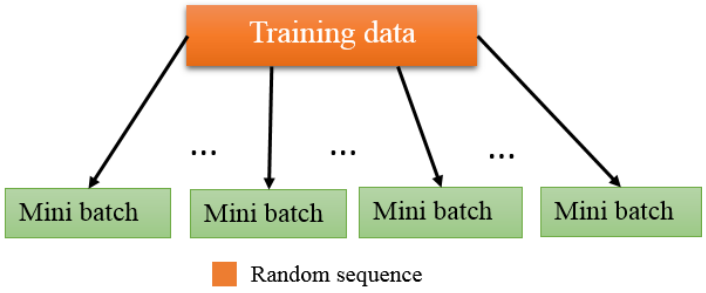


Figure 3.9: Randomization technique #3.

3.3.4 Experimental Results on Randomization Techniques

MNIST benchmark dataset was used to investigate the performance of these three types of randomization techniques. 44000 total number of training images were selected to train the deep belief network (DBN). 10000 test images were utilized to test the performance of the network after the training process was done. Fig. 3.10 showed the performance of DBN using different types of dataset randomization. Experiment results showed that randomization technique #3 outperformed the other

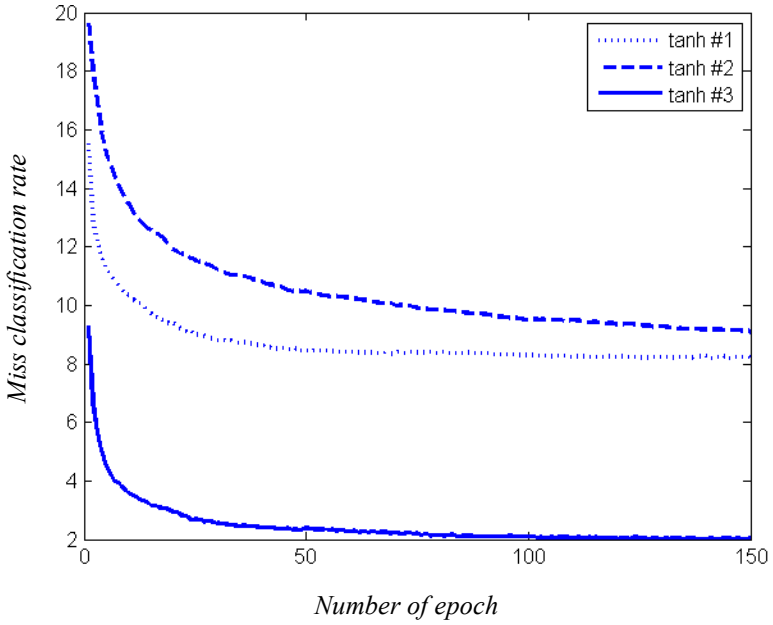


Figure 3.10: Misclassified rate for different types of dataset randomization with tanh activation function.

two types of randomization. Therefore, it showed that the number of sample images in a mini batch did not necessary to be equal in order to archive lower misclassification rate. Randomization technique #3 broke any predefined structure that may exist in the original dataset, thus, leads to a better generalization of the network during the training process.

3.4 Experimental setup

The experiments were conducted using MATLAB 2013b software and TitanX GPU. Two types of neural network were implemented for evaluate the performance of the activation functions. Deep belief network (DBN) was implemented for saturated types of activation function evaluation. Deep feedforward network was implemented for unsaturated and adaptive activation functions evaluation. DBN had an additional pre-training algorithm before the backpropagation algorithm training process. This pre-training process helped saturated type of activation functions to eliminate the saturation problem and the vanishing gradient problem. Five layers of the DBN was implemented for this evaluation. DBN was pre-trained 30 epoch using the restricted Boltzmann machine algorithm [95] as illustrated in Appendix A. Deep feedforward network (DFN) was implemented for the evaluation of the unsaturated type and adaptive type of activation functions. It consists of five layers fully connected feedforward network. Unlike DBN, the training process was only required the backpropagation algorithm. The main differences between the DBN and DFN was the additional pre-training algorithm. The parameters for both networks were showed in Table 3.1.

Table 3.1: *Parameters used for DBN and DFN*

No.	Items	Parameters for DBN	Parameters for DFN
1	No. of epoch for pre-train	30	-
2	No. of epoch for fine-tune	100	
3	No. of test repeated	3	
4	Type of weights	Gaussian	Xavier
5	No. of training images	60000	
6	No. of testing images	10000	
7	Regularization method	Weight decay	

The backpropagation algorithm was using a mini batch stochastic gradient descent backpropagation algorithm. The detail implementation of the pre-training and

backpropagation algorithm was illustrated in Appendix A. Small random Gaussian weight initialization with zero mean and 0.01 standard deviation [106] was used to initialize the weights parameter of saturated activation functions. For unsaturated activation function, the weight initialization is using Xavier weight initialization [94]. MNIST dataset contains 60000 training images and 10000 testing images were used to train and verify the performance of the network. The misclassification rate shown in Eq. (3.21) was calculated to evaluate the performance of each activation function. The sum of correct classified denoted the total number of outputs was matched with the targeted output.

$$Missclassification\ rate = \left(1 - \frac{sum\ of\ correct\ classified}{no.of\ input\ images}\right) \times 100\% \tag{3.21}$$

3.5 Investigation on Activation Functions

Saturated and unsaturated types of activation function were investigated in deep belief network (DBN) and four layered fully connected network using type #3 dataset randomization in the previous section as shown in Table 3.1 and Fig. 3.10. By comparing the saturated activation functions, i.e. sigmoid and tanh, MSAF and MSAFsymm, the experimental results showed that the performance was improved

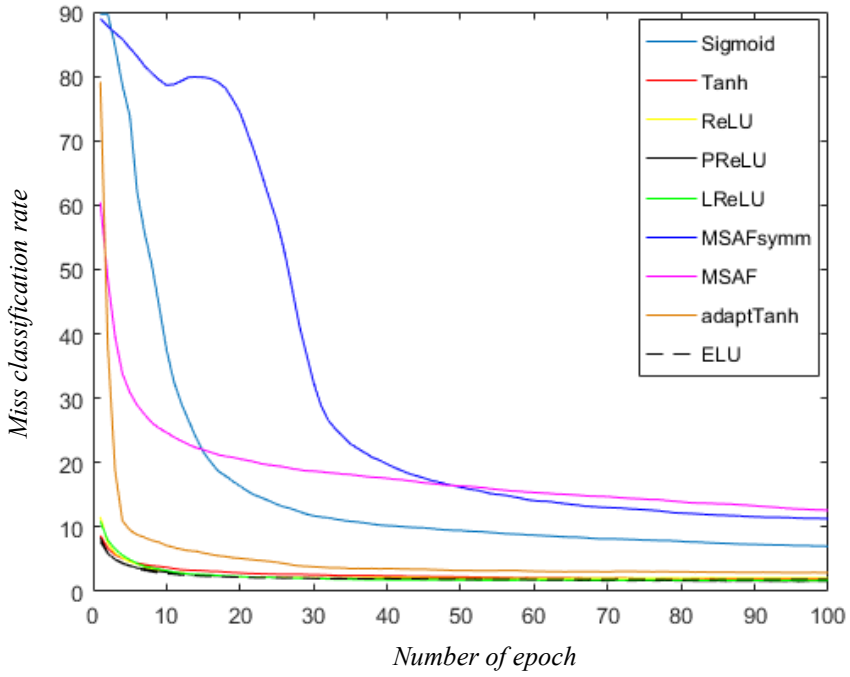


Figure 3.11: *Test misclassification rate for different types of activation function.*

greatly if the activation function is symmetrical on its origin. Tanh activation function reduced the misclassification rate to 1.86%. MSAF-symm reduced the MSAF misclassification rate to 11.28%. If compared to tanh activation function, MSAF did not show classification rate improvement after adding additional state in its activation function. The main drawback of the saturated activation function was required pre-training to eliminate vanishing gradient problem.

Therefore, we further investigated tanh activation function by adding two trainable parameters as shown in Eq. (3.13). By adding two trainable parameters to adjust the slope and amplitude of the tanh activation function, adaptive tanh network could train without pre-training process. Thus, simplified the tanh activation function training process. However, the misclassification rate of adaptive tanh was 1% higher than the tanh activation function.

Unsaturated activation functions i.e. ReLU, LReLU, PReLU, and ELU did not suffer vanishing gradient problem. Thus, unsaturated activation functions required less computational time since they did not do the pre-training process and did not has an exponential term in its function compared to the saturated activation function. ReLU activation function obtained the highest misclassification rate among the unsaturated type of activation function which was 2.08%. This was due to the error gradient of the neurons fallen into the negative region was zero. Thus, the neurons would not learn throughout the training process of the network.

Table 3.2: *Activation functions experimental results in MNIST dataset.*

Activation Functions	Train misclassification rate	Test misclassification rate	Pre-train
Sigmoid	7.1	7.01	Yes
Hyperbolic Tangent	0.17	1.86	Yes
MSAF	13.16	12.59	Yes
MSAF_symm	14.29	11.28	Yes
ReLU	19.86	2.08	No
LReLU	0	1.68	No
PReLU	**0.01**	**1.6**	**No**
ELU	0.03	1.88	No
Adaptive tanh	0.67	2.93	No

LReLU activation function reduced ReLU activation function's misclassification rate by allowing a small error gradient flowing through the negative

region of the activation function. PReLU activation function obtained the lowest misclassification rate among all types of activation function. PReLU activation function achieved 1.6% misclassification rate by adaptively learned the slope of the negative region in PReLU activation function during the training process of the network. PReLU activation function reduced LReLU activation function's misclassification rate to 1.68%.

3.6 Summary

In this chapter, three types of dataset randomization, different types of saturated, unsaturated, and adaptive activation functions were investigated in deep belief network (DBN) and fully connected network. Randomization is important for training a deep architecture neural network. Randomness forces DNN learn meaningful features from the raw data. Therefore, DNN can adapt well in real life scenario i.e. DNN still can perform well regardless the change of background, orientation or position of the object. We proposed #3 dataset randomization i.e. training images were randomized first before divided into mini- batches. Therefore, each mini batch did not have an equal number of samples for each class. #3 dataset randomization was then implemented into the activation functions analysis experiment. There are three types of activation function i.e. saturated activation function, unsaturated activation function and adaptive activation function. Saturated activation functions such as sigmoid, hyperbolic tangent multistate activation function required a pre-training process in order to eliminate the vanishing gradient problem during the training process of the network. The experimental results showed that the performance was improved greatly if the activation function is symmetrical on its origin. Tanh activation function could obtain better misclassification rate as compared to the non-symmetrical sigmoid activation function. MSAF-symm achieved better misclassification rate than MSAF's misclassification rate. As compared to tanh activation function, MSAF did not show classification rate improvement after adding additional state in its activation function. Unsaturated activation functions solved the vanishing gradient problem and provide sparsity to the network therefore, it would not overfit the training data. ReLU activation function obtained the highest misclassification rate i.e. 2.08% among the unsaturated type of activation function. This is due to the error gradient of the neurons fallen into the negative region is zero. As a result, it causes the vanishing gradient problem in the neurons to obtain an optimized weights parameter during the training

process. LReLU activation function reduced ReLU activation function's misclassification rate by allowing a small error gradient flowing through the negative region of the activation function. Adaptive activation function such as PReLU adjust the negative slope of the function according to the training process of the network. PReLU achieved the lowest misclassification rate i.e. 1.6% among all types of activation function. PReLU activation function further reduced the misclassification rate to 1.6%. Adaptive tanh is proposed to solve the tanh activation function saturation problem by adding two trainable parameters into the function. Adaptive tanh activation function can reduce the training iteration of the tanh activation function. However, the misclassification rate of adaptive tanh is 1% higher than the tanh activation function.

Chapter 4 Convolutional Deep Feedforward Network

4.1 Introduction

Deep feedforward network (DFN) is the fundamental architecture of all specialized deep architecture neural networks (DNNs), such as convolutional neural network, and recurrent neural network (RNN). DNN could perform better recognition capability than human eyes, which generally obtain 5% - 10% of misclassification rate. The use of DNN has shown significant performance in the applications of speech recognition [93], object recognition [92], pedestrian detection [90], handwritten digit recognition [107], and character text generation [108]. In the current trend of development, researchers have moved forward to apply deeper and wider neural network architecture to achieve a high recognition rate.

The development of deep structures neural networks has been exponentially growing after AlexNet [109] showed a drastic drop of top-5 error from 26.2% to 15.4% in the ImageNet large scale visual recognition challenges [110]. In the subsequent years, DNNs such as ZF net [111], VGG net [112], and GoogLeNet [113] was developed in deeper and wider neural networks to compete in ImageNet Challenge. Microsoft research team [114] developed the largest DNN structure containing 152 layers and hence it achieved 3.6% of misclassification rate in the challenge. However, the study of very deep network architecture always associates with the stacking of network layers and large computing resources.

The recent DNNs are configured based on the fundamental architecture of the convolutional neural network (CNN) [115]. CNN has layers of convolutional and pooling computing operation in order to obtain multiple feature maps across a full resolution image using local receptive fields. The number of feature maps is stacked in deeper layers to increase accuracy. To address the large computational issue, convolutional based DNN requires a supercomputer with multiple graphical processing units (GPUs) to preserve the good accuracy. Stacking of convolutional layers may attract huge attention in the research field, however, the effect of stacking up fully connected neural network (FCN) has not yet been discovered. FCN has all nodes connected between layers with the correlation of input pixel value while the CNN has a connection only to a local region in an input with parameters sharing.

Klambauer et al. [116] introduced self-normalizing to normalize the output of activation function with the convergence of zero mean and unit variance. This convergence property allows FCN to be trained in multiple layers without having vanishing and exploding gradient problems. Exploding gradient is a problem arises when training a deep network. The error gradient can accumulate to become a very large value causes the network unable to learn from the training data. Hinton et al. [95] proposed a layer-wise pre-training algorithm in deep belief network to solve the vanishing gradient problem. Other FCN architectures such as dropout network [104], maxout network [117], and dropConnect network [105] solved the overfitting problems of the FCN. Lin et al. [118] proposed network in network (NiN) combining convolutional and fully connected network as a micro network in NiN architecture. In the micro-network, it consists of convolutional layer (CL), pooling layer (PL) and the fully connected network (FL) which act as a potent function approximator. As a result, NiN is able to reduce the test error of CIFAR-10 and CIFAR-100 to 8.81% and 35.6% respectively. Lee et al. [119] combined a CL with DBN to scale down the large image and invariant to local translation. With the above-mentioned literature reviews, the effect of stacking FCN and CNN sequentially has not been fully studied instead of going deeper architecture.

In this chapter, an investigation of the effect to stack three types of network layers sequentially, i.e. (1) CL, (2) PL and, (3) FL. The layers are sequentially added in the network to understand the effect of networks layers in terms of classification accuracy. Subsequently, convolutional deep feedforward network (C-DFN) consisting of one CL, one PL and five FLs is proposed in the network. The CL is used as a trainable feature extractor and obtains features which are invariance to translations, rotation and scale before input to the four layers of FCN which acts as universal approximator for classification. The main contributions of this chapter are: (1) Investigation of the effect to stack three type network layers, i.e. CL in three filter size, PL and FL, (2) the proposed C-DFN consisting of one CL, one PL and four FLs for classification, (3) Four different types of architecture performance i.e. DFN, C-DFN, DBN, ad C-DBN were investigated on MNIST dataset, Daimler pedestrian dataset, and INRIA person dataset. This chapter is organised as follows: section 4.2 describes the architecture of C-DFN. Section 4.3 describes the experimental setup and experimental results discussion. Section 4.4 concludes this chapter.

4.2 Convolutional Deep Feedforward Network

C-DFN is made up of a CL, a PL, and followed by FLs as shown in Fig. 4.1. CL is used as a trainable feature extractor to extract unique features from the input image which are invariant to scale, rotation, and translation. In this section, the C-DFN architecture is described in detail and the architecture can be divided into two parts, i.e. forward propagation and backpropagation computation.

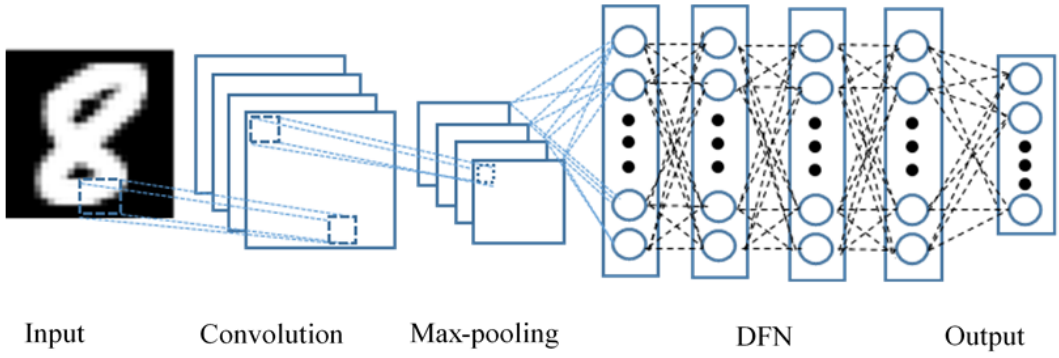


Figure 4.1: *Convolutional deep feedforward network.*

4.2.1 Forward Propagation

The input data is forward propagated from the input layer to multiple hidden layers until it reaches the output layer to produce an output signal. In the following section, the forward propagation computation is described in detail in each layer of the C-DFN.

4.2.1.1 Convolutional layer

In the configuration of CL, the input image, d_i is convoluted with $i \times j$ learnable kernels, k_{ij} and go through the activation function, $f(\cdot)$ to form the output feature map, v_j^l. The convolution operation can be defined as follows:

$$v_j^l = f\left(\sum_{i \in M_j} d_i^{l-1} * k_{ij}^l + b_j^l\right) \tag{4.1}$$

where M_j represents input maps. d_i^{l-1} represents i^{th} input channel. v_j^l denotes the output of the CL. Each output map is connected to an additive bias, b_j after the input image is convoluted with learnable kernels. This process can identify the salient features based on the local receptive field scanning.

4.2.1.2 Max-pooling layer

Once the input image is convoluted with kernels, a PL is used to build up spatial and configural invariance. It could reduce the half computational time of the feature maps computing process. Max-pooling performed in the non-overlapping neighbourhood using Eq. (4.2).

$$z_{ijk} = max_{n,m}(v_{i,j+n,k+m}) \tag{4.2}$$

Max-pooling obtains the maximum value in the neighbourhood of $n \times m$. The output of the subsampling layer is subsequently arranged into an input vector of deep feedforward network (DFN).

4.2.1.3 Deep feedforward network

Deep feedforward network is made up of multiple FL. The input signal is multiplied with the weight connection and then plus with a bias. A FL can be formed using Eq. (4.3).

$$x_j^l = \sum_{i=1}^{m} w_{ji} \times y_i^{l-1} + b_i \tag{4.3}$$

where m is the total number of inputs applied to neuron j. y_i^{l-1} is the input of FL from the output of the previous layer. The output signal y_j^l of the FL is given as follows:

$$y_j^l = f(x_j^l) \tag{4.4}$$

where $f(\cdot)$ is the activation function of the network. The output signal is then compared with the targeted output to produce an error backpropagated into the network layer by layer which is described in the next section.

4.2.2 Backpropagation

In the backpropagation, an error signal is produced by comparing the output of the network with the targeted output. The resulting error signal is propagated through the network in a backward direction, layer by layer. In this chapter, the stochastic gradient descent backpropagation algorithm [9] is used to update the weights connection in C-DFN. After the parameters of DFN is updated, the error signal is backpropagated to the PL and CL. For the PL, the local gradient from the DFN is

multiplied back to the PL. The local gradients of the CL for kernels update is defined as follows [120]:

$$\delta_j^l = f'(u_j) \cdot up(\delta_j^{l+1}) \tag{4.5}$$

where $f'(\cdot)$ is the derivatives of activation function, u_j is $\sum_{i \in M_j} d_i^{l-1} * k_{ij}^l + b_j^l$, $up(\cdot)$ as the pre-activation input, and $up(\cdot)$ represents up-sampling the local gradients from layer l+1, which is the local gradients from PL. The gradient of bias, b_j is the sum of all the entries of δ_j^l as shown in Eq. (4.6):

$$\frac{\partial E}{\partial b_j} = \sum_{q,r} (\delta_j^l)_{qr} \tag{4.6}$$

Finally, the gradients to update the kernels are computed using Eq. (4.7).

$$\frac{\partial E}{\partial k_{i,j}^l} = \sum_{q,r} (\delta_j^l)_{qr} (p_i^{l-1})_{qr} \tag{4.7}$$

where $(p_i^{l-1})_{qr}$ is the patch in v_i^{l-1} that was multiplied elementwise by $k_{i,j}^l$ during convolution in order to compute the element at (q, r) in the output convolution map v_j^l. After training is completed, the trained weights were tested in the test dataset.

4.3 Experimental Results and Discussion

Four types of network structure were investigated, i.e. DBN, C-DBN, DFN and C-DFN. The experiments involved the change of filter size in a CL, stacking a number of CL, stacking a number of FL, and stacking a number of convolutional and pooling layer (CP). All the network structures were developed using MATLAB 2013b software and TitanX GPU. The convolutional kernel was initialized with Gabor filter [121]. Four Gabor filters [122, 123] were used as the convolution kernels in the CL to extract the pertinent information from the raw image 28×28 pixels.

In the experiments, Leaky Rectified Linear Unit (LReLU) with a=0.25 was set in the activation function of the network. Table 4.1 shown the parameters used for DBN, C-DBN, DFN and C-DFN. Besides that, weight decay 1×10^{-4} was implemented during the training process. All the experiments were repeated 3 times and the average results were recorded in this section. The network structures investigation was performed on handwritten digit dataset (MNIST). The architecture performance

comparison between DFN, DBN, C-DFN, and C-DBN were conducted on MNIST dataset [115], INRIA person dataset[17], and Daimler classification dataset [18].

In MNIST dataset, 60k of training images were randomized using #3 data randomization in section 3.3.2 and divided into 600 mini-batches. Therefore every mini-batch had a non-equal distribution of samples in each class for the stochastic effect to increase generalization [123]. 10k of testing images was used to test the performance of the network. The number of epochs was set in 100 epochs for training. The learning rate, η was fixed to 0.1 and the momentum value was fixed to 0.9.

In INRIA person dataset, 5820 images were selected for training and 4516 images were selected for testing the network performance. The number of positive images and the number of negative images were equal in both testing and training dataset. The images were converted into greyscale and resized into 80×40. #3 data randomization was used to randomize dataset and rearranged into 582 mini-batches. The learning rate was fixed to 0.01 and momentum rate was fixed to 0.1.

In Daimler classification dataset, 9600 images were used for training and 9600 images were used for testing the network performance. The number of positive images and the number of negative images were equal in both datasets. The training dataset was randomized using #3 dataset randomization and divided into 960 mini-batches. The learning rate is fixed to 0.01 and momentum rate is fixed to 0.1. In the following section, each layer of the C-DFN had been investigated.

Table 4.1: *Parameters for DBN, DFN, C-DFN, and C-DBN*

No.	Items	Parameters
1	No. of epoch for pre-train	30
2	No. of epoch for fine-tune	100
3	No. of test repeated	3
4	Type of weights	Xavier
5	No. of training images	60000
6	No. of testing images	10000
7	Regularization method	Weight decay

4.3.1 Experiment on Convolutional Filter Size

Filter size experiments investigated the relationship between filter sizes and the performance of the network. CPFL network consists of one CL, one PL, and one FL. Table 4.1 showed the MNIST dataset experimental results. This experiment

investigated three different sizes of Gabor filter used in CL i.e. 3×3, 5×5, and 7×7. Based on the experimental result, smaller filter size achieved the lowest misclassification rate 11.86% in MNIST dataset. This is because smaller filter size was able to capture very fine details of the image while having a bigger filter size could leave out minute details in the image. The filter size effect could be observed clearly in Fig. 4.2.

Figure 4.2: Input for fully connected layer in different types of network structure (a) raw input (28×28) (b) Output of CP(3×3) (c) Output of CP(5×5) (d) 7×7 CP (e) Output of CPCP (3×3)

Table 4.2: *Misclassification rate of different Gabor filter sizes.*

Network Structure	Train Misclassification	Testing Misclassification
(3×3) CPFL	**11.94**	**11.86**
(5×5) CPFL	14.00	13.35
(7×7) CPFL	16.51	17.66

4.3.2 Experiment on Stacking Fully Connected Layers

As a result of the experiment in the section 4.2.1, the best filter size for convolutional layer was 3×3. Therefore, in this experiment, the filter size was fixed to 3×3. CP2FL was referred to one CL, one PL and two FLs. This experiment was stacked up to five FLs sequentially to investigate the performance of the network.

Table 4.3: *Misclassification rate of different number of stacked fully connected layer.*

Network Structure	Training Misclassification	Testing Misclassification
CPFL	11.94	11.86
CP2FL	1.15	4.29
CP3FL	0.63	2.83
CP4FL	**0.76**	**2.6**
CP5FL	0.7	2.86

Table 4.3 showed that stacked up to four FLs obtained the lowest misclassification rate which is 2.6% in MNIST dataset. Stacking five layers of FLs did not reduce the misclassification rate. This was due to the excessive number of neurons in a network that led to overfitting problem as well.

4.3.3 Experiment on Stacking Convolutional Layers

This experiment using CPFL as a baseline to investigate the network performance by stacking multiple layers of the CL. 2CPFL was referred to two CLs, one PL and one FL. Table 4.4 showed that adding a convolutional layer on CPFL reduced the misclassification rate from 11.86% to 10.91%. Adding one more convolutional layer on 2CPFL the misclassification rate increases to 14.41%. However, according to [112], stacking of two CLs with 3×3 filter size has the same effective receptive field of the 5×5 CL. Therefore, in this experiment the training and testing performance of 2CPFL had achieved a better recognition rate than CPFL with 5×5 filter size. Similarly, 3CPFL had performed better than CPFL with 7×7 filter size.

Table 4.4: *Misclassification rate of different number of stacked convolutional layer.*

Network Structure	Training Misclassification	Testing Misclassification
CPFL	11.94	11.86
2CPFL	**11.7**	**10.91**
3CPFL	15.55	14.41

4.3.4 Experiment on Stacking Convolutional-Subsampling Layers

Similar to experiments in section 4.3.3, this experiment applied CPFL as a baseline to investigate the effect of adding PL after every CL of the network structure. Experimental results in Table 4.5 showed that adding more PL contributed to worse misclassification rate. The effect of PL was to reduce the dimensionality of feature maps while remaining the rotational and shift invariant features of the local region. Therefore, adding more PLs introduced the information loss for the classification task.

Table 4.5: *Misclassification rate of different number of stacked CP layer.*

Network Structure	Training Misclassification	Testing Misclassification
CPFL	**11.94**	**11.86**
CPCPFL	15.59	14.68
CPCP2FL	16.3	14.56
CPCPCPFL	68.81	69.97

4.3.5 Comparison of DFN, DBN, C-DFN and C-DBN

As a result, CP4FL network structure consisting of one CL, one PL and four FLs obtained the lowest misclassification rate 2.6% in MNIST dataset. In this network, the relationship of inter-connected pixels is fully linked with weights parameter although the number of convolutional kernels is small. In addition, a stack of equal length of the FL has relatively improved the performance of the recognition rate when the network is going deeper.

In this experiment, CP4FL is renamed as C-DFN. This section compared DFN, C-DFN, DBN, and C-DBN architecture performance on MNIST dataset, Daimler pedestrian dataset, and INRIA person dataset. In addition, PReLU activation function is added to the networks for further investigation. The network structures were showed in Table 4.6. In Table 4.7, C-DFN with LReLU activation function achieved the lowest misclassification rate among all the network architectures.

Table 4.6: *Network structure of DFN, DBN, C-DFN, and C-DBN on MNIST dataset.*

DFN and DBN	C-DFN and C-DBN
Layer 1 – 784 Layer 2 – 784 Layer 3 – 784 Layer 4 – 784 Layer 5 - 10	Layer 1 – 28×28 Layer 2 – 26×26×4 Layer 3 – 13×13×4 Layer 4 – 676 Layer 5 – 676 Layer 6 – 676 Layer 7 – 676 Layer 8 – 10

Table 4.7: *Four types of network structures misclassification rate in three types of dataset.*

Dataset	Misclassified rate (%)							
	LReLU				PReLU			
	DFN	C-DFN	DBN	C-DBN	DFN	C-DFN	DBN	C-DBN
MNIST	1.73	**1.32**	1.74	1.48	1.74	1.59	1.60	**1.47**
INRIA	16.45	**13.11**	17.27	13.42	15.72	12.69	15.90	**12.60**
Daimler	13.85	**13.81**	15.42	15.47	13.49	**13.39**	15.16	15.31
Average	10.68	9.41	11.48	10.12	10.32	9.22	11.04	9.79

C-DFN with LReLU reduced DFN's misclassification rate to the average of 9.41%. C-DFN with PReLU minimized the average of 9.22% misclassification rate.

C-DBN with LReLU reduced misclassification rate to the average of 10.12%. C-DBN with PReLU reduced the average of 9.79% DBN's misclassification rate. Therefore, the CL acts as a trainable feature extractor improved the network performance significantly. Moreover, it reduced 14% of the trainable parameters in DFN. Based on the experimental results obtained from three different benchmark datasets, the trainable feature extractor is therefore applicable to different types of situations and applications.

4.4 Summary

DFN is the fundamental network architecture in the recent development of deep learning neural networks for image classification. Going deeper and wider network architecture has attracted large research attention due to the improvement of classification accuracy. However, the study of very deep network architecture always associates with the stacking of network layers and involvement of large computing resources. Three basic types of neural layers are frequently stacked, i.e. CL, PL and FL. In this chapter, the stacking of these basic layers are sequentially added for performance investigation. The effect of stacking CL, PL and FL contributed to the recognition effect differently in terms of deeper layers. In CL, the smallest size of the receptive field captures better local region of an input. However, if the size of the convolutional layer has not grown wider, the recognition rate may become poor due to the loss of connecting information. The requirement of increasing the number of filters across one layer is important to improve the connectivity of local information. PL has the effect of subsampling for reducing dimensionality and introducing some degree of invariance to local translation and distortion in the input. In a FL, the relationship of inter-connected pixels are linked with weights parameter. Therefore, the stack of equal-length of the FL has relatively improved the performance of the recognition rate when the network is grown deeper. In addition, experimental results showed that a CL with four FLs (C-DFN) performed significantly better than DFN, DBN, and C-DBN using three datasets. C-DFN with LReLU reduced misclassification rate to the average of 9.41%. C-DFN with PReLU minimized the average of 9.22% misclassification rate. C-DBN with LReLU reduced misclassification rate to the average of 10.12%. C-DBN with PReLU reduced the average of 9.79% misclassification rate. Therefore, the CL acts as a trainable feature extractor improved the network performance significantly. Moreover, it reduced 14% of the trainable parameters in DFN. Based on the

experimental results obtained from three different benchmark datasets, the trainable feature extractor is therefore applicable to different types of situations and applications. In the next chapter, the proposed C-DFN is implemented for VRUs detection.

Chapter 5 Vulnerable Road Users Detection

5.1 Introduction

Vulnerable road users (VRUs) detection is a challenging and important task in autonomous vehicles. A VRUs detection system acts as an extra set of eyes for drivers, helping them avoid potentially catastrophic collisions. Fig. 5.1(a) showed the traditional VRUs detection algorithm and Fig. 5.1(b) demonstrated the proposed deep learning VRUs detection algorithm. The traditional VRUs detection had two main steps: (1) Feature detection phase and (2) Training classification phase. In the first phase, hand-engineered feature detector such as Histogram of gradient (HoG) [17, 29, 124-127], Haar-like features [39, 124, 125], and Gabor features [121] were designed to capture the shape of VRUs. Colour histograms [126], and colour-self-similarity (CSS) [127] were used to detect colour features. Local binary pattern (LBP) [128] detected texture, depth, motion, and their combinations were the examples of features used as input for image classification. Subsequently, the features detected in the first phase was used as input to train machine learning classifier in the second phase. Machine learning classifier referred to shallow architecture neural networks such as multilayer perceptron [9], support vector machine (SVM) [129] and Adaboost [130]. In the testing phase, the images in testing dataset were used as the input for the trained network for classification.

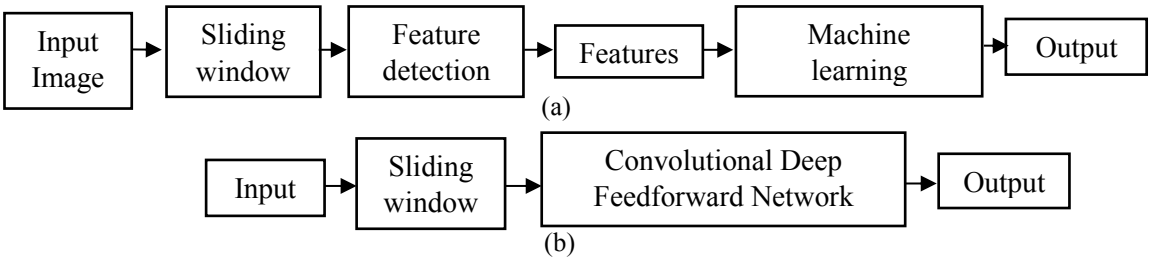


Figure 5.1: *Differences between (a) traditional VRUs detection algorithm and (b) proposed C-DFN VRUs detection algorithm.*

Traditional VRUs detection algorithm such as HoG+CSS+linear SVM [61, 131] Aggregated Channel Features (ACF) and Adaboost classifier [132-134] were utilized for pruning the window to obtain the region of interest. Tome et al. [88] investigated different types of region proposal methods and utilized ACF for region proposal for pedestrian detection. In recent years, deep architecture neural networks

(DNN) were potentially more capable than shallow architecture neural networks to detect and classify VRUs using camera imagery. Compared with handcrafted features, the automatic features generated by deep neural network (DNN) such as convolutional neural network (CNN) [134] and deep belief network (DBN) [95] were more capable of expressing the details of internal properties of the data.

The proposed deep learning VRUs detection algorithm shortened the process of pedestrian detection by utilizing the trainable feature mechanism in a deep learning algorithm to detect the important features in the input image. Therefore, the main difference between traditional pedestrian detection algorithm and proposed deep learning pedestrian detection algorithm was that the latter approach has the trainable feature mechanism where the feature detector was done automatically by deep learning pedestrian detection algorithm. The proposed VRUs detection algorithm focused on the medium range of pedestrian which was within 30 to 80 pixels tall as the critical range for vehicle's VRUs detection. When the system detects pedestrian on the vehicle path, a stopping distance is needed for a driver to react. The stopping distance depended on two factors [135]: (a) thinking distance: driver took time to react to a situation. During this reaction time, the car carried on moving. The thinking distance was the distance travelled in between the driver had realised to perform braking until he had performed the action braking. (b) Braking distance: The braking distance was the distance for a vehicle to stop completely after the brakes was applied. There were a few factors that might increase the stopping distance. Fig. 5.2 showed the stopping distances for a vehicle at different speeds (km/h).

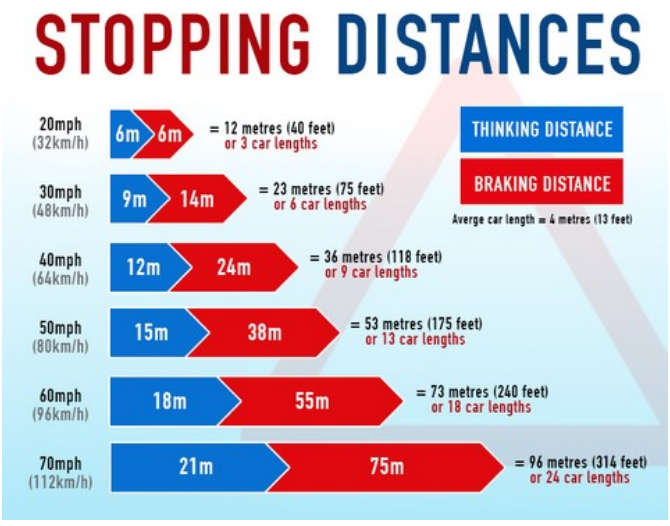


Figure 5.2: *Stopping distance for vehicle in different speeds (km/h). [135]*

Moreover, the thinking distance could be increased by tiredness, distraction, alcohol and drugs. Braking distance could be increased by greater speed, poor road conditions, car condition i.e. bald tyres, poor brakes or full of people in a vehicle. Therefore, the proposed VRUs detection system could assist driver to shorten the stopping distance of the vehicle. This chapter is organised as follows: section 5.1 described benchmark dataset pre-processing and the pedestrian detection procedure. Section 5.2 described the experimental setup and experimental results discussion. Section 5.3 concluded this chapter.

5.2 Dataset pre-processing and final model implementation

This section illustrated the procedure to implement the convolutional deep feedforward network (C-DFN) proposed model into VRUs detection model. Section 5.1.1 illustrated the dataset pre-processing procedure before used it for training and testing final model performance. Section 5.1.2 explained the training phase implementation and training procedure. Section 5.1.3 described the testing phase implementation and testing procedure.

5.2.1 Dataset Pre-processing

Before using the video-based Caltech pedestrian dataset [10] to evaluate the final model, pre-processing and image extraction from the video is needed. In this chapter, scenario cal1 was chosen to evaluate the proposed model i.e. S0 – S5 used as the training dataset, S6 – S10 used as the testing dataset. The pre-processing tasks included:

(1) **Extract seq format video file and its annotation file.** Seq file could be extracted to a directory of images using Piotr's Matlab Toolbox. The video files were extracted every 30^{th} frame of each video's images and its annotations based on the evaluation methodology set by Caltech pedestrian dataset.

(2) **Filtering ground truth bounding box.** Ground truth bounding box (BB) format was defined as [*x, y, width, height, ignored*]. *x*, *y* defined as top left corner of the BB coordinate. *ignored* was a binary number for ground truth filtering. The ignored bounding box, bb_{ig} was used to exclude some portion of the dataset that was not considered in this detection model. The following BB were set as ignored bounding box bb_{ig}:

a) Any BB under 30 pixels height
b) Any BB above 80 pixels height
c) Truncated by image boundaries
d) Ambiguous case: BB labelled as 'Person?'
e) BB labelled as 'People'

(3) **Extract medium range (30-80 pixels high) positive image (pedestrian) and negative image (background) for training phase data.** The positive images were cropped from Caltech training dataset (set 0 – set 5) using the annotation of the dataset. Different sizes of the positive images were resized to 80×80×3 and saved according to the video folder. The negative images (80×80×3) were cropped randomly from the images without any annotation BB. The cropped negative images were then added into the training dataset and randomized the sequence of the images.

(4) **Randomize ground truth bounding box and labels.** Ground truth BBs were randomized based on the sequence of the images and the corresponding classification labels were set for training phase.

5.2.2 Training Phase

Fig. 5.3 showed the training samples were resized to 80×80×3 as the training inputs of the C-DFN. Fig. 5.4 showed the proposed model implemented into VRUs training dataset. The training inputs were convoluted sequentially with four 3×3×3 Gabor filters. The output of the convolutional layer (CL) was then max pooled with a 2×2 neighbourhood. The output of the pooling layer (PL) was then used as the first layer of the DFN for the detection. The five output of the C-DFN consists of the detected BB and the score of classification results which as [*X, Y, width, height, score*].

(a)

(b)

Figure 5.3: *Training samples (a) negative images (b) positive images.*

The output of the C-DFN was then compared with the targeted output and the error was backpropagated to update the weights parameter of the networks. Stochastic gradient descent was used as the backpropagation algorithm. After that, the misclassification rate and mean square error was calculated to justify the training progress of the network.

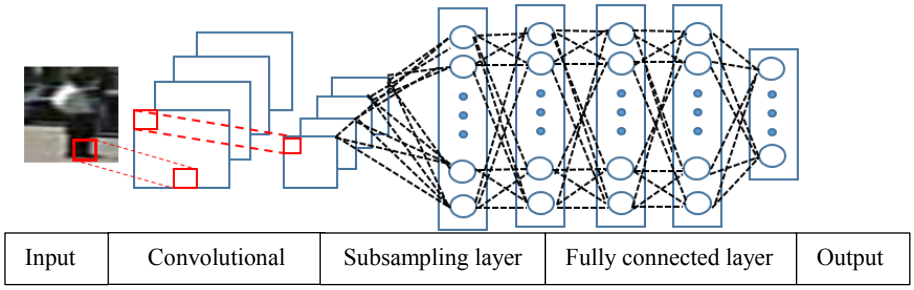


Figure 5.4: *C-DFN training phase.*

5.2.3 Testing Phase

After the network had been trained, the network was tested on the testing dataset s6-s10. In the testing phase, the detection system consists of two stages: region of interest extraction (ROI) and pedestrian detection. In the ROI extraction phase, a sliding window (80×80×3) with a stride size of five was used as a search window to detect pedestrian on the road exhaustively. The searching region started at one-third of the original image. As shown in Fig. 5.5, the search region was only focused on the yellow box region (320×640×3) to avoid searching pedestrian on the sky and buildings. The sliding window cropped 80×80×3 on the input image and used as the input for C-DFN. Table 5.1 showed the parameters used for C-DFN VRUs detection system.

Table 5.1: Parameters for C-DFN VRUs detection system

No.	Items	Parameters
1	No. of epoch for training	100
2	No. of test repeated	3
3	Type of weights	Xavier
4	No. of training set	s0-s5
5	No. of testing set	s6-s10
6	Regularization method	Weight decay

In short, the VRUs detection system took a single frame of video as input and return BB and score for each detection. The tracking system was not included in this final model as it is out of the scope of this thesis. In order to evaluate detectors

performance, miss rate, false positives per image (FPPI) and accuracy of the detection were calculated for each image.

5.3 Experimental results and Discussion

C-DFN was implemented for VRUs detection task and evaluated on the Caltech pedestrian benchmark dataset. C-DFN is a single pipeline model without using image pre-processing and feature extraction method to detect VRUs. C-DFN utilized deep architecture neural network trainable feature mechanism to shorten the traditional detection process. In addition, C-DFN simplified the training process and can be modified according to the application. This section showed the experimental detection performance and discussion of the holistic C-DFN, holistic stacked C-DFN and part-based C-DFN detection and part-based detection to investigate the detection performance of C-DFN.

5.3.1 Experiment on C-DFN Holistic VRUs Detection

Fig. 5.5 showed the holistic C-DFN VRUs detection system. In the VRUs detection process, the raw images captured by a camera attached on a vehicle was used as the input of C-DFN. The output of the C-DFN was a classification score 0 or 1 and the detection BB. Fig. 5.6 showed the experimental detection images extracted from the Caltech pedestrian testing dataset. Green BB indicated the ground truth bounding box, while the red BB indicated the detection BB. The number 0 or 1 on the top left corner of the green BB indicated that BB was ground truth BB or ignore BB respectively. The number showed on the top left corner of the red BB indicated the area of overlap calculated using Eq. (2.1).

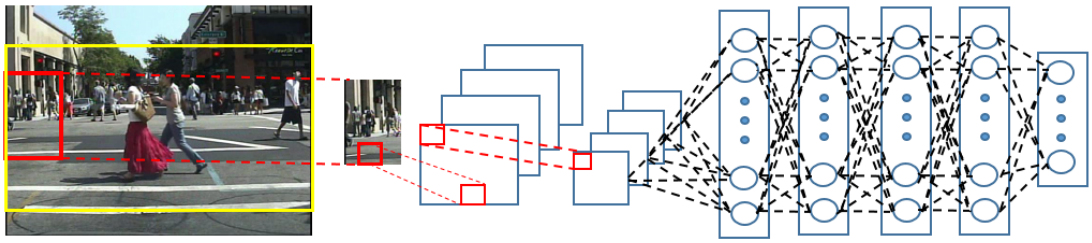


Figure 5.5: *C-DFN for holistic VRUs detection.*

Test image (a) shown in Fig. 5.6 showed the holistic C-DFN detected one out of three pedestrians in the image. This is due to the undetected pedestrian is covered up with the shadow of the building and the other pedestrian. Therefore, the detection model is hard to identify the presence of a pedestrian in that region. Test image (b)

shown in Fig. 5.6 detected the pedestrian correctly when the pedestrian moving out of the shadow region. Test image (c) shown in Fig. 5.6 showed the miss detection case of the holistic C-DFN under a complex lighting and background environment. Test image (d) shown in Fig. 5.6 showed the holistic C-DFN successfully detected a medium scale pedestrian (30-80 pixels high) on the walkway. In the next section, holistic C-DFN was modified into stacked C-DFN and the experimental result was discussed.

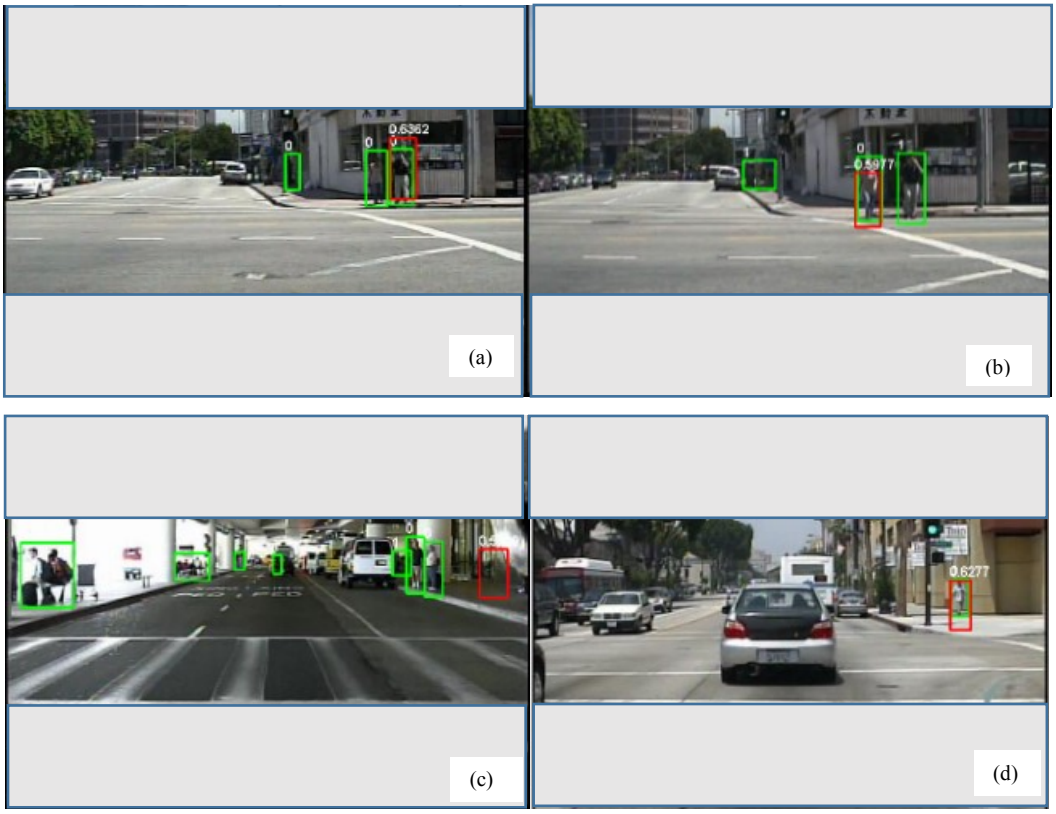


Figure 5.6: *C-DFN holistic VRUs detection results*.

5.3.2 Experiment on Stacked C-DFN Holistic VRUs Detection

Fig. 5.7 showed the architecture of two stacked C-DFN for holistic VRUs detection system. The first C-DFN act as a ROI detector. The output of the ROI as shown in Fig. 5.8 was used as the input of the second C-DFN. The output of the second C-DFN was the classification score and the detection BB. Fig. 5.9 showed the two stacked C-DFN experimental detection results. Test image (a) in Fig. 5.9 showed stacked two C-DFN detected the pedestrian covered by shadow. However, it failed to detect the other two pedestrians. Test image (b) in Fig. 5.9 showed stacked two C-DFN was able to detect a medium scale pedestrian in the video image. Test image (c) in Fig. 5.9 showed stacked two C-DFN detected a partially occluded pedestrian. Test image

(d) in Fig. 5.9 showed the stacked two-C-DFN can detect the medium scale pedestrian correctly even under a complex background on the scene.

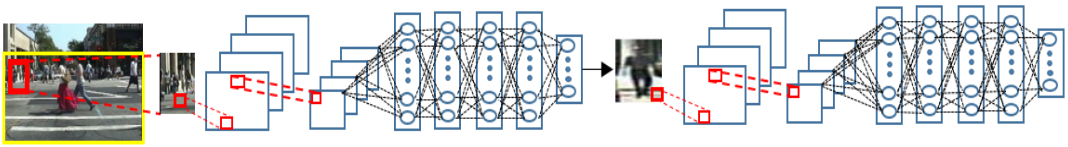


Figure 5.7: *Stacked two C-DFN for holistic pedestrian detection.*

Figure 5.8: *Region of interest.*

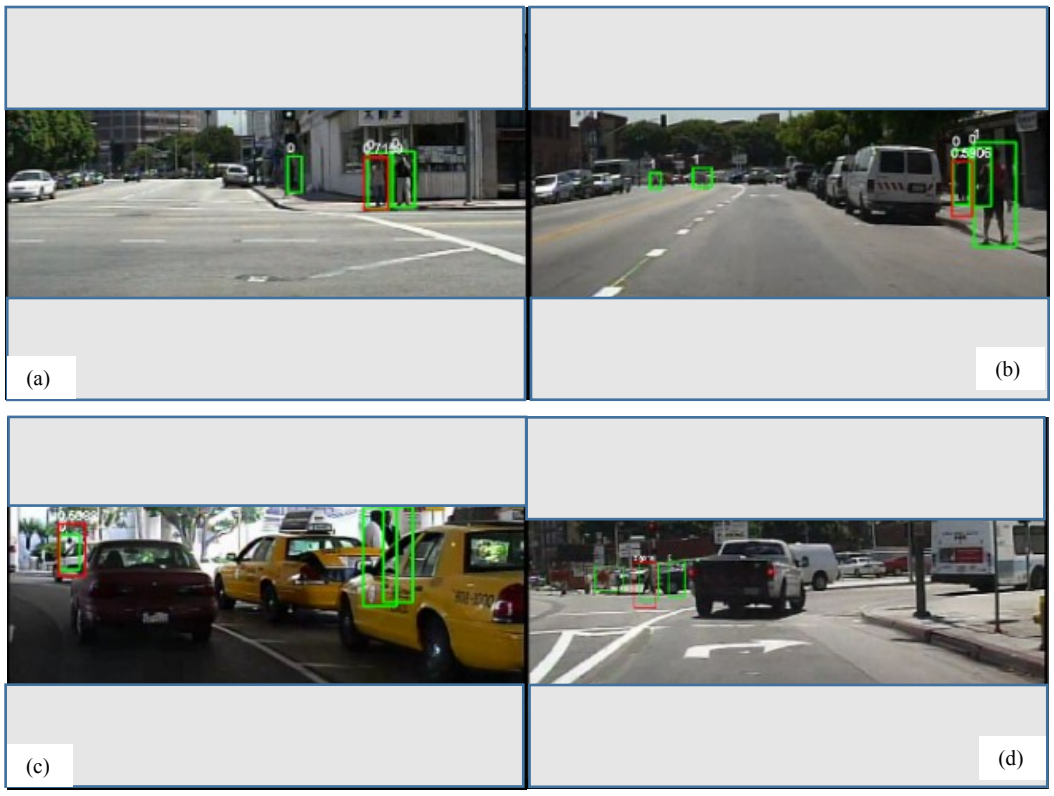


Figure 5.9: *Stacked two C-DFN VRUs detection results.*

Next, the performance of stacking three C-DFN for holistic pedestrian detection was further investigated. The experimental detection results as shown in Fig. 5.10. Test image (a) in Fig. 5.10 showed the detection result of three C-DFN obtained similar detection result as the holistic C-DFN. Test image (b) in Fig. 5.10 showed three C-DFN detected one pedestrian and miss detected the other one due to the lighting condition. Test image (c) in Fig. 5.10 showed three C-DFN detected one of the

pedestrians walking on the walkway but miss detected the pedestrian on the road beside the vehicle. The next frame of the test image (c) showed in the test image (d). Test image (d) in Fig. 5.10 showed that the miss detected a pedestrian in the test image (c) is detected in the test image (d). Another similar detection results are shown in the test image (e) and test image (f) in Fig. 5.10. The undetected pedestrian is detected in the next video image. Part-based stacked C-DFN is implemented and investigated in the next section.

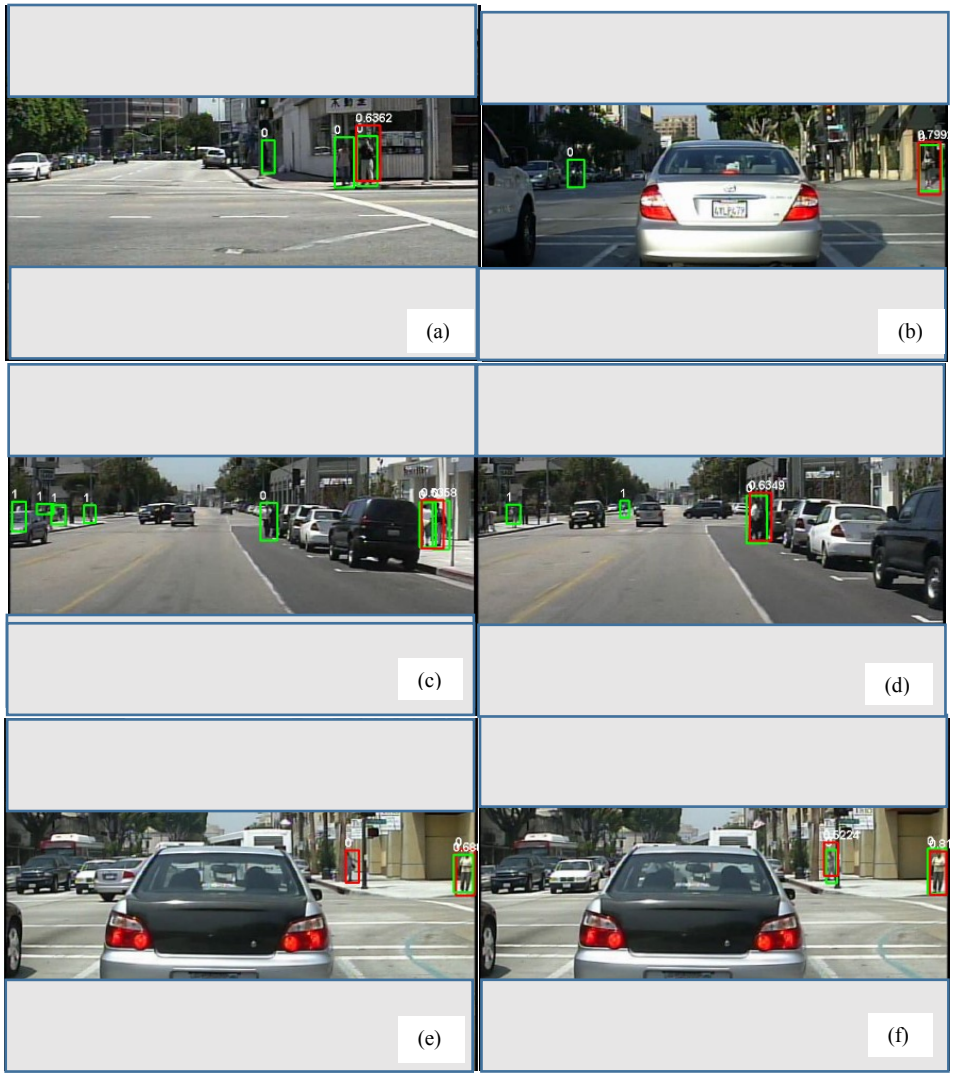


Figure 5.10: *Stacked three C-DFN detection results.*

5.3.3 Experiment on Stacked C-DFN Part-based VRUs Detection

Part-based stacked C-DFN was developed to solve the pedestrian occlusion problem and improve the detection performance. Fig. 5.11 showed the part-based approach of stacked C-DFN. The stacked C-DFN was trained as a holistic pedestrian detection which is the same approach in section 5.3.2. The first C-DFN output

bounding box used as a ROI for the following detection model. For the medium scale part-based VRUs detection, the VRUs is divided into two parts i.e. upper body and lower body [56]. Therefore, the training image for part-based VRUs detection was divided into the upper body part and lower body part to train the part-based C-DFN respectively. The upper body and lower body division is based on the body ratio stated in [56]. Upper body consists of head and neck (16%), and torso (34%) while the lower body consists of legs (50%). The ROIs obtained from the stacked C-DFN is then divided into two as the input for part-based C-DFN. The output of the part-based C-DFN was the classification scores of the ROIs. The classification scores are then matched with the output BBs.

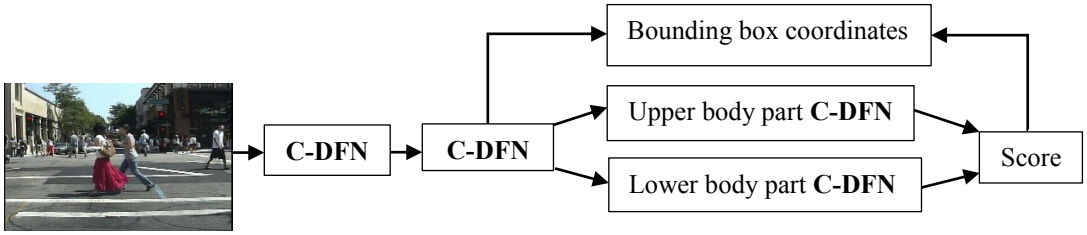


Figure 5.11: Stacked *C-DFN part-based detection.*

Test image (a) in Fig. 5.12 showed that three pedestrians in the image were detected successfully. As compared to the holistic VRUs detection result, the part-based VRUs detection model had shown a significant improvement. Test image (b) in Fig. 5.12 showed the output of part-based C-DFN. The detection BB matched correctly with the ground truth BB. Test image (c) in Fig. 5.12 detected all the pedestrians correctly. The green BB label as 1 was set as ignored BB. Therefore, it is not considered in the detection evaluation. Test image (d) in Fig. 5.12 detected 50% of the pedestrians in the scene. Test image (e) in Fig. 5.12 showed all medium scale VRUs (30-80 pixels high) were detected correctly. Test image (f) in Fig. 5.12 showed a partially occluded pedestrian was detected.

The overall detection results comparison between holistic VRUs and part-based VRUs detection model were summarized in Table 5.2. Table 5.2 showed the average miss rate per image and false positive per image (FPPI) for testing set 6 to set 10 of the Caltech pedestrian datasets. For holistic VRUs detection results shown in Table 5.3 and Table 5.4, 3 stacked C-DFN obtained the best performance which reduced 10.09% of the 2 stacked C-DFN's miss rate and reduced 14.47% of the C-

Figure 5.12: *Stacked C-DFN part-based VRUs detection results.*

Table 5.2: VRUs *detection results comparison.*

Categories	C-DFN	2 Stacked C-DFN	3 Stacked C-DFN	Part-based
Miss Rate (%)	57.89	53.51	43.42	**42.47**
FPPI	18.21	5.68	5.28	**5.45**

DFN's miss rate. C-DFN obtained the best performance which reduced 10.09% of the 2 stacked C-DFN's miss rate and reduced 14.47% of the C-DFN's miss rate. The FPPI of the 3 stacked C-DFN reduced 0.4% of the 2 stacked C-DFN's FPPI and reduced 12.96% of the C-DFN's FPPI. However, part-based model archived the lowest miss

detection rate and FPPI among all the detection models. Part-based model further reduced 0.95% of the 3 stacked C-DFN's miss rate and similar FPPI as shown in Table 5.5. Thus, it proves that the body parts verification helps the model to detect VRUs better.

Table 5.3: *Holistic VRUs detection results comparison between C-DFN and 3 Stacked C-DFN.*

Categories	C-DFN	2 Stacked C-DFN	3 Stacked C-DFN	Average
Miss Rate (%)	57.89	53.51	43.42	14.47
FPPI	18.21	5.68	5.28	12.93

Table 5.4: *Holistic VRUs detection results comparison between 2 Stacked C-DFN and 3 Stacked C-DFN.*

Categories	C-DFN	2 Stacked C-DFN	3 Stacked C-DFN	Average
Miss Rate (%)	57.89	53.51	43.42	10.09
FPPI	18.21	5.68	5.28	0.40

Table 5.5: *Holistic VRUs detection results comparison between 2 Stacked C-DFN and 3 Stacked C-DFN.*

Categories	C-DFN	2 Stacked C-DFN	3 Stacked C-DFN	Part-based	Average
Miss Rate (%)	57.89	53.51	43.42	**42.47**	0.95
FPPI	18.21	5.68	5.28	**5.45**	0.17

5.4 Summary

In this chapter, the proposed model C-DFN was implemented into the VRUs detection system. C-DFN is a single pipeline model without using image pre-processing and feature extraction method to detect VRUs. C-DFN utilize a deep architecture neural network trainable feature mechanism to shorten the traditional detection process. In addition, C-DFN simplified the training process and it is able to modify according to the application. First, the benchmark Caltech dataset was pre-

processed before using it for training and testing the final model. In the training phase, the training images were extracted from the dataset videos. Positive images were extracted using the given ground truth BBs. Negative images were cropped randomly from the background of the road scene. The positive and negative images were then used to train C-DFN. In the testing phase, the uncropped test images were used as the input images for the trained C-DFN. In addition, the C-DFN was further modified into C-DFN holistic VRUs detection, stacked C-DFN holistic VRUs detection and part-based VRUs detection model. For holistic VRUs detection results, 3 stacked C-DFN obtained the best performance which reduced 10.09% of the 2 stacked C-DFN's miss rate and reduced 14.47% of the C-DFN's miss rate. The FPPI of the 3 stacked C-DFN reduced 0.4% of the 2 stacked C-DFN's FPPI and reduced 12.93% of the C-DFN's FPPI. However, part-based model archived the lowest miss detection rate and less FPPI among all the detection models. Part-based model reduced 0.95% of the 3 stacked C-DFN's miss rate. The body parts verification helps the model to detect VRUs better. Therefore, the part-based model can detect the medium scale VRUs even if the images are at low-resolution.

Chapter 6 Conclusions

Vulnerable road users (VRUs) detection system plays an important role in reducing the number of fatalities on the road. There are many challenges in trying to detect VRUs on the road using a camera. The challenges are mainly due to the variability of VRUs appearance, weather conditions and variability of the background. This chapter summarizes the contribution of this research and its future potential.

6.1 Summary and contributions

In the third chapter of this thesis, a simple deep feedforward network (DFN) was developed as a baseline. DFN was then used to investigate the relationship between the activation functions and network performance. Saturation activation functions, unsaturated activation functions and adaptive activation functions were investigated in four layered DFN. The saturated activation function namely the hyperbolic tangent (tanh), a pre-processing algorithm known as the greedy layer-wise pretraining algorithm was developed to eliminate the vanishing gradient problem during the training process of the network. Adaptive activation function i.e. parametric rectified linear unit (PReLU) achieved the lowest misclassification rate i.e. 1.6% among all types of the activation function. PReLU activation function reduced 0.08% of leaky rectified linear unit (LReLU) activation function's misclassification rate. Adaptive tanh was proposed to solve the tanh activation function saturation problem by adding two trainable parameters into the function. Adaptive tanh activation function shortens the tanh activation function training process. However, the misclassification rate of adaptive tanh is 1% higher than the tanh activation function.

In addition, the importance of dataset randomization related to the performance of the deep architecture neural network (DNN) was explored. Different types of randomization of the input data could make a significant improvement to the network performance. The randomness of the input data forces DNN learn meaningful features from the raw data. Based on the experimental result #3 randomization method was proposed. #3 randomization method randomized the dataset before divided into mini batches. This method allowed each mini batch does not have an equal number of samples for each class. The proposed randomization method was then applied in every classification experiment in this research.

In chapter 4, three types of neural network layers i.e. convolutional layer (CL), subsampling layer (PL) and fully connected layer (FL) were investigated. Based on the experimental results, a convolutional deep feedforward network (C-DFN) was proposed. C-DFN used a CL as a trainable feature extractor and four layers of standard DFN used as a classifier. In addition, the performance of C-DFN was further investigated and compared with another network architecture such as DFN, DBN, and C-DBN. Experimental results showed that a CL with four FLs (C-DFN) performed better than DFN, DBN, and C-DBN using three datasets.

C-DFN with LReLU reduced misclassification rate to the average of 9.41%. C-DFN with PReLU minimized the average of 9.22% misclassification rate. C-DBN with LReLU reduced misclassification rate to the average of 10.12%. C-DBN with PReLU reduced the average of 9.79% misclassification rate. Therefore, the CL acts as a trainable feature extractor improved the network performance significantly. Moreover, it reduced 14% of the trainable parameters in deep feedforward network. Based on the experimental results obtained from three different benchmark datasets, the trainable feature extractor is therefore applicable to different types of situations and applications.

In chapter 5, the proposed C-DFN was implemented for VRUs detection task. C-DFN is a single pipeline model without using image pre-processing and feature extraction method to detect VRUs. C-DFN utilizes DNN trainable feature mechanism to shorten the traditional detection process. In addition, C-DFN simplified the training process and is able to modify according to the application. C-DFN was further modified into C-DFN holistic pedestrian detection, stacked C-DFN holistic pedestrian detection, and stacked C-DFN part-based pedestrian detection. For holistic VRUs detection results, 3 stacked C-DFN obtained the best performance which reduced 10.09% of the 2 stacked C-DFN's miss rate and reduced 14.47% of the C-DFN's miss rate. The false positive per image (FPPI) of the 3 stacked C-DFN reduced 0.4% of the 2 stacked C-DFN's FPPI and reduced 12.96% of the C-DFN's FPPI. However, part-based model archived the lowest miss detection rate and less FPPI among all the detection models. Part-based model reduced 0.95% of the 3 stacked C-DFN's miss rate. The body parts verification helps the model to detect VRUs better. Therefore, the part-based model can detect the medium scale VRUs even under low-resolution images.

6.2 Future directions

The future direction and suggestions to design a better DNN for VRUs are required to improve the current drawbacks and limitations. First, the activation function is an important key to improve the network classification rate. However, the existing activation function still has room of improvement to avoid the DNN overfitting problem. In order to design a better activation function, there are several properties needed to be fulfilled: (a) nonlinearity (b) continuous differentiable, (c) range, (d) monotonic, (e) approximate identity near the origin. Adaptive linear piecewise activation function is expected to be the future trend to improve the DNN robustness and classification performance. In addition, this type of activation function can estimate the convex function and non-convex function by adjusting the linear piecewise function according to the training process of the DNN. Therefore, this type of activation function can reduce the DNN's overfitting problem effectively.

Second, DNN network structure is going deeper and wider to improve the classification rate and detection rate. However, this required supercomputer to shorten the processing time and a huge amount of training dataset is needed to tune the weights parameter of the DNN. There is a limited number of publications that discuss the design of DNN structure. Majority of the DNN structure papers only suggested adding more layers into the DNN structure. Therefore, this research investigated different types of combination network structure performance. The network structure investigation can be extended further to the new DNN structure such as AlexNet, and GoogLeNet to identify the necessity of going deeper and wider network to improve the accuracy and generalization of the network.

Third, a larger benchmark dataset such as ImageNet benchmark dataset is required to improve its detection results of the proposed C-DFN. The training dataset used in this research is insufficient to train the deep architecture neural network to achieve adequate training rate. Moreover, a code optimization technique is required to apply on the algorithm design to enhance the real time pedestrian detection. C-DFN can be further implemented to other VRUs detection systems such as cyclist and motorcyclist. C-DFN detection system can be used in other applications such as object detection and motorcyclist detection system by using object detection dataset and motorcyclist dataset. Regularization methods such as Dropout and DropConnect can

be adopted to reduce the redundant hidden neurons in the C-DFN. Besides that, the pre-processing methods to extract foreground information can be applied to shorten the region of interest (ROI) search region and simultaneously to reduce the FPPI for pedestrian detection.

Bibliography

[1] *Global status report on road safety 2018,* Licence: CC BYNC-SA 3.0 IGO, Geneva: World Health Organization, 2018.

[2] T. Ruxyn, "Death Rates On Malaysian Roads Is 3rd Highest Globally, More Than China And India," SAYS, 2017.

[3] D. Gerónimo *et al.*, "Survey of Pedestrian Detection for Advanced Driver Assistance Systems," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 32, no. 7, pp. 1239-1258, 2010.

[4] W. Santos, "13 examples of advanced driver assistance systems (ADAS)," Lifewire, Linkedin, 2017.

[5] M. Bernas *et al.*, "A Survey and Comparison of Low-Cost Sensing Technologies for Road Traffic Monitoring," *Sensors (Basel, Switzerland),* vol. 18, no. 10, pp. 3243, 2018.

[6] X. Cao, H. Qiao, and J. Keane, "A Low-Cost Pedestrian-Detection System With a Single Optical Camera," *IEEE Transactions on Intelligent Transportation Systems,* vol. 9, no. 1, pp. 58-67, 2008.

[7] H. Estl, "Advanced safety and driver assistance systems paves the way to autonomous driving," automotive, Texas Instruments, 2014.

[8] A. Gad, "Beginners Ask "How Many Hidden Layers/Neurons to Use in Artificial Neural Networks?"," *Towards Data Science*, Medium, 2018.

[9] S. Haykin, "Neural Networks A Comprehensive Foundation," Prentice Hall PTR, 1998, p. 842.

[10] P. Dollar *et al.*, "Pedestrian Detection: An Evaluation of the State of the Art," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 34, no. 4, pp. 743-761, 2012.

[11] Z. Lin, and L. S. Davis, "A Pose-Invariant Descriptor for Human Detection and Segmentation," *Computer Vision – ECCV 2008: 10th European Conference on Computer Vision, Marseille, France, October 12-18, 2008, Proceedings, Part IV*, D. Forsyth, P. Torr and A. Zisserman, eds., pp. 423-436, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[12] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A Discriminatively Trained, Multiscale, Deformable Part Model," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[13] P. Sabzmeydani, and G. Mori, "Detecting Pedestrians by Learning Shapelet Features," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1-8.

[14] S. Mittal *et al.*, "Pedestrian detection and tracking using deformable part models and Kalman filtering," in *SoC Design Conference (ISOCC)*, 2012 International, 2012, pp. 324-327.

[15] K. Pasupa, and W. Sunhem, "A comparison between shallow and deep architecture classifiers on small dataset," in *2016 8th International Conference on Information Technology and Electrical Engineering (ICITEE)*, 2016, pp. 1-6.

[16] P. K. Gaddigoudar *et al.*, "Pedestrian detection and tracking using particle filtering," in *2017 International Conference on Computing, Communication and Automation (ICCCA)*, 2017, pp. 110-115.

[17] N. Dalal, and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005.*, 2005, pp. 886-893 vol. 1.

[18] S. Munder, and D. M. Gavrila, "An Experimental Study on Pedestrian Classification," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 28, no. 11, pp. 1863-1868, 2006.

[19] C. Wojek, S. Walk, and B. Schiele, "Multi-cue onboard pedestrian detection," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2009.*, 2009, pp. 794-801.

[20] S. Munder, and D. M. Gavrila, "An Experimental Study on Pedestrian Classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 28, no. 11, pp. 1863-1868, 2006.

[21] M. Everingham *et al.*, "The Pascal Visual Object Classes (VOC) Challenge," *International Journal of Computer Vision,* vol. 88, no. 2, pp. 303-338, 2010/06/01, 2010.

[22] P. Viola, M. J. Jones, and D. Snow, "Detecting pedestrians using patterns of motion and appearance," in *Proceedings Ninth IEEE International Conference on Computer Vision*, 2003, pp. 734-741 vol.2.

[23] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision,* vol. 60, no. 2, pp. 91-110, November 01, 2004.

[24] S. Maji, A. C. Berg, and J. Malik, “Classification using intersection kernel support vector machines is efficient,” in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1-8.

[25] P. Dollár *et al.*, “Integral Channel Features,” in *British Machine Vision Conference*, London, 2009.

[26] X. Wang, T. X. Han, and S. Yan, “An HOG-LBP human detector with partial occlusion handling,” in *2009 IEEE 12th International Conference on Computer Vision*, 2009, pp. 32-39.

[27] D. Park *et al.*, “Exploring Weak Stabilization for Motion Feature Extraction,” in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2882-2889.

[28] C. Wojek, and B. Schiele, "A Performance Evaluation of Single and Multi-feature People Detection," *Pattern Recognition: 30th DAGM Symposium Munich, Germany, June 10-13, 2008 Proceedings*, G. Rigoll, ed., pp. 82-91, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[29] K. Lee *et al.*, “Human detection using Histogram of oriented gradients and Human body ratio estimation,” in *2010 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT)*, 2010, pp. 18-22.

[30] W. R. Schwartz *et al.*, “Human detection using partial least squares analysis,” in *IEEE 12th International Conference on Computer Vision*, 2009, pp. 24-31.

[31] J. Hariyono, and K. H. Jo, “Pedestrian action recognition using motion type classification,” in *2015 IEEE 2nd International Conference on Cybernetics (CYBCONF)*, 2015, pp. 129-132.

[32] H. Liu *et al.*, “A Novel Multi-Feature Descriptor for Human Detection Using Cascaded Classifiers in Static Images,” *Journal of Signal Processing Systems,* vol. 81, no. 3, pp. 377-388, December 01, 2015.

[33] P. Dollar, S. Belongie, and P. Perona, “The Fastest Pedestrian Detector in the West,” in *British Machine Vision Conference*, 2010, pp. 68.1--68.11.

[34] N. Woonhyun, H. Bohyung, and H. Joon Hee, “Improving object localization using macrofeature layout selection,” in *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops*), 2011, pp. 1801-1808.

[35] P. Dollar *et al.*, “Feature Mining for Image Classification,” in *2007 IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1-8.

[36] P. Dollár *et al.*, "Fast Feature Pyramids for Object Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 36, no. 8, pp. 1532-1545, 2014.

[37] R. Benenson *et al.*, "Seeking the Strongest Rigid Detector," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 3666-3673.

[38] S. Zhang, R. Benenson, and B. Schiele, "Filtered channel features for pedestrian detection," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 2015, pp. 1751-1760.

[39] S. Zhang, C. Bauckhage, and A. B. Cremers, "Informed Haar-Like Features Improve Pedestrian Detection," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 947-954.

[40] W. Nam, P. Dollar, and J. H. Han, "Local decorrelation for improved pedestrian detection," in P*roceedings of the 27th International Conference on Neural Information Processing Systems*, Montreal, Canada, 2014, pp. 424-432.

[41] S. Zhang *et al.*, "Exploring Human Vision Driven Features for Pedestrian Detection," *CoRR,* vol. abs/1501.06180, 2015.

[42] Y. Ding, and J. Xiao, "Contextual boost for pedestrian detection," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 2895-2902.

[43] A. D. Costea, and S. Nedevschi, "Word Channel Based Multiscale Pedestrian Detection without Image Resizing and Using Only One Classifier," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 2393-2400.

[44] S. Maji, A. C. Berg, and J. Malik, "Classification using intersection kernel support vector machines is efficient," in *2008 IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2008*, 2008, pp. 1-8.

[45] W. R. Schwartz *et al.*, "Human detection using partial least squares analysis," in *2009 IEEE 12th International Conference on Computer Vision*, 2009, pp. 24-31.

[46] S. Walk *et al.*, "New features and insights for pedestrian detection," in *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 2010, pp. 1030-1037.

[47] S. Paisitkriangkrai, C. Shen, and A. Hengel, "Efficient Pedestrian Detection by Directly Optimizing the Partial Area under the ROC Curve", *2013 IEEE International Conference on Computer Vision*, Sydney, NSW, Australia, 2013, pp. 1057-1064, doi: 10.1109/ICCV.2013.135.

[48] P. Doll, R. Appel, and W. Kienzle, "Crosstalk cascades for frame-rate pedestrian detection," in *Proceedings of the 12th European conference on Computer Vision - Volume Part II*, Florence, Italy, 2012, pp. 645-659.

[49] J. Marín *et al.*, "Random Forests of Local Experts for Pedestrian Detection," in *2013 IEEE International Conference on Computer Vision*, 2013, pp. 2592-2599.

[50] D. Park, D. Ramanan, and C. Fowlkes, "Multiresolution Models for Object Detection," *Computer Vision – ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part IV*, K. Daniilidis, P. Maragos and N. Paragios, eds., pp. 241-254, Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

[51] S. Paisitkriangkrai, C. Shen, and A. v. d. Hengel, "Pedestrian Detection with Spatially Pooled Features and Structured Ensemble Learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 38, no. 6, pp. 1243-1257, 2016.

[52] A. Leithy, M. N. Moustafa, and A. Wahba, "Cascade of Complementary Features for Fast and Accurate Pedestrian Detection," in *2010 Fourth Pacific-Rim Symposium on Image and Video Technology (PSIVT)*, 2010, pp. 343-348.

[53] Z. Wang *et al.*, "A High Accuracy Pedestrian Detection System Combining a Cascade AdaBoost Detector and Random Vector Functional-Link Net," *The Scientific World Journal,* vol. 2014, pp. 7, 2014.

[54] W. Ke *et al.*, "Pedestrian detection via PCA filters based convolutional channel features," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP*), 2015, pp. 1394-1398.

[55] L. Guo *et al.*, "Pedestrian detection for intelligent transportation systems combining AdaBoost algorithm and support vector machine," *Expert Systems with Applications,* vol. 39, no. 4, pp. 4274-4286, 2012/03/01/, 2012.

[56] A. Prioletti *et al.*, "Part-Based Pedestrian Detection and Feature-Based Tracking for Driver Assistance: Real-Time, Robust Algorithms, and

Evaluation," *Intelligent Transportation Systems, IEEE Transactions on,* vol. 14, no. 3, pp. 1346-1359, 2013.

[57] H. Shimizu, and T. Poggio, "Direction estimation of pedestrian from multiple still images," in *2004 IEEE Intelligent Vehicles Symposium*, 2004, pp. 596-600.

[58] X. Zhang *et al.*, "Pedestrian detection based on hierarchical co-occurrence model for occlusion handling," *Neurocomputing,* vol. 168, pp. 861-870, 2015/11/30/, 2015.

[59] A. Ess, B. Leibe, and L. V. Gool, "Depth and Appearance for Mobile Scene Analysis," in *2007 IEEE 11th International Conference on Computer Vision*, 2007, pp. 1-8.

[60] A. Bar-Hillel *et al.*, "Part-Based Feature Synthesis for Human Detection," *Computer Vision – ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part IV*, K. Daniilidis, P. Maragos and N. Paragios, eds., pp. 127-142, Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

[61] O. Wanli, and W. Xiaogang, "Single-Pedestrian Detection Aided by Multi-pedestrian Detection," in *2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 3198-3205.

[62] W. Ouyang, X. Zeng, and X. Wang, "Single-Pedestrian Detection Aided by Two-Pedestrian Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 37, no. 9, pp. 1875-1889, 2015.

[63] W. Ouyang, and X. Wang, "Joint Deep Learning for Pedestrian Detection," in *2013 IEEE International Conference on Computer Vision*, 2013, pp. 2056-2063.

[64] B. Li, Y. Chen, and F. Wang, "Pedestrian Detection Based on Clustered Poselet Models and Hierarchical and or Grammar," in *IEEE Transactions on Vehicular Technology*, 2015, pp. 1435-1444.

[65] Y. Junjie *et al.*, "Robust Multi-resolution Pedestrian Detection in Traffic Scenes," in *Computer Vision and Pattern Recognition (CVPR)*, 2013 IEEE Conference on, 2013, pp. 3033-3040.

[66] M. Pedersoli *et al.*, "Toward Real-Time Pedestrian Detection Based on a Deformable Template Model," *Intelligent Transportation Systems, IEEE Transactions on,* vol. 15, no. 1, pp. 355-364, 2014.

[67] M. Mathias *et al.*, "Handling Occlusions with Franken-Classifiers," in *2013 IEEE International Conference on Computer Vision,* 2013, pp. 1505-1512.

[68] S. Zhang, C. Bauckhage, and A. B. Cremers, "Efficient Pedestrian Detection via Rectangular Features Based on a Statistical Shape Model," *IEEE Transactions on Intelligent Transportation Systems,* vol. 16, no. 2, pp. 763-775, 2015.

[69] G. Chen *et al.*, "Detection Evolution with Multi-Order Contextual Co-occurrence," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 1798-1805.

[70] D. Levi, S. Silberstein, and A. Bar-Hillel, "Fast Multiple-Part Based Object Detection Using KD-Ferns," in *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 947-954.

[71] P. F. Felzenszwalb *et al.*, "Object Detection with Discriminatively Trained Part-Based Models," *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. 32, no. 9, pp. 1627-1645, 2010.

[72] W. Ouyang, and X. Wang, "Single-Pedestrian Detection Aided by Multi-pedestrian Detection," in 2*013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 3198-3205.

[73] Y. Chen *et al.*, "Pedestrian detection by learning a mixture mask model and its implementation," *Information Sciences,* vol. 372, pp. 148-161, 2016/12/01/, 2016.

[74] M. Enzweiler, and D. M. Gavrila, "Monocular Pedestrian Detection: Survey and Experiments," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 31, no. 12, pp. 2179-2195, 2009.

[75] W. Ouyang, and X. Wang, "A discriminative deep model for pedestrian detection with occlusion handling," in *2012 IEEE Conference on Computer Vision and Pattern Recognition,* 2012, pp. 3258-3265.

[76] W. Ouyang, X. Zeng, and X. Wang, "Modeling Mutual Visibility Relationship in Pedestrian Detection," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 3222-3229.

[77] Z. Xingyu, O. Wanli, and W. Xiaogang, "Multi-stage Contextual Deep Learning for Pedestrian Detection," in *2013 IEEE International Conference on Computer Vision (ICCV)*, 2013, pp. 121-128.

[78] P. Sermanet *et al.*, “Pedestrian Detection with Unsupervised Multi-stage Feature Learning,” in *2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 3626-3633.

[79] P. Luo *et al.*, “Switchable Deep Network for Pedestrian Detection,” in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 899-906.

[80] A. Angelova, A. Krizhevsky, and V. Vanhoucke, “Pedestrian Detection with a Large-Field-Of-View Deep Network,” in *ICRA* 2015.

[81] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” in *NIPS*, 2012.

[82] L. Wang, B. Zhang, and W. Yang, "Boosting-Like Deep Convolutional Network for Pedestrian Detection," *Biometric Recognition: 10th Chinese Conference, CCBR 2015, Tianjin, China, November 13-15, 2015, Proceedings*, J. Yang *et al.*, eds., pp. 581-588, Cham: Springer International Publishing, 2015.

[83] R. Benenson *et al.*, "Ten Years of Pedestrian Detection, What Have We Learned?," *Computer Vision - ECCV 2014 Workshops: Zurich, Switzerland, September 6-7 and 12, 2014, Proceedings, Part II*, L. Agapito, M. M. Bronstein and C. Rother, eds., pp. 613-627, Cham: Springer International Publishing, 2015.

[84] J. Hosang *et al.*, “Taking a deeper look at pedestrians,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 2015, pp. 4073-4082.

[85] R. Girshick *et al.*, “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation,” in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 580-587.

[86] Y. Tian *et al.*, “Deep Learning Strong Parts for Pedestrian Detection,” in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1904-1912.

[87] Y. Tian *et al.*, “Pedestrian detection aided by deep learning semantic tasks,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 5079-5087.

[88] D. Tomè *et al.*, "Deep Convolutional Neural Networks for pedestrian detection," *Signal Processing: Image Communication,* vol. 47, pp. 482-489, 2016/09/01/, 2016.

[89] L. Hailong, W. Zhendong, and Z. Jianwu, "Pedestrian detection based on deep learning model," in *2016 9th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI),* 2016, pp. 796-800.

[90] S.-I. Jung, and K.-S. Hong, "Deep network aided by guiding network for pedestrian detection," *Pattern Recognition Letters,* vol. 90, pp. 43-49, 2017/04/15/, 2017.

[91] X. Jiang *et al.*, "Speed up deep neural network based pedestrian detection by sharing features across multi-scale models," *Neurocomputing,* vol. 185, pp. 163-170, 2016/04/12/, 2016.

[92] A. Uçar, Y. Demir, and C. Güzeliş, "Moving towards in object recognition with deep learning for autonomous driving applications," in *2016 International Symposium on INnovations in Intelligent SysTems and Applications (INISTA)*, 2016, pp. 1-5.

[93] O. Abdel-Hamid *et al.*, "Convolutional Neural Networks for Speech Recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing,* vol. 22, no. 10, pp. 1533-1545, 2014.

[94] Xavier Glorot, and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Artificial intelligence and statistics (AISTATS)*, Sardinia, Italy, 2010.

[95] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computing,* vol. 18, no. 7, pp. 1527-1554, 2006.

[96] Chenghao Cai *et al.*, "Deep neural network with multistate activation functions," *Computational Intelligence and Neuroscience,* vol. 2015, pp. 1-10, 2015.

[97] Bing Xu, Ruitong Huang, and M. Li, "Revised saturated activation functions," in *International conference on Learning Representation*, 2016.

[98] Xavier Glorot, Antoine Bordes, and Y. Bengio, "Deep Sparse Rectified Neural Network," in I*nternational conference on Artificial Intelligence and Statistics (AISTATS)*, Fort Lauderdale, FL, USA, 2011.

[99] Andrew L. Maas, Awni Y. Hannun, and A. Y. Ng, “Retifier Nonlinearities Improve Neural Network Acostic Models,” in *International conference on Machine Learning*, Atlanta, Georgia, USA, 2013.

[100] Xiaojie Jin *et al.*, “Deep Learning with s-shaped rectified linear activation units ” *CoRR,* vol. abs/1512.07030, 13 Aug 2018, 2015.

[101] I. J. Goodfellow *et al.*, “Maxout networks,” *arXiv preprint arXiv:1302.4389*, 2013.

[102] Kaiming He *et al.*, “Delving deep into rectifiers: surpassing human-level performance on ImageNet classification,” *CoRR,* vol. abs/1502.01852, 17 Apr 2019, 2015.

[103] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, “Fast and Accurate Deep Network Learning by Exponential Linear Units,” *CoRR,* vol. CoRR, 2015.

[104] N. Srivastava *et al.*, “Dropout: a simple way to prevent neural networks from overfitting,” *J. Mach. Learn. Res.,* vol. 15, no. 1, pp. 1929-1958, 2014.

[105] L. Wan *et al.*, “Regularization of neural networks using dropconnect,” in *Proceedings of the 30th International Conference on International Conference on Machine Learning* - Volume 28, Atlanta, GA, USA, 2013, pp. III-1058-III-1066.

[106] G. Hinton, "A Practical Guide to Training Restricted Boltzmann Machines," *Neural Networks: Tricks of the Trade*, Lecture Notes in Computer Science G. Montavon, G. Orr and K.-R. Müller, eds., pp. 599-619: Springer Berlin Heidelberg, 2012.

[107] M. M. A. Ghosh, and A. Y. Maghari, “A Comparative Study on Handwriting Digit Recognition Using Neural Networks,” in *2017 International Conference on Promising Electronic Technologies (ICPET)*, 2017, pp. 77-81.

[108] H. T. Zheng *et al.*, “Automatic Generation of News Comments Based on Gated Attention Neural Networks,” *IEEE Access,* vol. 6, pp. 702-710, 2018.

[109] A. Krizhevsky, S. Ilya, and H. G. E, “ImageNet Classification with Deep Convolutional Neural Networks,” *Neural Information Processing Systems*, pp. 1097-1105, 2012.

[110] J. Deng *et al.*, “ImageNet: A large-scale hierarchical image database,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248-255.

[111] M. D. Zeiler, and R. Fergus, “Visualizing and Understanding Convolutional Networks,” *CoRR,* vol. abs/1311.2901, 2013.

[112] K. Simonyan, and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” *CoRR,* vol. abs/1409.1556, 2014.

[113] C. Szegedy *et al.*, “Going deeper with convolutions,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1-9.

[114] K. He *et al.*, “Deep Residual Learning for Image Recognition,” *CoRR,* vol. abs/1512.03385, 2015.

[115] Y. Lecun *et al.*, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE,* vol. 86, no. 11, pp. 2278-2324, 1998.

[116] G. Klambauer *et al.*, “Self-Normalizing Neural Networks,” *CoRR,* vol. abs/1706.02515, 2017.

[117] I. Goodfellow *et al.*, “Maxout Networks,” in *Proceedings of the 30th International Conference on Machine Learning, Proceedings of Machine Learning Research*, 2013, pp. 1319--1327.

[118] M. Lin, Q. Chen, and S. Yan, “Network In Network,” *CoRR,* vol. abs/1312.4400, 2013.

[119] H. Lee *et al.*, “Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations,” in *Proceedings of the 26th Annual International Conference on Machine Learning*, Montreal, Quebec, Canada, 2009, pp. 609-616.

[120] J. Bouvrie, "Notes on Convolutional Neural Networks," *cogprints.org*, 2006.

[121] N.Petkov, and M.B.Wieling, "Gabor filter for image processing and computer vision," *Intelligent Systems*, Matlabserver, 2008.

[122] J. L. Chu, and A. Krzyżak, “Analysis of Feature Maps Selection in Supervised Learning Using Convolutional Neural Networks,” in *Advances in Artificial Intelligence*, Cham, 2014, pp. 59-70.

[123] M. M. Lau, and K. H. Lim, “Investigation of activation functions in deep belief network,” in *2017 2nd International Conference on Control and Robotics Engineering (ICCRE)*, 2017, pp. 201-206.

[124] P. Geismann, and G. Schneider, "A two-staged approach to vision-based pedestrian recognition using Haar and HOG features." *2008 IEEE Intelligent Vehicles Symposium*, Eindhoven, Netherlands, 2008, pp. 554-559, doi:10.1109/IVS.2008.4621148.

[125] Y. Zhang, W. Liu, C. Mo and Z. Li, "Pedestrian detection in complex scene using full binary tree classifiers based on locally assembled Binary Haar-like features." *2011 9th World Congress on Intelligent Control and Automation*, Taipei, Taiwan, 2011, pp. 1180-1184, doi:10.1109/WCICA.2011.5970702.

[126] Z. Jiang, D. Q. Huynh, W. Moran, S. Challa and N. Spadaccini, "Multiple Pedestrian Tracking Using Colour and Motion Models." *2010 International Conference on Digital Image Computing: Techniques and Applications*, Sydney, NSW, Australia, 2010, pp. 328-334, doi: 10.1109/DICTA.2010.63.

[127] B. Zeng, G. Wang, and X. Lin, "Color self-similarity feature based real-time pedestrian detection," *Journal of Tsinghua University (Science and Technology)*, 2012, pp. 571-574.

[128] G. Gan, and J. Cheng, "Pedestrian Detection Based on HOG-LBP Feature," in *2011 Seventh International Conference on Computational Intelligence and Security*, 2011, pp. 1184-1187.

[129] Z. Chen, K. Chen, and J. Chen, "Vehicle and Pedestrian Detection Using Support Vector Machine and Histogram of Oriented Gradients Features," in *2013 International Conference on Computer Sciences and Applications*, 2013, pp. 365-368.

[130] Z. Huang, "Vehicle Pedestrian Detection Algorithm Based on AdaBoost," in *2015 International Conference on Intelligent Transportation, Big Data and Smart City*, 2015, pp. 973-976.

[131] L. Wang, B. Zhang, and W. Yang, "Boosting-Like Deep Convolutional Network for Pedestrian Detection," *Biometric Recognition.* pp. 581-588.

[132] C. Xiaogang *et al.*, "Pedestrian detection with deep convolutional neural network," *Springer International Publishing Switzerland*, 2015.

[133] Y. Tian *et al.*, "Pedestrian detection aided by deep learning semantic tasks," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5079-5087, 2015.

[134] L. Bo-Yao, and C. S. Chen, "Two Parallel Deep Convolutional Neural Networks for pedestrian detection," in *2015 International Conference on Image and Vision Computing New Zealand (IVCNZ)*, 2015, pp. 1-6.

[135] Driving Test Success | Driving Theory Test & Learner Driver Revision. "*Stopping Distances and The Theory Test 2018*", 2018. [online] Available at:

<https://driving-test-success.myshopify.com/page/stopping-distances-and-the-theory-test> [Accessed 20 March 2021].

Appendix A

A1: Deep Belief Network

Deep belief network consists of a generative architecture i.e. restricted Boltzmann machine (RBM) and a discriminative learning algorithm i.e. stochastic gradient descent. It is made up by stacking multiple layer of RBM. The training process of DBN had been divided into pre-training and fine-tuning process. Before stacking the RBM, every single layer of RBM is pre-trained unsupervised using contrastive divergence algorithm. After pre-training every layer of the RBM, the RBMs are stacked together to form a DBN. Finally, fine-tune the DBN using stochastic gradient descent backpropagation algorithm.

A1.1: Pre-training

Restricted Boltzmann Machine (RBM) is an undirected graphical model which consists of a visible layer and a hidden layer as shown in Fig. A1. RBM has a weight connection between visible and hidden units but there is no weight connection in between the visible-visible and hidden-hidden units.

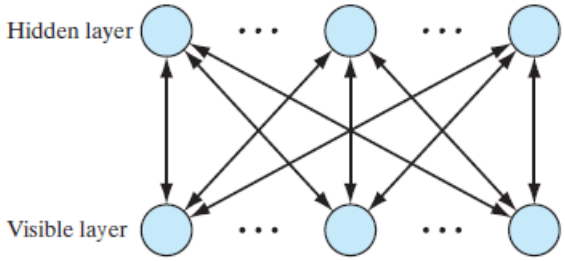


Figure A1: *Restricted Boltzmann Machine.*

The energy of the joint configuration for the visible and hidden units is defined as:

$$E(v,h;\theta) = -\sum_{j=1}^{J}\sum_{i=1}^{I} w_{ji}v_i h_j - \sum_{i}^{I} b_i v_i - \sum_{j}^{J} c_j h_j \qquad \text{(A1.1)}$$

where $\theta = \{w, b, c\}$ are the model parameters: w_{ji} denotes the symmetrical weights connection between visible units, i and hidden units, j; b_i and c_j are the bias terms for visible layer and hidden layer respectively. The marginal probability of the modal assigned to a visible vector, v is

$$p(v;\theta) = \frac{\sum_h \exp(-E(v,h;\theta))}{z} \tag{A1.2}$$

where $z = \sum_v \sum_h \exp(-E(v,h;\theta))$ is a normalization factor or partition function. The derivative of the log-likelihood with respect to the model parameter θ is:

$$\frac{\partial\, log\, p(v;\theta)}{\partial\theta} = E_h\left[\frac{\partial E(v^{(t)},h)}{\partial\theta}\middle| v^{(t)}\right] - E_{v,h}\left[\frac{\partial E(v,h)}{\partial\theta}\right] \tag{A1.3}$$

where $E_h[\cdot]$ denotes an expectation with respect to the data distribution and $E_{v,h}[\cdot]$ is an expectation with respect to the distribution by the model. However, the expectation $E_{v,h}[\cdot]$ is hard to compute. Therefore, contrastive divergence is used to approximate the expectation $E_{v,h}[\cdot]$ in Eq. (A1.3).

Contrastive divergence (CD) is a training algorithm which consists of one step Gibbs sampling. It updates all the hidden units using Eq. (A1.4) followed by updating all the visible units in parallel using Eq. (A1.5) to obtain the reconstruction samples by the model. The conditional distribution over a hidden neurons h and visible vector v in the RBM is defined as follows:

$$\mathrm{p}(h_{0j} = 1|v_0) = \frac{1}{1 + \exp\left(-\left(c_j + \sum_j w_{ji} v_i\right)\right)} \tag{A1.4}$$

$$\mathrm{p}(v_{1i} = 1|h_0) = \frac{1}{1 + \exp\left(-\left(b_i + \sum_i h_{0j} w_{ji}\right)\right)} \tag{A1.5}$$

where x_0 and h_{0j} is the sampling for visible and hidden layer at time 0 respectively. x_{1i} is the reconstructed visible samples by the model. The weights is then updated based on the following equation:

$$\Delta w_{ji} = \eta\left(E_{data}(v_i h_j) - E_{model}(v_i h_j)\right)$$

$$= \eta(\mathrm{p}(h_{0j} = 1|x_0)x_0' - \mathrm{p}(h_{1j} = 1|x_1)\mathrm{p}(x_{1i} = 1|h_0)') \tag{A1.6}$$

where η is the learning rate, x_0' is the transposed visible layer. The input bias *b* updates in RBM is:

$$\Delta b = \eta(\mathrm{p}(h_{0j} = 1|x_0) - \mathrm{p}(h_{1j} = 1|x_1)) \tag{A1.7}$$

The hidden bias *c* updates in RBM:

$$\Delta c = \eta(x_0 - \mathrm{p}(x_{1i} = 1|h_0)) \tag{A1.8}$$

where x_0 is the input data.

After training a layer of RBM, a new layer of RBM is stacked on the top of the trained RBM and the input layer of the new RBM is the hidden layer of the previous RBM. This training method is also known as the greedy layer-wise pre-training. The overall procedure of this pre-training method is shown as follows:

1) First, an RBM (**v**, $\mathbf{h_1}$) is trained by using CD learning procedure and all parameters of trained RBM are fixed.
2) Then, a new RBM (**h1**, **h2**) is stacked on the top of trained RBM and the same procedure in step (1) is repeated.
3) After all layers of DBN is pre-trained, fine-tuning in backpropagation manner is carried out to adjust the weights layer by layer jointly to improve the performance of the network.

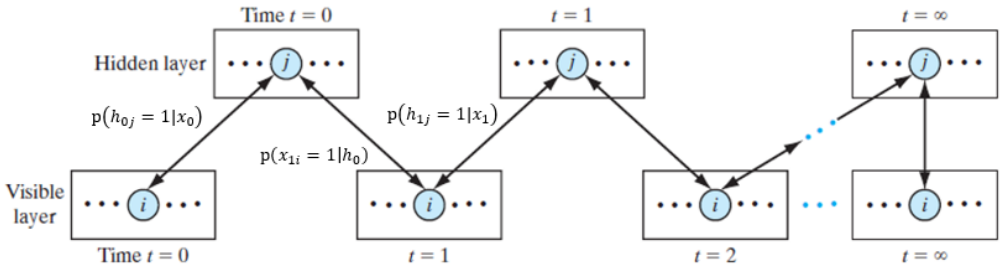


Figure A2: Gibbs sampling.

A1.2: Fine-tuning

Fine-tuning is the training process applied for all the layers in deep belief network (DBN) after the pre-training process is completed. In this chapter, stochastic gradient descent training algorithm is used to fine-tune all the weights parameter in DBN. The error energy of the output layer neurons, *k* is given by:

$$E(n) = \frac{1}{2}\sum_{k \in C} e_k^2(n) \tag{A1.9}$$

where C denoted all neurons in the output layer of DBN. An average error is computed by summing $E(n)$ over all *n* and divided by the total number of example *N*.

$$E_{average} = \frac{1}{N}\sum_{n=1}^{N} E(n) \tag{A1.10}$$

The error signal $e(n)$ at the output $y(n)$ at n training example is defined by:

$$e_k(n) = d_k(n) - y_k(n) \tag{A1.11}$$

where $d_k(n)$ is a vector of the targeted output for n training examples. The backpropagation algorithm applies a correction $\Delta w_{ji}(n)$ to weight $w_{ji}(n)$ by using the delta rule:

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n) \tag{A1.12}$$

where η denotes learning rate, $y_i(n)$ input signal of neuron j. The correction weight by adding momentum to increase the speed of learning is given as follows:

$$\Delta w_{ji}(n) = \alpha \Delta w_{ji}(n-1) + \eta \delta_j(n) y_i(n) \tag{A1.13}$$

where α denotes the momentum constant, $\Delta w_{ji}(n-1)$ is the last correction weight. The local gradient for the output layer $\delta_k(n)$ which is equal to the product of the corresponding error $e_k(n)$ and the derivative $y'_k(v_k(n))$ of the activation function.

$$\delta_k(n) = e_k(n) y'_k(v_k(n)) \tag{A1.14}$$

The local gradient for hidden layer $\boldsymbol{\delta_j(n)}$ is defined by:

$$\delta_j(n) = y'_j\big(v_j(n)\big) \sum_k \delta_k(n) w_{kj}(n) \tag{A1.15}$$

where the local gradient $\delta_j(n)$ is the changes required in weights.