

School of Earth and Planetary Science

**Farm to Table Meteorites: An End to End Exploration of the Solar System's Past,
Present, and Future.**

Seamus Lon Anderson

ORCID: 0000-0002-8914-3264

This Thesis is Presented for the Degree of

Doctor of Philosophy

of

Curtin University

April 2023

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

Signed: Seamus L. Anderson

Date: September 8 2022

A handwritten signature in black ink, appearing to read 'Seamus L. Anderson'. The signature is fluid and cursive, with a large initial 'S' and a long, sweeping tail.

Table of Contents

Title Page	1
Table of Contents	3
Acknowledgements	5
Attributions.....	7
Copyrights	14
Chapter I: Introduction	16
References	16
Chapter II: Significance and Methodology	17
Abstract.....	17
Significance	17
Meteorite Searching with Drones and Machine Learning (MAPS and ApJL).....	17
The Murrili (H5) Meteorite (MAPS).....	18
Space Resource Utilization Using The Silicate-Sulfuric Acid Process.....	18
Methodology.....	19
Meteorite Searching with Drones and Machine Learning	19
Meteorite Characterization	26
The Silicate Sulfuric Acid Process	31
Chapter III: Meteorite Searching Pt. I.....	42
Abstract.....	42
INTRODUCTION	42
The Desert Fireball Network	43
Meteorite Recovery	43
Previous Drone-Meteorite Recovery Methods	44
METHODS	45
Machine Learning Software	46
Model Detection Sorting Interface	49
RESULTS.....	51
DISCUSSION AND FUTURE WORK	53
Chapter IV: Meteorite Searching Pt. II (Published in: <i>Astrophysical Journal Letters</i> , 930(2), p.L25).....	58
1. Introduction.....	58
2. Methods	60
2.1 Fireball Observations and Modelling	60

2.2 Drone Surveying and Machine Learning.....	61
3. Results and Discussion.....	64
Chapter V: The Murrili (H5) Meteorite.....	69
ABSTRACT	70
INTRODUCTION	70
SAMPLE ALLOCATION AND ANALYTICAL METHODS	72
RESULTS.....	77
DISCUSSION	91
CONCLUSIONS	92
Chapter VI: The Silicate Sulfuric Acid Process (Published in: <i>Acta Astronautica</i> , 188, p.57-63)	99
Abstract.....	99
1. Introduction	99
1.1 Sulfur Availability	100
1.2 Silicate-Bound Resources.....	101
2. Physical and Chemical Pathways of the SSAP	102
2.1 Pre-processing	103
2.2 Sulfuric Acid Synthesis	104
2.3 Silicate Dissolution and Silica Extraction	105
2.4 Metal and Oxide Production.....	106
3. Results and Discussion.....	108
3.1 Products and Uses.....	110
3.2 Engineering and Logistical Considerations	111
3.3 Comparison to other ISRU Methodologies	111
4. Conclusions.....	112
5. Acknowledgments.....	112
5. References.....	113
Supplementary Materials	118
Chapter VII: Conclusions	125
Work Done	125
Work To Do.....	125
Appendix A (Meteorite Searching Code).....	127
Appendix B (Meteorite Classification Code)	209
Appendix C (ISRU Code)	225

Acknowledgements

Mom, Dad, Lukka, and Neave - Where would I be without all of you? Thank you for all of your love and support throughout my life. All my love :)

Martin - The Bushmaster General, how could I have done this without you? You were there at every step of this process, constantly pushing me further, acknowledging my progress while looking to the future asking questions that required me to think about the larger picture, and critical details simultaneously. You taught me everything I know about not just surviving but thriving in the most difficult settings I have ever known. Without you I would have fallen short, leaving this work incomplete. No words can comprehensively describe my thanks, though I hope those above have given at least some indication. Thank You^{Thank You}

Phil M - I cannot thank you enough for taking a chance on me, I was completely on the outside of research. You opened the door and gave me my first opportunity to enter the field that would eventually become the focus of my career: Space Mining. I hope that you continue to inspire and educate new students into the field of planetary science, and I hope to work with you in the future. Thank you.

Addie - Your guidance, support, and dissatisfaction with anything short of excellence forced me to achieve more than I thought I was capable of. Without your help I seriously doubt that I would have been accepted to a PhD program. I hope you continue to mentor and inspire students, as you did for me. Thank you.

Gretchen and Phil - You taught me everything I know about meteorites and so much more about the academic process. I know you will continue to pass this knowledge on to others. Thank You.

Ellie and Hadrien - You two are the older academic siblings that I never had. You are among the smartest, most caring and positive people I know. I learn from you two all the time and I know that you two will continue to enrich the lives of all those around you. Merci.

Andrea and Milana - Both of you made what is normally one of the most stressful times in our lives, to be nothing short of spectacular and splendid. I consider you both to be lifelong friends, Hvala!

Ben - No words can express how happy I am to have you in my professional and personal life, offering a fresh and wise perspective in any situation. Thank you, and Level Up!

Erika - You have been nothing short of supportive and generous both before and during my PhD (not to mention the internship). Thank you for all of your guidance and continued collaboration, the latter of which will hopefully be in person soon!

Billy, Dawn, Erwin and Lizzie - The Corner Meadow Mafia, without you guys talking up undergrad research, I'm not sure I would've ever wanted to get involved. You are all great friends and I hope that you continue to bring the same warmth to all those around you. <3

Mr. Chimiak - Educator Ultissimo, although you were my latin teacher, you taught me so much more. You imparted upon me the importance of recognizing patterns and applying one system of understanding to another. You stretched and strengthened my mind to teach me how to learn in any situation, a skill I continue to use. You are an inspiration, thank you.

Mr. Moore - The math comedian, because of you I graduated from liking math to loving it. After your Pre-Calculus class I decided to major in a STEM degree instead of history, a decision that obviously led to this

thesis. Your sense of humor and personification of the subject was, and hopefully still is, an inspiration and laughter to all. Thank you.

Ms. Bowen - Without your poignant, intelligent, and progressive illustration of the social constructs that semi-visibly define the world around us, I would be a much more insular and ignorant member of society. You taught me to not only consider the present but also the past and how it is intimately intertwined with how we shape our future. Thank you.

Patrick - Out of respect for the friendship we had, I acknowledge that our shared experiences eased the difficulty of much of this thesis.

Evan and Todd - You're Obsessed with me and I love it! You both know this, but I'll repeat it here: my time living with you two has been the most enriching and relaxing time in Australia. Love you both, and I hope we continue to be in each other's lives <3

Ms. Ahrens - We didn't really like each other. You laughed in my face after I told you that I was majoring in Physics, so thank you for being that special sort of motivation.

Josh, Julie, Wesley, Cody, Stephanie, Sumayya, Lex, Jeff, Jacob, Akbar, Gillian, Emily, Mike, Sean and everyone at the CMR - I learned so much from all of you, and am so grateful for the experiences we shared. Thank You.







Anthony, Renae, Martin C., John, Francesca, Rob, Taryn, Jack, Nat, Raiza, Fergus, Stuart, Trent, Katie, Kathy, Nick, Aaron, Hugo, and everyone at the EPS - Thanks to all of you for all of the side chats, griping sessions, forgotten memories and so much more.

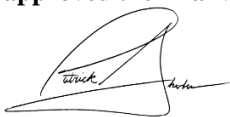
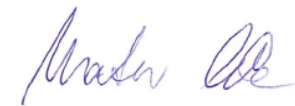

The anonymous reviewers - Thank you for your honesty, time, and effort that you invested into reviewing this work.

You - Thank you for reading, or at the very least skimming this thesis. I hope that you may take some knowledge from these works and apply them towards solving the problems of tomorrow.

Attributions

Table 1. Attributions for Chapter III (Drone Searching - Meteoritics and Planetary Science)

Co-Author	Conception and Design	Acquisition of Data and Method	Data Conditioning Manipulation	Analysis and Statistical Method	Interpretation and Discussion	Total Contribution
Martin Towner	✓	✓			✓	7 %
I acknowledge that these represent my contribution to the above research output and I have approved the final version. Signed: 						
Phil Bland	✓					5 %
I acknowledge that these represent my contribution to the above research output and I have approved the final version. Signed: 						
Christopher Haikings	✓	✓	✓			3 %
I acknowledge that these represent my contribution to the above research output and I have approved the final version. Signed: 						
William Volante	✓					1 %
I acknowledge that these represent my contribution to the above research output and I have approved the final version. Signed: 						
Eleanor Sansom	✓	✓				1 %
I acknowledge that these represent my contribution to the above research output and I have approved the final version. Signed: 						
Hadrien Devillepoix	✓	✓				1 %
I acknowledge that these represent my contribution to the above research output and I have approved the final version. Signed: 						

Patrick Shober	✓					1 %
I acknowledge that these represent my contribution to the above research output and I have approved the final version. Signed:						
						
Benjamin Hartig	✓					1 %
I acknowledge that these represent my contribution to the above research output and I have approved the final version. Signed:						
<i>Benjamin Hartig</i>						
Martin Cupák	✓					1 %
I acknowledge that these represent my contribution to the above research output and I have approved the final version. Signed:						
						
Trent Jansen-Sturgeon	✓					1 %
I acknowledge that these represent my contribution to the above research output and I have approved the final version. Signed:						
						
Robert Howie	✓					0.5 %
I acknowledge that these represent my contribution to the above research output and I have approved the final version. Signed:						
<i>Robert Howie</i>						
Gretchen Benedix	✓					0.5 %
I acknowledge that these represent my contribution to the above research output and I have approved the final version. Signed:						



<i>Yvonne K Benedict</i>						
Geoff Deacon	✓	✓				2 %
I acknowledge that these represent my contribution to the above research output and I have approved the final version. Signed:						
						
Total Contribution from Co-Authors						25 %

Table 2. Attributions for Chapter IV (Drone Searching - Astrophysical Journal Letters)

Co Author	Conception and Design	Acquisition of Data and Method	Data Conditioning Manipulation	Analysis and Statistical Method	Interpretation and Discussion	Total Contribution
Martin Towner	✓	✓		✓		10 %
I acknowledge that these represent my contribution to the above research output and I have approved the final version. Signed:						
<i>M Towner</i>						
John Fairweather		✓		✓		5 %
I acknowledge that these represent my contribution to the above research output and I have approved the final version. Signed:						
						
Phil Bland	✓					0.5 %
I acknowledge that these represent my contribution to the above research output and I have approved the final version. Signed:						
<i>Phil Bland</i>						
Hadrien Devillepoix	✓	✓	✓	✓		2.5 %




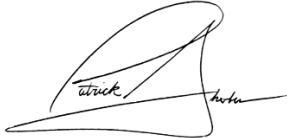

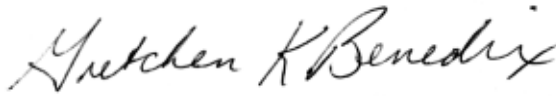
I acknowledge that these represent my contribution to the above research output and I have approved the final version. Signed:						
						
Eleanor Sansom	✓					0.5 %
I acknowledge that these represent my contribution to the above research output and I have approved the final version. Signed:						
						
Martin Cupák	✓					0.5 %
I acknowledge that these represent my contribution to the above research output and I have approved the final version. Signed:						
						
Patrick Shober	✓					0.5 %
I acknowledge that these represent my contribution to the above research output and I have approved the final version. Signed:						
						
Gretchen Benedix	✓					0.5 %
I acknowledge that these represent my contribution to the above research output and I have approved the final version. Signed:						
						
Total Contribution from Co-Authors						20 %

Table 3. Attributions for Chapter V (The Murrili (H5) meteorite - Meteoritics and Planetary Science). The Attributions for this chapter are slightly more complex. Since orbital meteorites such as Murrili offer an incredibly rare view into solar system science, we as meteoriticists and planetary scientists are obligated to study these rocks in as many ways as possible. This is best accomplished by sending samples of the meteorite to world-class experts, each of whom performs their specialized analysis to help the lead investigator piece

together the puzzle that is each meteorite’s history. As such, I collected no data. I was however, with support from Gretchen Benedix, responsible for writing of manuscript as well as interpreting the results from the 10 analyses performed upon this meteorite to discover the history of this meteorite. I must also point out that much of the original text for each method and its result was written by each of the collaborators, which I then edited or reduced to fit together as a cohesive story. Aside to this, some of the authors listed on this publication (Bland, Paxman, Towner, Cupák, Strangeway, Stuart, Jansen-Sturgeon, Devillepoix, Sansom, Howie), did not assist with the geologic analysis (the main focus of this paper) though were crucial enough to the recovery of the meteorite (although included in this paper, it is the main subject of focus in Bland et al., 2016; Sansom et al., 2020*) to go beyond a simple Acknowledgement. And Tragically, before this manuscript was fully published, Alex Bevan passed away. He co-authored this paper by providing his invaluable analysis of the meteorite’s thin sections and graciously helped me personally by providing an introduction to thin section analysis. For all these reasons I have grouped the contributions from most authors together.

Co Author	Conception and Design	Acquisition of Data and Method	Data Conditioning Manipulation	Analysis and Statistical Method	Interpretation and Discussion	Total Contribution
Gretchen Benedix	✓	✓	✓	✓	✓	15 %
I acknowledge that these represent my contribution to the above research output and I have approved the final version. Signed: 						
Lucy Forman		✓	✓			(see below)
Luke Daly		✓	✓			
Richard Greenwood		✓	✓			
Ian Franchi		✓	✓			
Jon Friedrich		✓	✓			
Robert Macke		✓	✓			
Sean Wiggins		✓	✓			
Daniel Britt		✓	✓			
Sean Cadogan		✓	✓			



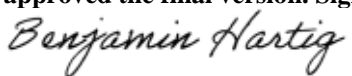
Matthias Meier		✓	✓			35 %
Colin Maden		✓	✓			
Henner Busemann		✓	✓			
Kees Welten		✓	✓			
Marc Caffee		✓	✓			
Fred Jourdan		✓	✓			
Celia Mayers		✓	✓			
Trudi Kennedy		✓	✓			
Belinda Godel		✓	✓			
Lionel Esteban		✓	✓			
Kelly Merigot		✓	✓			
Alex Bevan	✓	✓	✓			
Phil Bland	✓					
Jonathan Paxman	✓					
Martin Towner	✓					
Martin Cupák	✓					
Eleanor Sansom	✓					
Robert Howie	✓					
Hadrien Devillepoix	✓					
Trent Jansen-Sturgeon	✓					



D Stuart	✓					
D Strangeway	✓					
Total Contribution from Co-Authors						50 %

*Bland, P.A., Towner, M.C., Sansom, E.K., Devillepoix, H., Howie, R.M., Paxman, J.P., Cupak, M., Benedix, G.K., Cox, M.A., Jansen-Sturgeon, T. and Stuart, D., 2016, August. Fall and recovery of the murrili meteorite, and an update on the desert fireball network. In *79th Annual Meeting of the Meteoritical Society* (Vol. 79, No. 1921, p. 6265).

Sansom, E.K., Bland, P.A., Towner, M.C., Devillepoix, H.A., Cupák, M., Howie, R.M., Jansen-Sturgeon, T., Cox, M.A., Hartig, B.A., Paxman, J.P. and Benedix, G., 2020. Murrili meteorite's fall and recovery from Kati Thanda. *Meteoritics & Planetary Science*, 55(9), pp.2157-2168.

Table 4. Attributions for Chapter VI (The Silicate-Sulfuric Acid Process - Acta Astronautica)

Co Author	Conception and Design	Acquisition of Data and Method	Data Conditioning Manipulation	Analysis and Statistical Method	Interpretation and Discussion	Total Contribution
Eleanor Sansom	✓				✓	7 %
I acknowledge that these represent my contribution to the above research output and I have approved the final version. Signed: 						
Patrick Shober	✓					3 %
I acknowledge that these represent my contribution to the above research output and I have approved the final version. Signed: 						
Ben Hartig	✓					2 %
I acknowledge that these represent my contribution to the above research output and I have approved the final version. Signed: 						
Hadrien Devillepoix	✓					1 %
I acknowledge that these represent my contribution to the above research output and I have approved the final version. Signed:						

						
Martin Towner	✓					1 %
I acknowledge that these represent my contribution to the above research output and I have approved the final version. Signed: 						
Total Contribution from Co-Authors						14 %

Copyrights

All permissions to reuse published material in this thesis have been obtained from Wiley Publishing, Elsevier, and The American Astronomical Society.

There can be no success without failure, and failure is no success at all.
-Anonymous

Remember kids, the only difference between science and screwing around, is writing it down.
-Adam Savage

Chapter I: Introduction

Quite simply, meteorites are rocks that fall from the sky. The more than 60,000 recovered samples in the global meteorite collection represent many parent and source bodies including Mars, Luna, and an array of Asteroids. Organizations such as the Global Fireball Observatory (GFO; Devillepoix et al. 2020) are uniquely enabled to observe their atmospheric entry to Earth and triangulate the meteoroids' luminous, ablative flight using their overlapping camera stations to image the night sky. From these observations, planetary scientists can predict both where the resulting meteorite landed on the Earth's surface, as well as infer where they most-recently originated from in the solar system. Recovering these meteorites bequeaths a wealth of inter-planetary knowledge upon whosoever recovers and worships them analytically. Although networks such as the Desert Fireball Network (DFN; Bland et al. 2012) have been sufficiently automated to facilitate this process, one logistical bottleneck remained: meteorite recovery.

The main focus of this PhD project aimed to develop a novel methodology employing drones and machine learning to address this problem: recovering meteorites with greatly reduced human labor. Unfortunately, 18 months into the project the global pandemic COVID-19 reached Western Australia and severely disrupted my progress towards completing this goal. Unsure when life would return to a relatively normal state, I was forced to adjust my PhD project in the possible scenario that I would be unable to see the meteorite recovery system to fruition, due to travel and work restrictions. This adjustment added the analysis and in-depth characterization of the meteorites that were previously recovered by the DFN, to my PhD project.

While writing the manuscript for the Murrili meteorite, I began thinking about how this detailed meteorite information could be used in the future. One personal research interest of mine, since I began working as a research assistant, was the utilization and extraction of resources in space, or In Situ Resource Utilization (ISRU). The continuity of my work, from collecting the meteorite from its natural landing place, progressing to geochemical analysis, and finally finishing with theorizing about future uses of meteoritic ores in space, warrants the title of this thesis: "Farm to Table Meteorites: An End to End Exploration of the Solar System's Past, Present and Future"

These three projects, although related in subject material to form a cohesive story from meteorite fall, to find, to characterization, to future use, forced me to grow and develop a wide set of tools and skills to overcome such a great methodologically diverse set of problems. The next Chapter (II) details these methodologies and the significance of each of the four peer-reviewed manuscripts I published for my PhD thesis.

References

- Devillepoix, H.A.R., Cupák, M., Bland, P.A., Sansom, E.K., Towner, M.C., Howie, R.M., Hartig, B.A.D., Jansen-Sturgeon, T., Shober, P.M., Anderson, S.L. and Benedix, G.K., 2020. A global fireball observatory. *Planetary and Space Science*, 191, p.105036.
- Bland, P.A., Spurný, P., Bevan, A.W.R., Howard, K.T., Towner, M.C., Benedix, G.K., Greenwood, R.C., Shrubbený, L., Franchi, I.A., Deacon, G. and Borovička, J., 2012. The Australian Desert Fireball Network: A new era for planetary science. *Australian Journal of Earth Sciences*, 59(2), pp.177-187.

Chapter II: Significance and Methodology

Abstract

This chapter first outlines the significance of the main body Chapters: III, IV, V, & VI, and fits them into the context of advancing planetary science, from considerably expediting the meteorite recovery process, to adding a well-defined data point in the growing collection of orbital meteorites, to possible future uses of these meteorites in their native space environments. Although each of the following chapters contains a methodology section, the current chapter includes the methodological deleted scenes reel that I used to accomplish my PhD thesis.

Significance

Meteorite Searching with Drones and Machine Learning (MAPS and ApJL)

Every night and day extraterrestrial material falls into the Earth's upper atmosphere. The largest and most robust breed of this material originates from asteroids, which most often orbit between Mars and Jupiter. The atmospheric entry of these meteoroids is so energetic that their outer sections vaporize and luminesce across sky. Those that appear brighter than Venus (in the night sky), are called fireballs. Of the fireballs that the DFN observes, about 5% are able to survive, in one or multiple pieces, the ablative, 'bright flight' atmospheric entry to eventually slow enough to terminal velocity before they land on the surface of the Earth, becoming a meteorite. The Desert Fireball Network (DFN; Bland et al., 2012; Howie et al., 2017) is one of many networks (Spurný et al., 2006; Trigo-Rodríguez et al., 2006; Colas et al., 2015) whose mission aims at observing and recovering these 'orbital meteorites'. We call them as such because their atmospheric entry was photographically recorded well enough to determine their pre-entry orbit, from which planetary scientists can infer their orbital evolution within the Solar system (example: Jenniskens et al., 2012). Inversely, the shooting star observations from fireball networks are also used to predict where a surviving meteorite may have landed, using numerical simulations and weather models.

Until now, the best way to recover these orbital meteorites was to assemble a group of people, show them what a meteorite looks like, and lead them to walk in a line 2~10 m apart, scanning the ground for the fallen meteorite until it is found. This is inefficient. From the institutional knowledge of the DFN, I estimate that it takes 6 people ~30 days to cover a fall line. Combining this with a rough estimate (based on the Vigilance Decrement in Human Factors Psychology, (See et al., 1995)) that a human has a ~50 % chance of recognizing the meteorite, I estimate that it takes an average of 360 labor-days to recover a fallen meteorite. This is somewhat supported by our success rate of 20 %, considering we have embarked on ~40 searching trips with a total of 8 recovered meteorites (averaging ~300 days per recovered meteorite).

The two papers that I published, first in Meteoritics and Planetary Science (MAPS, Anderson et al., 2020), and later in The Astrophysical Journal Letters (ApJL, Anderson et al., 2022), both detail the methodology and approach we used to recover a meteorite with drones and machine learning, which was accomplished using 12 labor-days (when considering a single, full scale survey trip alone). Our success in the second of these two papers has and

will enable planetary scientists to more quickly recover meteorites, allowing more time and effort to be focused on characterizing the population of near Earth, and Asteroid Belt objects, and how mass is transferred between solar system bodies.

The Murrili (H5) Meteorite (MAPS)

This line of scientific discovery clearly acts as a perfect segue into the subject of the next published paper which I have included in my thesis: the characterization of orbital meteorites recovered from the DFN. Although I was not present for the recovery, initial characterization, and sample allocation for the Murrili (H5) meteorite, the scope of my PhD was expanded to include the interpretation of the results from this meteorite's analytical studies (Anderson et al., 2021a), as well as begin and facilitate the analyses of the 4 new meteorites that *were* recovered during my PhD. As my statistical fate would have it, 7 of the 8 DFN meteorites (including Murrili) are classified as ordinary chondrites, which are so named because of their common nature, comprising ~80% of the meteorites in the global collection (Meier, 2022).

The main noteworthy anomaly that we observed in Murrili was the incongruent results from the shock state analyses performed via X-ray Computed Tomography (CT) and Optical Thin Section analysis. The preferential foliation of the metal grains within the meteorite indicate that Murrili likely experienced moderate to significant shock loading (overbearing pressure), while the behavior of transmitted light through the thin-section seems to suggest a more mild shock history. In this paper we also note that the CT data and the thin-section were sampled from separate regions of the meteorite. This may suggest that ordinary chondrites are less uniformly shocked on the demiscale (~10 cm) than previously thought.

Space Resource Utilization Using The Silicate-Sulfuric Acid Process

Given the above-mentioned advances in planetary science, I couldn't help but wonder: what can we do with all this information, about meteorites and where they come from? This led me to write the fourth paper and final main body chapter of this thesis: The Silicate Sulfuric Acid Process (SSAP; Anderson et al., 2021b). It outlines a theoretical process which incorporates some industry proven ore-processing techniques, to refine material on the Moon, Mars and asteroids. The main products of the SSAP are iron metal and silica grains, both of which would serve as ideal feedstock to create steel and spacecraft-quality windows on these heavenly bodies.

The significance of this paper lies in the exploration of non-volatile element resources in space. Whereas most ISRU studies focus on exploiting water or carbon dioxide ice to manufacture fuel on other planetary bodies, the creation of solid structures and products is often overlooked, even though their importance cannot be overstated. Even if the process that I proposed does not result in a feasible implementation, it will hopefully spur further exploration and consideration around harvesting resources in space for on-site construction.

Methodology

Meteorite Searching with Drones and Machine Learning

Drone Operations

I used two drones for this portion of the project, the first being the DJI Mavic Pro for data collection early in the project, as well as follow up searching once a meteorite candidate had been identified while in the field. The second drone I used was the DJI M300 with a Zenmuse P1 camera (48 MP, 35 mm lens), for surveying, as well as training data collection later in the project. The high cost of the M300 system (25,000 AUD) made it necessary for me to prove the feasibility of this approach using the less expensive Mavic Pro (2,000 AUD), hence why I used it for early data collection, as no other option was available. Prior to purchasing the M300, we borrowed or rented the use of SPECTRE UAV's M600 and Sony $\alpha 7rIII$ (42 MP camera) when we needed to test large-scale surveying components of the project. Since we acquired the M300 though, we use it exclusively for data collection, to ensure minimal variation in quality between survey and training data.

For the Mavic Pro, I connected its remote controller via USB cable to our Ipad to get the largest possible view from the camera and flew it manually with the DJI GS Pro application when collecting images. The M300 has its own remote controller that doubles as the smart device to allow for manual control, complex flight routes, computer-planned flights, etc., and additionally has an HDMI output to mirror the touch screen display to a separate monitor. Training data collection (which is explained in more detail in the next section) consisted of flying the Mavic at a height between 5 and 8 m (12-18 m for the M300), while another person placed a 'meteorite' on the ground, stepped back ~1-2 m and pointed at it for ~5 s. I would then take an image with the drone, and we would repeat the process often using different 'meteorites', placed in different backgrounds local to within 30 cm of the 'meteorite'.

Later in the process, once meteorite candidates proceeded to the point of warranting a second drone visit, with the Mavic Pro, I typed each candidate's GPS coordinate into GS Pro's (Ipad) waypoint flight mode, then activated the flight. These waypoint flights usually consisted of 10-20 candidates per flight, maintaining an altitude of 20 m. When the drone approached a waypoint, I paused the automated flight and took manual control, and guided the drone to the candidate while I viewed the original image on the computer as well as the live feed. Once the drone was ~1 m from the ground with the candidate in frame, I took at least 3 pictures as well as video (for later viewing since the live feed is lower resolution than saved data).

Planning the survey flights for the M300 mostly occurred on my desktop computer, prior to the trip. It started with a map provided by the DFN team which showed the bounds of the 90% probability area, from their Monte Carlo simulations and darkflight propagation (Towner et al., 2022). From the .kml file they provided, I looked for any large elevation variations and areas within, using Google Earth. Using its polygon tool, I created 4 or more slightly overlapping polygons such that each polygon maintained an elevation variation of ± 3 m. I saved each of these polygons as separate .kml files onto an SD card which was inserted into the M300's smart controller. Using the controller, I could save and plan flights by importing the saved .kml files. All of this was done prior to a trip.

To power these drones, the field trips required a 2 kW gasoline generator, set to 'eco' mode to conserve fuel automatically when the current draw was not at maximum. The only exception to this occurs when the machine

learning model is training, in which case we set it to full throttle, due to previously encountered black outs on the computer. The M300 came equipped with a charging case capable of housing all of our 8 batteries (4 flight sets), charging only 2 of the 8 batteries (1 set) at once. This mostly allowed for continuous survey flights until ~5 hours into the day, at which time a 30-45 minute break is usually needed to allow the charging rate to catch up. During midday and afternoon, when the temperatures of the batteries could reach ~45 °C by the end of a flight, we placed the empty, hot batteries into a camp refrigerator (set to its coldest setting; powered by either the generator or the dual battery-solar panel system installed on our vehicle) before recharging. The smart controller for the M300 also had a removable battery which was recharged by the charging station, separate from the flight batteries. The batteries for the Mavic Pro had 2 chargers that could house a total of 5 batteries (5 flight sets) and charge 2 batteries at once. These did not encounter the same high temperatures as the M300, and we had enough batteries (6 total) to fly the Mavic Pro non-stop, though the Ipad required recharging twice per day.

Machine Learning - Training

Using the Drone Operations and methodology described above, we collected training data. This consisted of two parts: meteorite (True; 1) and non-meteorite (False; 0) images. We collected non-meteorite images by flying the drone parallel to the fall line (~200 m offset), easily resulting in as many as 500 images, each of which contain ~9000 tiles, often totaling more than 1,000,000 tiles. The meteorite images were a little more tricky. I originally tried procedurally editing meteorites onto native backgrounds, but these models never achieved a high validation accuracy when applied to images of real black rocks and meteorites on the ground. Next we would gather rocks onsite that roughly looked ‘meteoritic’, meaning no particular extremely elongated axis, and roughly rounded on its edges, but not necessarily rounded as a whole. We painted these blacks with black spray paint, using both matte and glossy flavors. We used these ‘synthetic meteorites’ as stand-ins for real meteorites, as early tests showed us they were indistinguishable from real meteorites at our resolution (1.8 mm/pixel). We placed the rocks in a line ~3-10 m apart (20-40 at a time), ensuring the ~30 cm radius background in which each rock was placed is notably different than the last. This ensured good variety for the model, since the background could easily change between: desert pavement, bush, grass, dead tree, different kinda bush, flowering bush, other flowering bush, red dirt, brown dirt, desiccation cracks, white rocks, beige rocks, grey rocks, roo poo, etc.

While the pilot flew the drone, another person would walk along the line ~2-3 m to one side and point to the nearest rock as they walked past it. The pilot would follow along with the drone taking 1 picture per rock pointed to. We used these fake meteorites early in the process to show the feasibility of the concept, which is proven and discussed in Anderson et al. (2020). Once we were able to positively identify, with the algorithm, the real meteorites Madura Cave (L6) and Mundrabilla Fault (H5), we included real meteorites when collecting images. For the field trip included in Anderson et al. (2022), we used the real meteorites: Mulga North (H6), Wiluna (H5), and Camel Donga (Eucrite), all supplied by Peter Downes and Geoff Deacon at the Western Australian Museum. These totaled to 28 stones ranging in size from 2 to 14 cm (when measured on their longest axis). Since I would be summarily executed if any of these samples were lost, one person would place one meteorite on the ground and point to it, while the drone pilot took an image, then repeated the process with the same rock 5-10 times before returning the rock to its sample case, and selecting a new sample. Ensuring that each emplacement was in a different spot and orientation with respect to the last. Each method of data collection, whether it be with real or synthetic meteorites, would usually yield ~100,000 tiles with ‘meteorites’ for training. Next, we labeled (in the machine learning sense) our true images using ImageJ on our field computer. The fastest way to do this labelling was to load ~14 images into ImageJ, convert them into a stack (so that I could scroll through those selected with

the mouse wheel), and use the rectangle tool to identify and record the bounds of the meteorite. This was simply done by setting the results option to display X and Y bounds as well as the image name, and press the 'm' key whenever I drew a satisfying rectangle around the meteorite, finally exporting to a .csv or .txt file. I would repeat this in ImageJ until all had been labeled. This process is outlined in Figure 1 in Anderson et al. (2020).

The next phase is a semi-automated pipeline of sorts (see the Appendix for full code). In python we used the Pillow and numpy modules to generate 125x125 pixel tiles from both the True and False images. The False images were fully split into tiles with a stride set to 125 randomly rotated and changed the brightness, for the M300 this totalled 9126 per image. Using the True images, with the location information for the meteorites in each frame, I generated tiles by striding over the image (10 pixels), keeping the meteorite fully in frame each time, then rotating each of those strides 90°, ×3 iterations and randomly adjusting the brightness before saving. For the Kybo-Lintos (DFN 09; internal label, not official name) trip, this totaled (~50,000) true tiles and (>1,000,000) false tiles. This imbalance presented itself as the next problem: keeping the data set balanced during training.

To do this, I employed what I call 'revolver training'. Essentially it works by generating all of our training data which usually amounted to 1,000,000 - 4,000,000 False tiles and ~50,000 True tiles (taken only from local data). We then added 50,000 more True tiles generated from data collected on other trips, bringing the total number of True tiles to ~100,000. We then split 20% (~20,000) of the True tiles into a separate set for validation, which the model did not use for training. The model did however predict on this validation set during training, to provide feedback on how well the model was training. The remaining 80,000 True tiles formed the True half of the training set, which did not change for the course of the training. The False half was formed many times throughout the course of a training session.

Before I continue, I will explain some of the terminology. A training session is broken into epochs, each of which can be thought of as a single round of training, in which the model trains on all of the target data, once. In an epoch, the model will use all of the data in a training set to adjust its own weights, ideally improving the model's accuracy. If, like in our case, the training set is too large to be loaded into the GPU's local memory (12 GB), the dataset can be broken into batches which are fed to the model. Once the model has seen the whole target dataset by training on each batch, the epoch is complete. At the end of an epoch, the model, with its newly updated weights, will report its predictions on both the training and validation set, producing training and validation accuracy. In our case an epoch typically lasted ~30-90 seconds.

For revolver training, we used all the True training tiles, described above, as well as some of the False tiles at one time. Every 10 epochs I would automatically pause the training, move the 80,000 False tiles being used in training, out of the training set directory, and back to the False pool. The program I wrote would then randomly select 80,000 False tiles from the pool, and move these to the training set directory, forming the new training set, upon which the model trained for 10 epochs, before repeating the process again. This procedure would continue until the model had seen all of the False tiles twice. This approach to training would allow us to keep a balanced 'training set' during training epochs, while also enabling the model to see all of our False tiles. If you've been paying attention, you may have noticed a lack of discussion about the False validation set, because there wasn't one. Since the validation set in no way affects training, I only included True tiles within it in order to monitor the meteorite detection chance of the model. For the model I trained to use at the Kybo-Lintos fall, the model achieved a 99.93% training accuracy and a 91.05% validation accuracy. Training took ~2 hours on our field computer.

Machine Learning - Prediction

Once the revolver training had produced a high-accuracy model, we began predicting on our survey data using said model. This was automated such that the computer could process one flight's worth of images, with no human input required. One survey flight (30 min) worth of images usually took 60-90 min to process with the algorithm. The function that predicts over the images begins by looking for previously saved logs within the same directory, which can indicate if the function completed its predictions over all of the images for a flight. This is an important safety feature in the non-rare event that the computer loses power midway through predicting on a flight of images. This log checking step allows the model to pick up where it left off, with minimal data loss or recomputation. The function is mostly optimized though could be improved slightly. It works by opening each image (CPU), splitting into tiles (CPU), and feeding them to the GPU for prediction. Once the GPU has completed its prediction, the results are handed to a multiprocessing thread on the CPU to save the results, so that the next image can begin being processed at the same time. In the past we would pre-enter a prediction threshold, so that the function could highlight the over-threshold (interesting) tiles, though later in the process, when we became better at limiting false positives, this became cumbersome. Instead, now we simply save the prediction values of each tile for all images, x-y bounds, and original image name, so that later we can parse the results and select a threshold that better suits the distribution. I also wrote an option to still pre-enter the prediction value. When the results of each image were saved into individual .csv files, a histogram of the predictions for said image was also saved. This allowed me to quickly quantify how the model was performing for a flight, I often paused the prediction function to retrain.

Detection Sorting - False Positives vs Meteorite Candidates

The first step in sorting false positives from meteorite candidates began with reviewing the histograms produced by the prediction function. Figure 2 in Anderson et al. (2022) shows the difference between a good and bad distribution. Images that produced histograms like B were flagged for retraining, in which the entire image was used to make tiles for a new retraining set. If the rate of detections was too much (greater than 2 per image with a 70% confidence threshold), we would stop the prediction currently working on a flight of images and retrain it. In other instances when the detection was more tolerable, we just did the retraining after the main prediction and re-processed the few problematic images. From this point, the detections were sorted through a 4 stage elimination process: grid GUI (Stage I), zoom-pan GUI (Stage II), drone follow up (Stage III), in-person follow up (Stage IV)

Now that we had selected the detections for humans to view, we passed these interesting tiles into the first GUI that I made (Stage I). This grid-sorting GUI worked by displaying 9 tiles in a 3x3 grid to the user. The user would identify which of the tiles resembled a meteorite by pressing the corresponding number on the keypad (Figure 1). Once the meteorite-like tiles had been identified, the user pressed the 'Enter' button on the keypad to advance to the next set. The '-' button on the keypad would take the user to the previous set, while 'Backspace' eliminated the selections for the current set, in case of a mistake by the user. The program would also have a uniform probability of displaying 0, 1 or 2 tiles from the training data set, within the grid of 9. These acted as tests for the user, with the passed-test rate displayed at the top of the window. Although this was a good metric for understanding each user's performance, it more importantly forced the user to slow down and seriously consider each of the 9 tiles, preventing them from 'mashing' the 'next' button which would likely cause them to overlook meteorite candidates. This Stage I was designed as a first pass at any detections from the model, to quickly differentiate between dark spots (for later inspection) from obvious false positives (for elimination (bushes, blank patches of sand/dirt, etc.)).



Figure 1. The grid sorting GUI, that allows users a first pass at the model detections. The typical dimensions for the field of view (for each detection) was 67 x 67 cm.

Once all of the detections for a flight had been inspected via the grid GUI, we inspected the surviving candidates in the zoom-pan GUI (Stage II). This GUI is mostly comprised of code from user: <https://stackoverflow.com/users/7550928/foobar167>. Our modifications allow for easier selection of the images to be viewed, while also enabling us to quickly move to the next image in the selected directory. This GUI, like its name implies, displays an entire 48 MP image from the drone with the meteorite candidate tile outlined in a yellow box, allowing the user to move to any part of the image and zoom in and out. This step was important, because it allowed for up close inspection of each candidate individually to scrutinize its features, which often led to false positive elimination, since the grid GUI was only designed for the user to differentiate dark spots from obvious false positives. The Stage II GUI also allowed the user to inspect nearby features, which help us to eliminate more nuanced false positives such as animal droppings (Figure 2)

After each candidate had either passed through or been eliminated at Stage II, we compiled the surviving candidates manually into the DJI GS Pro app on the iPad, to plan follow up flights (Stage III) for the Mavic Pro. This is one bottleneck that should be alleviated in the future. Ideally, we would sort the candidates to minimize total travel distance, save to a .kml file, then export to a flight planning app. After the flight has been planned we begin the mission with the Mavic Pro, making sure to consider the location and current actions of the M300 survey, to avoid collision. As an added safety measure, the M300 flew at a height of 15 m, while the Mavic Pro was programmed to fly at 30 m to further mitigate a collision risk. When the Mavic arrived at a candidate, we paused the flight and manually descended the drone until it hovered ~1 m over the candidate, using the original image as a guide. If we could eliminate this candidate during flight, we would. If we were still unsure however, we would take 2-3 images with the Mavic for later inspection, since the live-feed has a lower resolution than images saved to the drone, which could then be inspected post-follow up flight on our field computer.

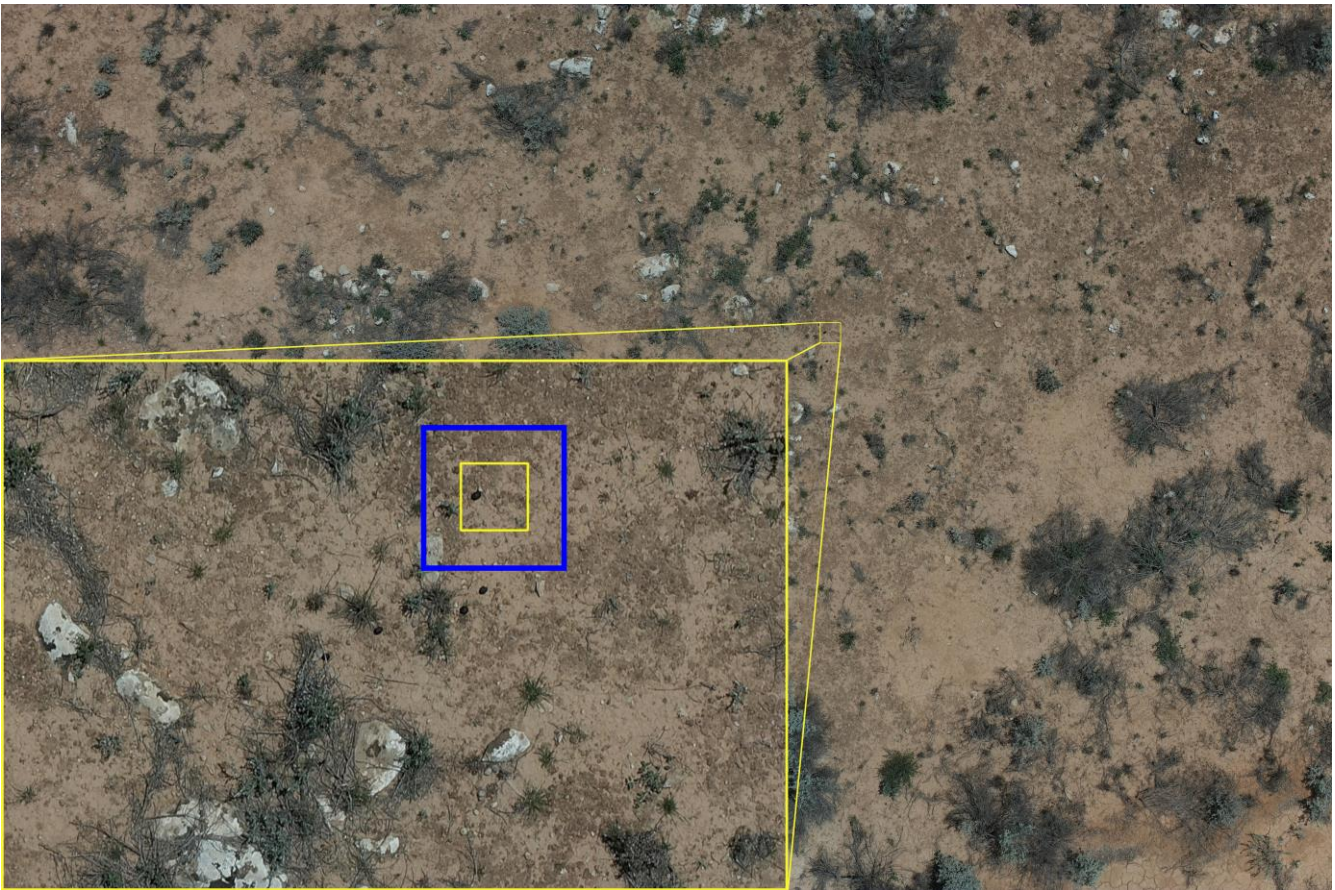


Figure 2. This figure shows the field of view and magnification of the zoom-pan gui as well as the limitations of the grid sorting gui. The background is a full image taken from the Zenmuse P1, the enlarged area is a typical view with the zoom-pan GUI (though not at maximum magnification), the limits of the grid gui's field of view is outlined in blue. When we originally sorted this detection, it seemed interesting enough to double check, though upon doing so, we realized it was likely 'roo-poo.

The final, Stage IV, of candidate elimination consisted of an in-person visit. We did this by loading the candidate coordinates into a GPS and walking out to its location, carrying the iPad with the original survey images of these candidates, as well as the Mavic drone in its case. Although we found each of the candidates fairly easily, taking

the smaller drone along with us ensured that we could find the candidate in case it eluded us. In the case of the Kybo-Lintos trip that yielded a recovered meteorite, we only visited 4 meteorite candidates in person, with each one taking ~30-45 min to walk to. Assuming that future trips may not be so lucky with an early detection, an electric mountain bike may prove to be invaluable for quickly moving from one candidate to another.



Figure 3. Images of the recovery of DFN 09, the first orbital meteorite to be recovered with a drone. (Clockwise from top left): The fusion crust of the meteorite, note the terrestrial alteration near the bottom; The scale of the whole rock (with a 14 cm felt pen for scale); The recovery team standing over the rock; Field setup, with computers and charging equipment in the tray of the 4WD vehicle; Image from the closeup of the meteorite, taken with the Mavic pro; The original image in which the meteorite appeared, with annotated yellow box showing the model detection.

On the Kybo-Lintos trip, we found the meteorite which appeared as the 3rd candidate we visited in person (Figure 3). The previous candidates we visited were remotely convincing but not overwhelmingly so. So when we first saw the would-be meteorite in the 2nd stage, and later the 3rd stage, I personally thought that it was the most convincing candidate we had seen so far, indistinguishable from the training data I was accustomed to labeling. Although we initially picked up the rock to inspect it, we immediately returned it to its original place. We then took dozens of pictures of the meteorite with a DSLR camera, and with the Mavic pro at various altitudes from 1 to 30 m. Once we were satisfied with the pictures taken, we used metal tongs to place the meteorite into a teflon bag which was placed into a cloth bag for contamination and physical protection, respectively. We also used a

nearby rock (being sure to use one of its untouched surfaces) to gently gather groundsoil directly adjacent to (<10 cm) the meteorite, which would serve as a background or control measurement in the event the meteorite were a carbonaceous chondrite and analyzed for organic content.

Meteorite Characterization

Once the meteorite has been recovered one question (though really many questions) remains: what do we do with them? As I said in Chapter I, because of Covid-19, the scope of my PhD project was expanded to include the analysis of the meteorites we recovered. Although much of the curation and sample processing for the two meteorites included in my thesis (Dingle Dell (L/LL5) and Murrili (H5)) was done prior to my arrival, the characterization of these two rocks was completed by me. Since Murrili fell and was recovered before Dingle Dell, it was the primary focus of analysis, and hence was fully published and included in my thesis. The characterization of Dingle Dell, however, was completed but still in the review stage, and therefore not eligible to be included in my thesis. My work did however also include the sample processing and classification of new meteorites that were recovered since I arrived (Arpu Kuilpu (H5), Madura Cave (L5)).

Much of the work I did interpreting the results for Murrili (H5) and Dingle Dell (L/LL5) does not make for an exciting methodological explanation, since most of it included simply reading published literature for other similar orbital meteorite characterizations, and comparing our results to theirs. I will however overview the new tool I developed to help expedite this process, as well as the sample handling pipeline. The first step when we received a new meteorite, began by imaging (with a digital camera) the exterior of the meteorite to document its fusion crust (completed by Wesley Lamont (Curtin HIVE), and Gregor MacGregor (Curtin School of Design and the Built Environment)). This would produce a 3D, exterior model of the meteorite (Madura Cave (L5)).

Next, our collaborator, Belinda Godel at CSIRO in Kensington, WA, would map the interior of the meteorite using X-ray Computed Tomography (CT). This data product would inform us of the general type of the meteorite, as we could differentiate between high (metal), medium (chromite, troilite), and low density (silicate) areas in the meteorite. If the whole meteorite was small enough (~5 cm on its longest dimension) we could make use of the micro-CT scanner, yielding a voxel (3D pixel) size of ~15 microns³ / voxel. If the meteorite was larger than this however, we would use the medical CT (~200 microns³ / voxel, at CSIRO) for the main mass, and send a ~50 g consolidated mass (separated from the main mass) to our collaborator, Jon Friedrich at Fordham University and The American Museum of Natural History (AMNH), for fine resolution CT scanning (~10 microns³ / voxel). Fine resolution CT scans, either of the main mass or a smaller piece would allow Jon Friedrich to calculate the porosity, bulk and grain density (given the mass), as well as the metal grain foliation of the sample (if it were an ordinary chondrite), which could help constrain its shock history.

After the initial CT scan at CSIRO, I broke the meteorite open/into small chips in one of two ways, either using a hammer and a Cr-V chisel, or a hand-turned wedge press. We avoided using a saw to prevent contamination from both the blade and the lubricant (usually water) which would contact all of the cut surface and likely soak further into the rest of the meteorite. Breaking the meteorite, although a less precise approach, benefited from only having a single point of contact between the tool and the sample, greatly limiting the amount of contamination. We usually tried to remove a ~50 g wedge which could, though not in a preferential way, include fusion crust used for fine resolution CT imaging. Removing such a wedge usually created ~10 small chips (0.5~2 g each) of the meteorite.

I would embed 2 chips of the meteorite into 2 thick-section epoxy mounts, which I then polished and coated with aerosolized carbon. These thick-sections were then imaged for us using the Scanning Electron Microscope (SEM) at the John de Laeter Centre (JdLC) at $3 \mu\text{m}^2 / \text{pixel}$, giving us a relative element map (values reported between 0-255, not calibrated, absolute counts). Analyzing these results was really annoying and tedious, consisting of manually importing the relevant element maps (Fe, Cr, S, Mg, Al, and Si; Ni map shown in Figure 4) into GIMP (GNU Image Manipulation Program), converting each of the binary images into RGB, overlaying them sequentially, adjusting some of the colors, then using the color select tool to count the number of pixels based on each color/mineral represented (7 total). This was repeated for a total of three times, for each thick section. I was able to automate this process, such that one keystroke would do all of this.

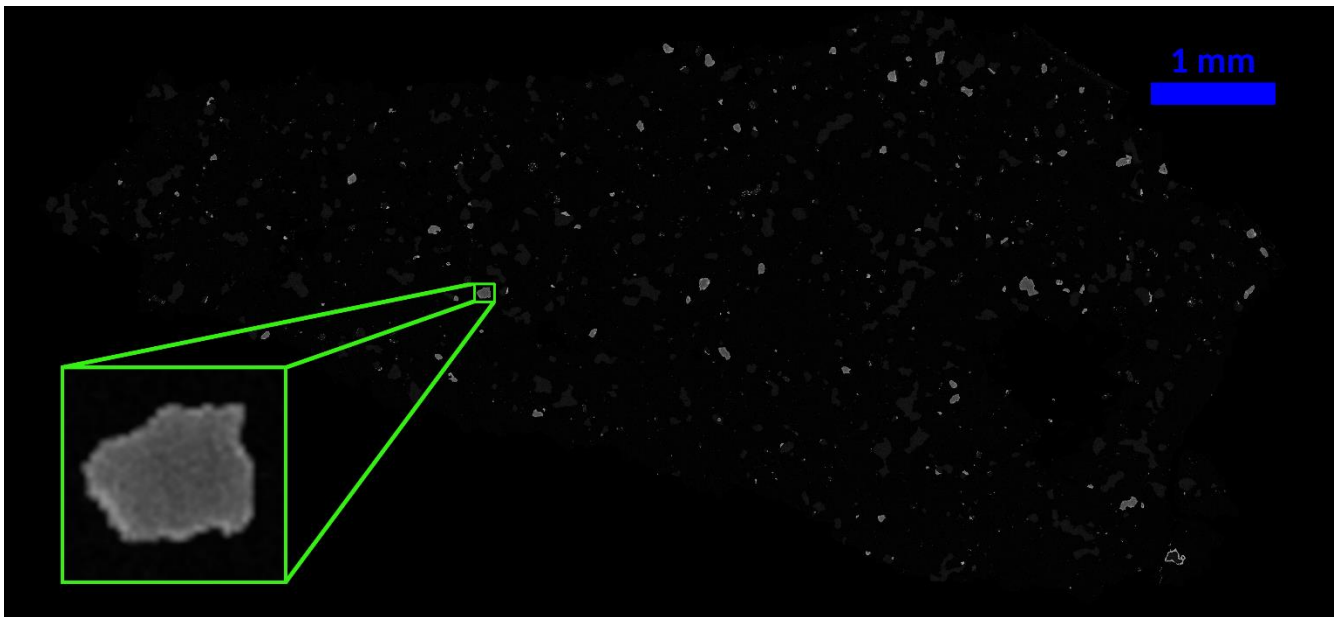


Figure 4. An SEM map displaying the Ni concentrations throughout the sample ($3 \mu\text{m}/\text{pixel}$). Faint Ni signal indicates Kamacite (5-10 wt% Ni), while brighter pixels indicate Taenite (~ 40 wt% Ni; enlarged subsection).

Using python (see Appendix for code), I imported the element maps and overlaid the S, Fe, and Cr maps together, with each color channel representing a different element (Red-S, Green-Iron, Blue-Cr); we called this the opaque map, since the minerals identifiable here (Figure 5) are opaque (Teanite-Kamacite, Troilite, Chromite). In the opaque map, the python script also makes adjustments based on the Cr and Fe content. In pixels where the Cr channel is higher than 200, the color of the pixel is changed to pink (230, 30, 145) to more easily identify the chromite. For Fe, if the value is below 200 and S is below 20, the Fe channel (Green) is changed to zero, for each pixel. This first threshold makes the olivine and pyroxene more consistent, and allows troilite to appear as orange (red + green). If Fe is higher than 200, the color of the whole pixel is changed to grey (100, 100, 100), to represent Fe-Ni metal. Separate from the opaque map, the script also overlays the Ca, Si and Mg images to form the silicate map (Figure 5), in which no colors or channels are adjusted. Adding the silicate and opaque map together yields the combined mineral map, in which the major minerals are easily distinguishable (Figure 6).

To train the neural network in the script, I loaded the mineral map into Image J and used the rectangle tool to select areas within the image that contained one mineral. The pixels within these areas formed the training set on which

the model was trained, effectively being loaded as (3x1 array ‘images’), into 1 of 8 categories (Chromite, high-Ca Pyroxene, low-Ca Pyroxene, Olivine, Plagioclase, Troilite, Phosphate minerals, and Fe-Ni metal). The model architecture is fairly simple and small, consisting of x6 Dense layers (10 neurons each, ReLu activation function) followed by a final Dense layer with 8 neurons and a softmax activation function. The model then predicts on each pixel and returns a 8x1 array containing the confidence values (between 0 and 1), or the likelihood, that the pixel in question belongs to each of the 8 categories (minerals) that I defined. Usually this worked out such that the highest value in this returned array was >0.9, while the other values were below 0.001, which made mineral determination easy. Unfortunately, some pixels return confidence values with no clearly defined answer, where the maximum value is ~0.3, with a median and average near 0.2. For these hard-to-determine cases, I suspect that the SEM resolution was not high enough to prevent two minerals from occasionally appearing in the same pixel, causing an unusual, in-between color. These pixels are instead counted as unknowns, usually accounting for ~1-2% of the modal mineralogy for a thick section.

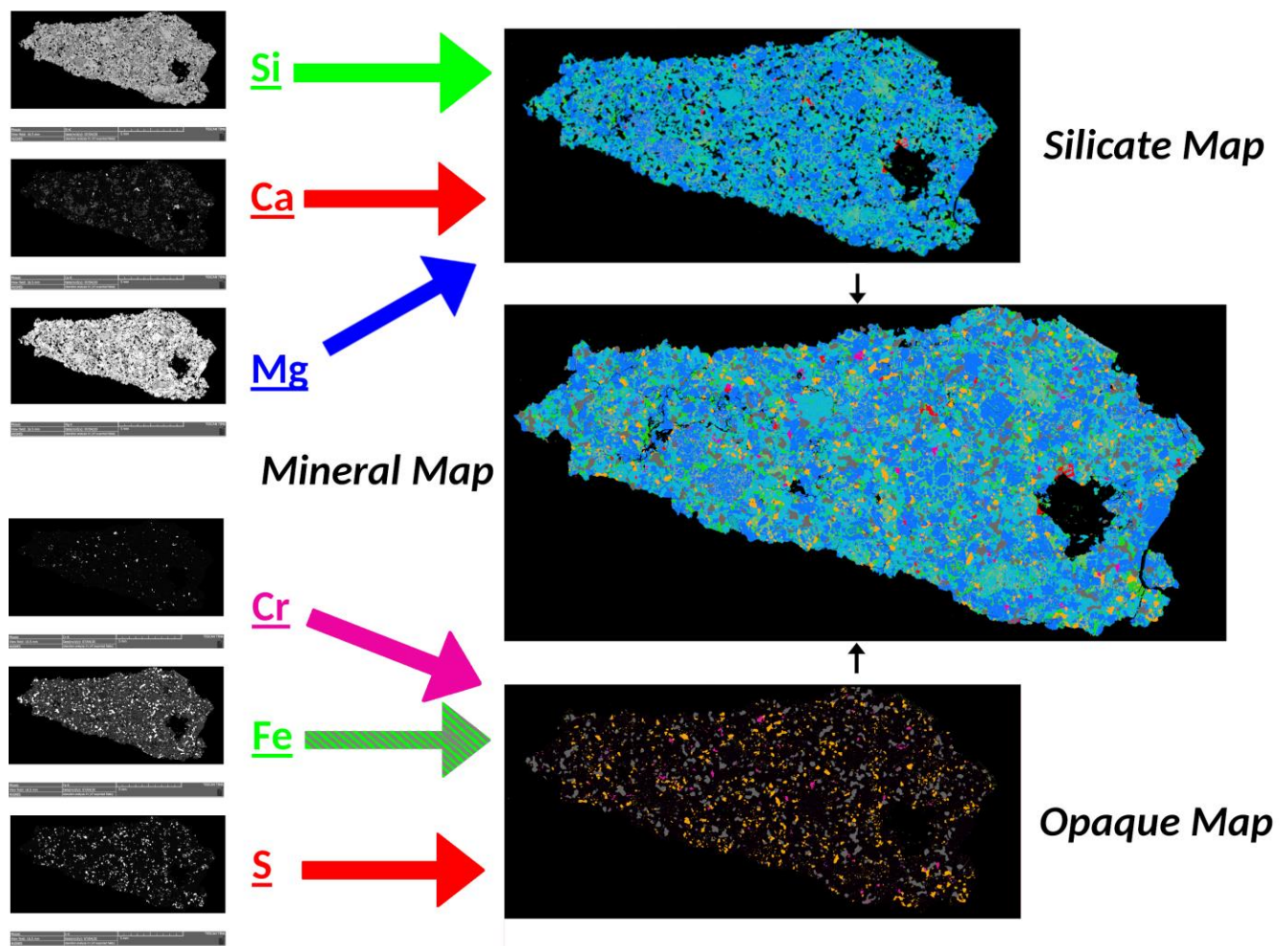
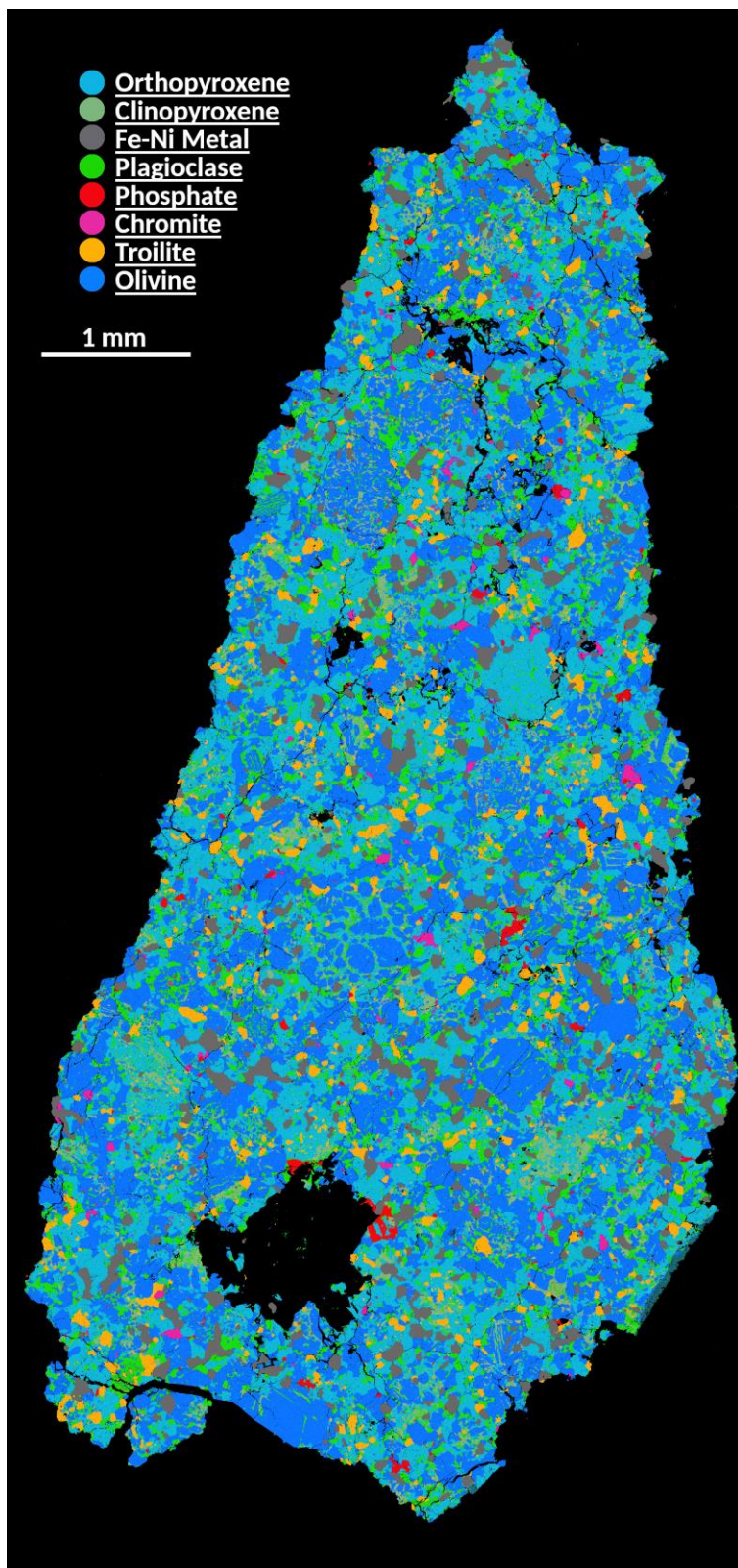


Figure 5. A flow chart illustrating how the automated python script constructs the various mineral maps from SEM-generated element maps. The S, Cr, Si, Ca, and Mg maps are converted to the colors shown, without any alteration to the values within the image. The Fe map is thresholded to Green, Grey, or zero depending on the value to more clearly represent Fe-Ni metal, FeS, and silicates.



I tested this script by performing both the manual and automated analyses for the new (at the time) meteorites Arpu Kuilpu (H5) and Madura Cave (L5), and comparing the results. The automated modal mineralogical analysis fell within the error established by using the manual method, validating this approach. After I determined the modal mineralogy using the SEM and the python tool, I took the thick-sections to the Centre for Microscopy Characterization and Analysis (CMCA) at the University of Western Australia, and with the help of Malcom Roberts, determined mineral chemistry via Electron Probe Microanalysis (EPMA). The best way we accomplished this was for me to navigate around the sample using the joystick, knob, and buttons controls of the EPMA machine while looking at the SEM mineral map on my nearby laptop. This allowed me to more clearly identify which mineral my selection point was focused on, since the EPMA only showed a greyscale Backscatter Electron (BSE) image. Although I selected all of the xy-coordinates to measure on each thin section, Malcolm Roberts calibrated the raw data to measured standards. I used the calibrated data to calculate mineral chemistry, and thus petrologic type, by comparing the average mol% of Ca in low-Ca pyroxene to published values (Scott et al., 1986). By this point in the meteorite analysis, I had sufficient information to officially classify the meteorite with the Meteoritical Bulletin.

Once we knew what type of meteorite we had on our hands, we could allocate and send samples to the rest of our collaborators, since some analyses are unnecessary or interpretations altered, depending on the meteorite type. I should also mention that

Figure 6. The final mineral map of Arpu Kuilpu (H6) (DFN 06). Although some altered chondrules can be identified, the texture is indicative of an equilibrated ordinary chondrite.

besides packaging and shipping the samples, I took no part in the lab work to produce and process the raw data for each measurement, besides those listed above (SEM/EPMA). Most of our collaborators ask for <500 mg of fusion-free meteorite to make their destructive measurements, so as an added margin we typically send between 0.5-1 g chips to each collaborating institution, all of which are listed in Table 1. The exception to this is the ~50 g secondary mass (mentioned earlier). This chunk of space rock was usually sent first to Matthias Laubenstein at the Italian National Institute for Nuclear Physics, who measured for and calculated the terrestrial age of the meteorite using short-lived radionuclides. The second analytical visit the secondary mass made was to Robert Macke at the Vatican Observatory who used laser-scanning and ideal gas pycnometry to measure bulk and grain density, and porosity.

Table 1. This table lists all of the collaborators, their institutions, the analyses they perform, and the sample type they require, with which I collaborated during the meteorite analysis portion of my thesis.

Collaborators	Institution	Analysis	Sample Type
Belinda Godel, Lionel Esteban	CSIRO - Kensington	CT	Whole rock or Consolidated Piece
Fred Jourdan, Celia Mayers	JdLC	Ar-Ar Chronology	Individual silicate grains
The TIMA Team	JdLC	SEM	Thick Section
Malcom Roberts	University of Western Australia	EPMA	Thick Section
Kees Welten, Marc Caffee	University of California – Berkeley & Purdue University	Cosmogenic Radionuclides	Total of 0.5 g (pieces)
Henner Busemann, Maden Colin, Matthias Meier	ETH Zurich	Noble Gases	Total of 0.5 g (pieces)
Robert Macke	Vatican Observatory	Helium Pycnometry	Main Mass, or consolidated piece above 50 g
Jon Friedrich	Fordham University	Bulk Chemistry	Total of 500 mg (pieces)
Jon Friedrich	American Museum of Natural History	CT	Main Mass, or consolidated piece above 50 g
Qing-Zhu Yin, Matthew Sanborn	University of California – Davis	Cr Isotopes	Total of 0.5 g (pieces)
Richard Greenwood, Ian Franchi	The Open University	O Isotopes	Total of 0.5 g (pieces)
Karen Ziegler	University of New Mexico	O Isotopes	Total of 0.5 g (pieces)
Jason Dworkin, Hannah McLain	NASA – Goddard Space Flight Center	Organic and Amino Acid Content	Total of 0.5 g (pieces), Adjacent soil sample
Matthias Laubenstein	The Italian National Institute for Nuclear Physics	Short Lived Radionuclides	Main Mass, or consolidated piece above 50 g
Anthony Jull	University of Arizona	Terrestrial Age	Total of 0.5 g (pieces)

The final visit of the secondary mass was to Jon Friedrich at Fordham University/AMNH, who took a CT scan of the meteorite at fine resolution, if the whole meteorite was too large for the micro-CT machine at CSIRO.

Some analyses are able to be performed rather quickly, while others can take 6 months or more, but once they had each been completed for a particular meteorite I would start to assemble the manuscript. Since there were so many methods performed upon each unsuspecting piece of space debris, I condensed each methodological description for the main text of the manuscript, but spared no detail in the supplementary materials to sate the hardcore meteoriticists. I then read an array of published materials on geochemical properties of other ordinary chondrites, as well as literature regarding the dynamical evolution of meteoroids and collisional families within the asteroid belt. From these texts I made interpretations for the genetic, or geochemical origins of the meteorite at hand, and how its originating body dynamically evolved over time to deliver such a rock to Earth. The entire time I was characterizing these meteorites, I wondered: what can the meteorites be used for? What resources could they possess?

The Silicate Sulfuric Acid Process

My introduction to space resources began when I first met my later-to-be first research supervisor, Dr Phil Metzger. In our first conversation he told me about his most recent research project using heat to extract water from select meteorites and simulated lunar soil. He then proceeded to talk about permanently shadowed crater regions on the poles, but especially the southern pole of the Moon, and how they harbored water ice. At first, I was skeptical, so I looked it up later and it was totally true. Ever since, I feel as though nearly all of my professional efforts should be made toward using space resources to improve the human experience and the preservation of Earth.

My first thoughts were rather crude: what temperatures were required to liberate various gasses from asteroidal, lunar or martian regolith? This was the perfect query for investigation via Thermogravimetric Analysis (TGA). A TGA experiment holds a powdered sample under vacuum, suspended from a weight balance to measure the mass of the sample as the apparatus is heated to high (>1000 °C) temperatures. Electively, a gas characterizing instrument (mass or infrared spectrometer) can be connected in-line between the experiment chamber and the vacuum pump, to better record the sample's gas release behavior as a function of temperature. I should also note that TGA and evolved gas analysis was the main focus of my summer internship at NASA Goddard Space Flight Center. During my literature review, I noticed that these experiments reported high (>1600 °C) melting temperatures for the major minerals of possible space ores. So, I started thinking of possible options to reduce the temperature required for thermal decomposition, causing gas release.

Like much of the work for this paper, I delved deep into the literature which reported using meteorites as the target for TGA and noticed that while most of these studies (Garenne et al., 2014; King et al., 2015; Gilmour et al., 2019) were concerned with the volatilization behavior of water, they also report the release of sulfur bearing gasses near 950 °C. I then began to think of possible uses for sulfur, inevitably leading me to sulfuric acid, which has shown an ability to dissolve silicate minerals (Van Herk et al., 1989; King et al., 2011; Lazaro et al., 2012) the major mineral group of the moon, mars and many asteroids (Heiken et al., 1991; Bland et al., 2004; Dunn et al., 2010; McSween et al., 2010; Nakamura et al., 2011). By using my basic understanding of chemistry accompanied by an expansive literature review (see Reference list in Chapter VI (Anderson et al., 2021b)), I theorized that the silicate minerals could be dissolved in an acid solution, then dried into metal sulfate compounds, which could be further heated until decomposing into metal oxides (at <1000 °C). By careful prior separation, it is also possible to isolate

silica particles from the solution (Lazaro et al., 2012), whose uses are discussed later. From the metal oxides, iron oxide can be reduced to iron metal with the proper application of carbon monoxide at a modest temperature (<800 °C), an already established industry process known as carbo-thermal reduction. These iron metal particles would serve as the perfect feedstock for laser-sintering 3D printing, an ideal process to be carried out on planetary surfaces and possibly micro-gravity. The silica particles, earlier separated from the ion-acid-aqueous solution, could be used to create fused silica, which is used to make spacecraft windows (Salem et al. 2013).

Again, most of the conceptual invention of this process originated from reading copious amounts of published papers related to individual, incorporated aspects of the process. During the first round of review, the reviewers suggested/insisted that I calculate the Gibbs Free Energy for each reaction to show that it was thermodynamically favorable. For this portion I used my knowledge of physics, chemistry, algebra, differential equations, and thermodynamics, as well as my proficiency in python (see Appendix for code) to create a collection of scripts to calculate the Gibbs Free Energy and progression of the reaction for various temperatures and pressures. This also required tabulated values (enthalpy (H) and entropy (S)) for each compound I evaluated, which I mostly acquired from the National Institute of Standards and Technology (NIST). Some compounds (minerals mostly but not most minerals) were not well characterized so I had to source the values from individual studies and assumed no thermal variation. This decision-assumption duo could be risky and cause miscalculation, but we had no other option as no other data was available.

Fortunately, most substances had data available through NIST, often beyond 1000 °C, though they were presented in a compact form (Figure 7), which I had to unpack with calculations. As shown in Figure 7, I saved the table of polynomial coefficients for the given temperature ranges, as a .csv file for each compound. I used these files and a python script to calculate H and S for each compound at each temperature, saving G (as a function of temperature) to a separate .csv file. From here, calculating the Gibbs Free energy for a reaction at a particular temperature was easy by simply following the derivation below. Though this was only accurate for a pressure of 1 atm, I wanted to show that some of these reactions, particularly the thermal decomposition steps, would benefit from lower pressures of medium to hard vacuum (10^4 - 10^8 atm). Using this derivation, I calculated the pressure and temperature-dependent Gibbs Free Energy for each reaction at 1 atm, 10^{-4} atm, and 10^{-8} atm, across temperatures from 20 °C to 1400 °C.

Once I had calculated the Gibbs Free energy for each of these temperature-pressure scenarios, used the relation to the Equilibrium Constant (K_{eq}) and its own definition to predict the progression of reaction, and therefore the concentrations of each reactant and product in each scenario. This last part about reactant/product concentration required some clever algebra and judicious use of Pascal's triangle, in conjunction with a python-based polynomial solver, to evaluate completely. By plotting the value of ΔG and K_{eq} for each constant pressure considered (1, 10^{-4} , and 10^{-8} atm), it was easy to see ideal pressure-temperature conditions to force a reaction to proceed.

The final numerical characterization I applied to the theoretical SSAP was an estimation of the products from a few different ores types. As stated in the paper, the main products of the SSAP are oxygen gas, Fe metal, and SiO₂ nano particulates, with other metal oxides (MgO, CaO, Al₂O₃) as secondary products. So I calculated the mass of each of these products produced if the SSAP was performed at 100% efficiency.

Condensed phase thermochemistry data

Go To: [Top](#), [References](#), [Notes](#)

Data compilation [copyright](#) by the U.S. Secretary of Commerce on behalf of the U.S.A. All rights reserved.

Quantity	Value	Units	Method	Reference	Comment
$\Delta_f H^\circ_{\text{solid}}$	-928.85	kJ/mol	Review	Chase, 1998	Data last reviewed in June, 1966
Quantity	Value	Units	Method	Reference	Comment
S°_{solid}	120.93	J/mol*K	Review	Chase, 1998	Data last reviewed in June, 1966

Solid Phase Heat Capacity (Shomate Equation)

$$C_p^\circ = A + B \cdot t + C \cdot t^2 + D \cdot t^3 + E/t^2$$

$$H^\circ - H^\circ_{298.15} = A \cdot t + B \cdot t^2/2 + C \cdot t^3/3 + D \cdot t^4/4 - E/t + F - H$$

$$S^\circ = A \cdot \ln(t) + B \cdot t + C \cdot t^2/2 + D \cdot t^3/3 - E/(2 \cdot t^2) + G$$

C_p = heat capacity (J/mol*K)

H° = standard enthalpy (kJ/mol)

S° = standard entropy (J/mol*K)

t = temperature (K) / 1000.

[View plot](#) Requires a JavaScript / HTML 5 canvas capable browser.

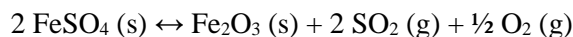
[View table.](#)

Temperature (K)	298. - 1200.	1200. - 2000.
A	32.55964	156.9548
B	305.4893	3.795713
C	-264.4100	3.459361
D	81.95661	-0.545610
E	-0.150805	-8.233778
F	-950.4793	-1000.792
G	79.43198	270.6282
H	-928.8480	-928.8480
Reference	Chase, 1998	Chase, 1998
Comment	Data last reviewed in June, 1966	Data last reviewed in June, 1966

Figure 7. The data page for condensed phase thermochemistry available on the NIST website. Although it does not explicitly list entropy or enthalpy, I calculate them using the polynomial (Shomate Equation) and coefficients provided. Note that this data assumes atmospheric pressure.

Gibbs Free Energy Derivation

For example, take the reaction below: the thermal decomposition of iron(II) sulfate into iron(III) oxide, oxygen, and sulfur dioxide.



We can define some terms and relation, such as the Gibbs Free Energy (G) for a substance (given tabulated values for Enthalpy (H) and Entropy (S), which are both temperature (T) dependent (dependence denoted by T subscript)). Note: $G_{2\text{FeSO}_4} = 2 G_{\text{FeSO}_4}$

$${}_T G_{2\text{FeSO}_4} = {}_T H_{2\text{FeSO}_4} - T {}_T S_{2\text{FeSO}_4}$$

To calculate the change in Gibbs Free Energy for a reaction, to determine if the reaction is spontaneous across a range of temperatures, we use:

$$\Delta G = G_{\text{products}} - G_{\text{reactants}}$$

Applied to our situation:

$${}_T\Delta G = ({}_T G_{\text{Fe}_2\text{O}_3} + {}_T G_{2\text{SO}_2} + {}_T G_{\frac{1}{2}\text{O}_2}) - {}_T G_{2\text{FeSO}_4}$$

But oh no! This is only good for predicting our reaction at atmospheric pressure, while our process calls for vacuum in some scenarios. Not to worry, here's some differential equations to the rescue. First we define Gibbs Free Energy and Enthalpy, combine the equations, then take the general derivative (where U is internal energy, P is pressure, V is volume):

$$G = H - TS$$

$$H = U + PV$$

$$G = U + PV - TS$$

$$\partial G = \partial U + (P \partial V + V \partial P) - (T \partial S + S \partial T)$$

Then we assume a quasi-static and reversible reaction (both of which are a bit of a stretch but the math is an estimation for reality anyways; Q is heat), such that:

$$\partial U = \partial Q - P \partial V$$

$$\partial Q = T \partial S$$

Substituting into the ∂G equation:

$$\partial G = T \partial S - P \partial V + (P \partial V + V \partial P) - (T \partial S + S \partial T)$$

Simplifying into:

$$\partial G = V \partial P - S \partial T$$

Now we assume that for this particular instance we're considering only one temperature and how the Gibbs Free Energy for the reaction changes when we keep T constant and vary P (don't worry we can recalculate with a different T later on but ∂P is what we're concerned about). This means $\partial T = 0$, and we can rewrite the above equation as:

$$\left(\frac{\partial G}{\partial P}\right)_{\partial T=0} = V$$

Note the $\partial T = 0$ subscript denotes the change in G with respect to P , at constant temperature. Now we can substitute/rearrange the ideal gas law ($PV = nRT$) into the above equation's right hand side, then integrate both sides with respect to P :

$$\int_{P_1}^{P_2} \frac{\partial G}{\partial P} \partial P = \int_{P_1}^{P_2} \frac{nRT}{P} \partial P$$

$$\int_{P_1}^{P_2} \partial G = \int_{P_1}^{P_2} \frac{nRT}{P} \partial P$$

$$G_{P_2} - G_{P_1} = nRT \ln\left|\frac{P_2}{P_1}\right|$$

We also know that P_1 is 1 atm, meaning that G_{P_1} is just ${}_T G$ at 1 atm, so we can simplify this even further to:

$${}_{TP}G = {}_T G + nRT \ln|P|$$

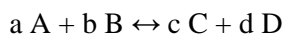
This equation above is the pressure dependent Gibbs Free Energy (below, note the $_{TP}$ subscript). For legal reasons I must state that n is the number of moles of gas, R is the ideal gas constant ($8.3145 \text{ J mol}^{-1} \text{ K}^{-1}$), P is pressure (atm), and \ln is the natural logarithm. This equation is applied to each substance that is in a gaseous phase. So for the decomposition of iron(II) sulfate into iron(III) oxide, sulfur dioxide, and oxygen, this equation would only be applied to the last two products, since they are in the gaseous phase.

But wait, there's more! Ever wanted to get serious about calculating the concentrations of the products and reactants? Wait no more! By using the following two wonder-equations we can calculate a single number, the equilibrium constant: K_{eq} , which is directly related to the [concentrations] of the reactants and products.

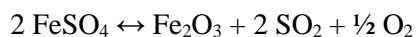
$${}_{TP}\Delta G = -RT \ln|K_{eq}|$$

$$K_{eq} = \frac{[C]^c [D]^d}{[A]^a [B]^b}$$

Given the general reaction:



So for our reaction:



We can write our concentrations in terms of each other, for fun let's write it in terms of FeSO₄. Take a moment to wrap your head around this system of equations, recalling conservation of mass.

$$1 - [\text{FeSO}_4] = [\text{SO}_2]$$

$$\frac{1}{2} (1 - [\text{FeSO}_4]) = [\text{Fe}_2\text{O}_3]$$

$$\frac{1}{4} (1 - [\text{FeSO}_4]) = [\text{O}_2]$$

And for more fun we can rewrite...

$$[\text{FeSO}_4] = a$$

We can then substitute the above relations into the K_{eq}-Concentration Equation

$$K_{\text{eq}} = \frac{(\frac{1}{2}(1-a))^1 (1-a)^2 (\frac{1}{4}(1-a))^{1/2}}{a^2}$$

And rewrite things a couple times...

$$4 a^2 K_{\text{eq}} = (1 - a)^{7/2}$$

$$16 a^4 K_{\text{eq}}^2 = (1 - a)^7$$

Then use Pascal's triangle to expand the right side of the equation above and ones like it, as this arithmetic was completed for each reaction.

$$16 a^4 K_{\text{eq}}^2 = 1 - 7a + 21a^2 - 35a^3 + 35a^4 - 21a^5 + 7a^6 - a^7$$

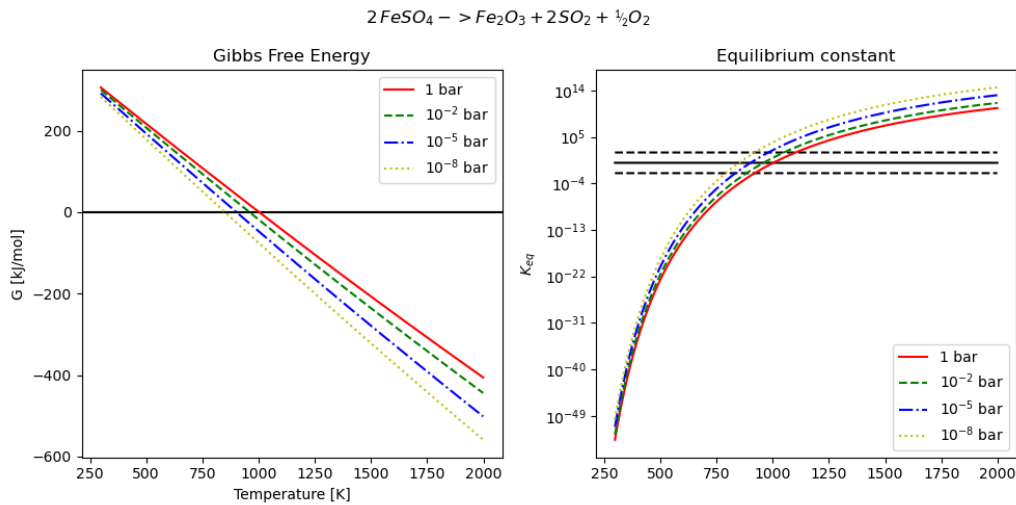
Re-written as:

$$0 = 1 - 7 a + 21 a^2 - 35 a^3 + (35 - 16 K_{\text{eq}}^2) a^4 - 21 a^5 + 7 a^6 - a^7$$

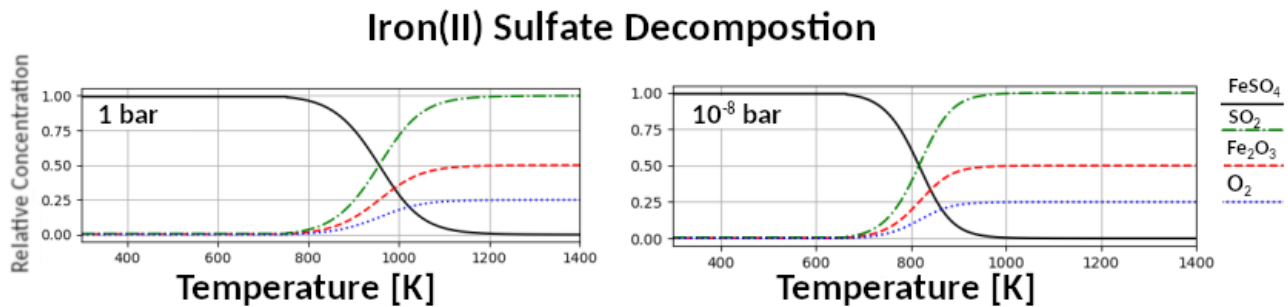
This is a fabulous equation, because we can enter the coefficients (1, -7, 21,...) into a polynomial solver (I used `numpy.roots()` in python) to calculate the possible values of 'a' that make this equation equal to zero. What this meant in practice is that for each reaction I would use the calculated values of ΔG at various pressures (10, 1, 10⁻⁴, and 10⁻⁸ atm; totaling 4 arrays) to calculate the K_{eq} for each constant pressure across the temperature range made available by the data. For an example, imagine that a temperature and a pressure combine for our beloved example reaction (FeSO₄ decomposition), such that K_{eq} = 1. Plugging this in gives us (roots = `np.roots((-1, 7, -21, 19, -35, 21, -7, 1))`):

```
array([ -0.51446146+1.33414631j, -0.51446146-1.33414631j,
        0.65650296+1.18949791j, 0.65650296-1.18949791j,
        0.2177188 +0.29589001j, 0.2177188 -0.29589001j,
        0.2804794 +0.j      ])
```

For this example, our solution, the correct value of a , and therefore the concentration of FeSO_4 , is the final element in the above array ($0.2804794+0.j$) as it is the only option with no imaginary component and is within the range from 0 to 1. By substituting the value of our concentration (0.2804794) into the relations above, we can calculate the concentrations of the other compounds in the reaction. By repeating the calculations for our predetermined Pressure-Temperature conditions (from $20\text{ }^\circ\text{C}$ to $1400\text{ }^\circ\text{C}$, or maximum temperature for which data was available; $10, 1, 10^{-4}$, and 10^{-8} atm discreet pressures), we can calculate, $TP\Delta G$ and K_{eq} .



As seen in above plot of our beloved reaction, it becomes spontaneous ($TP\Delta G = 0$) near $700\text{ }^\circ\text{C}$ (1000 K) depending on the pressure.



In the plot above I've taken the K_{eq} values at 1 and 10^8 bar (left and right, respectively), calculated the concentrations of all the reactants and products in terms of relative abundance, and plotted them as a function temperature (recall the system of equations with all concentrations in terms of $FeSO_4$). Note the lower temperature required for low-pressure decomposition. I repeated all of these steps for each reaction step listed in Anderson et al. 2021a

Yields Derivation

The final bit of math I did for this chapter was the calculations of the mass of the final products from the SSAP. For each ore (H, L, LL, CI, CM,-like asteroids, lunar highlands, lunar mare, and a Martian global average), I retrieved data from the literature (see Chapter IV references), specifically the abundances of various minerals in the ores and their mineral chemistries so that I could calculate the total number of mols of each end member on the reactant side, given 1 ton of the silicate-only ore. For an example I've shown the possible yield from just the Fayalite (Fe_2SiO_4 ; Olivine) in H chondrites. We start by normalizing each silicate mineral abundance to the total silicate abundance:

$$wt\%_{olivinenormalized} = \frac{wt\%_{olivine}}{wt\%_{olivine} + wt\%_{pyroxene} + wt\%_{plagioclase}}$$

$$m_{olivine} = (1000 \text{ kg}) * wt\%_{olivine \text{ normalized}}$$

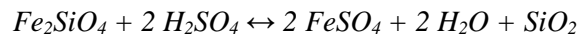
$$n_{mols \text{ Fa}} = m_{olivine} * mol\%_{Fa \text{ weighted}}$$

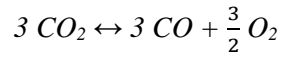
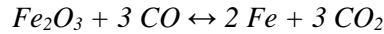
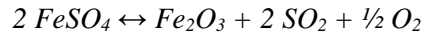
For instance, H chondrites have an average Olivine abundance of 32.86 wt%, which has an average Fayalite (Fe_2SiO_4) composition of 18.47 mol% (74.26 wt% total silicates). We can use this and the molar mass (amu subscript) of Fayalite and Forsterite (Mg_2SiO_4) to calculate the weighted mol percent, and therefore the number of mols of Fayalite:

$$mol\%_{Faweighted} = \frac{Fa_{mol\%}}{Fa_{mol\%}Fa_{amu} + Fo_{mol\%}Fo_{amu}}$$

$$n_{mols \text{ Fa}} = 536.67$$

Using the series of reactions laid out in Anderson et al. (2021b), I articulated the total number of mols of each product (Fe-metal, O_2 , etc.) and finally calculated the final mass.





Knowing that H₂O, SO₂ and CO are reused in this process, we can see that for every mol of Fayalite, 2 mols each of Fe metal and O₂, as well as 1 mol of SiO₂ are created. This means that for an H chondrite, the Fayalite (536 mols) could produce 1072 mols each of Fe metal and O₂, and 536 mols of SiO₂. Using this information and the relation between molecular mass and total mass, calculating the mass of each product was trivial:

$$m_{\text{product}} = \text{mols}_{\text{product}} * m_{\text{molecular}}$$

$$m_{\text{Fe}} = (1072 \text{ mols}) * (0.055845 \text{ kg/mol})$$

Calculations like these were repeated for every ore, with every mineral end member. The total products for an ore were then summed together and presented in Table 3 of Anderson et al. (2021b).

References

- Anderson, S., Towner, M., Bland, P., Haikings, C., Volante, W., Sansom, E., Devillepoix, H., Shober, P., Hartig, B., Cupak, M. and Jansen-Sturgeon, T., 2020. Machine learning for semi-automated meteorite recovery. *Meteoritics & Planetary Science*, 55(11), pp.2461-2471.
- Anderson, S., Benedix, G.K., Forman, L.V., Daly, L., Greenwood, R.C., Franchi, I.A., Friedrich, J.M., Macke, R., Wiggins, S., Britt, D. and Cadogan, J.M., 2021a. Mineralogy, petrology, geochemistry, and chronology of the Murrili (H5) meteorite fall: The third recovered fall from the Desert Fireball Network. *Meteoritics & Planetary Science*, 56(2), pp.241-259.
- Anderson, S.L., Sansom, E.K., Shober, P.M., Hartig, B.A., Devillepoix, H.A. and Towner, M.C., 2021b. The proposed Silicate-Sulfuric Acid Process: Mineral processing for In Situ Resource Utilization (ISRU). *Acta Astronautica*, 188, pp.57-63.
- Anderson, S. L., Towner, M.C., Fairweather, J., Bland, P.A., Devillepoix, H.A.R., Sansom, E.K., Cupak, M., Shober P.M., Benedix, G.K., 2022. Successful Recovery of an Observed Meteorite Fall Using Drones and Machine Learning. *The Astrophysical Journal Letters*, 930(2).
- Bland, P.A., Cressey, G. and Menzies, O.N., 2004. Modal mineralogy of carbonaceous chondrites by X-ray diffraction and Mössbauer spectroscopy. *Meteoritics & Planetary Science*, 39(1), pp.3-16.
- Bland, P.A., Spurný, P., Bevan, A.W.R., Howard, K.T., Towner, M.C., Benedix, G.K., Greenwood, R.C., Shrubný, L., Franchi, I.A., Deacon, G. and Borovička, J., 2012. The Australian Desert Fireball Network: A new era for planetary science. *Australian Journal of Earth Sciences*, 59(2), pp.177-187.
- Chollet, F. & others, 2015. Keras. Available at: <https://github.com/fchollet/keras>.
- Colas, F., Zanda, B., Vaubaillon, J., Bouley, S., Marmo, C., Audureau, Y., Kwon, M.K., Rault, J.L., Caminade, S., Vernazza, P. and Gattacceca, J., 2015, August. French fireball network FRIPON. In *Proceedings of the International*

Meteor Conference (Vol. 37).

- Dunn, T.L., McCoy, T.J., Sunshine, J.M. and McSween Jr, H.Y., 2010. A coordinated spectral, mineralogical, and compositional study of ordinary chondrites. *Icarus*, 208(2), pp.789-797.
- Garenne, A., Beck, P., Montes-Hernandez, G., Chiriac, R., Toche, F., Quirico, E., Bonal, L. and Schmitt, B., 2014. The abundance and stability of “water” in type 1 and 2 carbonaceous chondrites (CI, CM and CR). *Geochimica et Cosmochimica Acta*, 137, pp.93-112.
- Gilmour, C.M., Herd, C.D. and Beck, P., 2019. Water abundance in the Tagish Lake meteorite from TGA and IR spectroscopy: Evaluation of aqueous alteration. *Meteoritics & Planetary Science*, 54(9), pp.1951-1972.
- Heiken, G.H., Vaniman, D.T. and French, B.M., 1991. *Lunar Sourcebook, a user's guide to the Moon, Chapter 8*. P 357-475.
- Howie, R.M., Paxman, J., Bland, P.A., C Towner, M., Cupak, M., Sansom, E.K. and AR Devillepoix, H., 2017. How to build a continental scale fireball camera network. *Experimental Astronomy*, 43(3), pp.237-266.
- Jenniskens, P., Fries, M.D., Yin, Q.Z., Zolensky, M., Krot, A.N., Sandford, S.A., Sears, D., Beauford, R., Ebel, D.S., Friedrich, J.M. and Nagashima, K., 2012. Radar-enabled recovery of the Sutter’s Mill meteorite, a carbonaceous chondrite regolith breccia. *Science*, 338(6114), pp.1583-1587.
- King, A.J., Solomon, J.R., Schofield, P.F. and Russell, S.S., 2015. Characterising the CI and CI-like carbonaceous chondrites using thermogravimetric analysis and infrared spectroscopy. *Earth, Planets and Space*, 67(1), pp.1-12.
- King, H.E., Plümper, O., Geisler, T. and Putnis, A., 2011. Experimental investigations into the silicification of olivine: Implications for the reaction mechanism and acid neutralization. *American Mineralogist*, 96(10), pp.1503-1511.
- Lazaro, A., Brouwers, H.J.H., Quercia, G. and Geus, J.W., 2012. The properties of amorphous nano-silica synthesized by the dissolution of olivine. *Chemical Engineering Journal*, 211, pp.112-121.
- McSween Jr, H.Y., McGlynn, I.O. and Rogers, A.D., 2010. Determining the modal mineralogy of Martian soils. *Journal of Geophysical Research: Planets*, 115(E7).
- Meier, M.M.M. 2022, *Meteorite Orbits*, Matthias M. M. Meier, viewed 3 May 2022, <<https://www.meteoriteorbits.info/>>
- Nakamura, T., Noguchi, T., Tanaka, M., Zolensky, M.E., Kimura, M., Tsuchiyama, A., Nakato, A., Ogami, T., Ishida, H., Uesugi, M. and Yada, T., 2011. Itokawa dust particles: a direct link between S-type asteroids and ordinary chondrites. *Science*, 333(6046), pp.1113-1116.
- Salem, J.A., 2013. Transparent armor ceramics as spacecraft windows. *Journal of the American Ceramic Society*, 96(1), pp.281-289.
- Scott, E.R., Taylor, G.J. and Keil, K., 1986. Accretion, metamorphism, and brecciation of ordinary chondrites: Evidence from petrologic studies of meteorites from Roosevelt County, New Mexico. *Journal of Geophysical Research: Solid Earth*, 91(B13), pp.E115-E123.
- See, J.E., Howe, S.R., Warm, J.S. and Dember, W.N., 1995. Meta-analysis of the sensitivity decrement in vigilance. *Psychological bulletin*, 117(2), p.230.
- Spurný, P., Borovička, J. and Shrbený, L., 2006. Automation of the Czech part of the European fireball network: equipment, methods and first results. *Proceedings of the International Astronomical Union*, 2(S236), pp.121-130.
- Towner, M.C., Jansen-Sturgeon, T., Cupak, M., Sansom, E.K., Devillepoix, H.A.R., Bland, P.A., Howie, R.M., Paxman, J.P., Benedix, G.K. and Hartig, B.A.D., 2022. Dark-flight Estimates of Meteorite Fall Positions: Issues and a Case Study Using the Murrili Meteorite Fall. *The Planetary Science Journal*, 3(2), p.44.
- Trigo-Rodríguez, J.M., Llorca, J., Castro-Tirado, A.J., Ortiz, J.L., Docobo, J.A. and Fabregat, J., 2006. The Spanish fireball network. *Astronomy & Geophysics*, 47(6), pp.6-26.
- Van Herk, J., Pietersen, H.S. and Schuiling, R.D., 1989. Neutralization of industrial waste acids with olivine—The

dissolution of forsteritic olivine at 40–70 C. *Chemical Geology*, 76(3-4), pp.341-352. tric analysis and infrared spectroscopy. *Earth, Planets and Space*, 67(1), pp.1-12.

Chapter III: Meteorite Searching Pt. I

(Published in: *Meteoritics and Planetary Science*, 55(11), pp.2461-2471)

Machine Learning for Semi-Automated Meteorite Recovery

Seamus Anderson^{1*}, Martin Towner¹, Phil Bland¹, Christopher Haikings^{2,3}, William Volante⁴, Eleanor Sansom¹, Hadrien Devillepoix¹, Patrick Shober¹, Benjamin Hartig¹, Martin Cupak¹, Trent Jansen-Sturgeon¹, Robert Howie¹, Gretchen Benedix¹, Geoff Deacon⁵

¹Space Science and Technology Center, Curtin University, GPO Box U1987, Perth, WA 6845, Australia

²Spectre UAV Concepts, 191 St Georges Terrace, Perth, WA 6000, Australia

³Amotus Pty Ltd, Level 25/71 Eagle St, Brisbane City, QLD 4000, Australia

⁴Department of Psychology, Clemson University, 418 Brackett Hall, Clemson, SC, 29634

⁵Western Australian Museum, 49 Kew St, Welshpool, WA 6106, Australia

*Corresponding author. E-mail: seamus.anderson@postgrad.curtin.edu.au.

Abstract

We present a novel methodology for recovering meteorite falls observed and constrained by fireball networks, using drones and machine learning algorithms. This approach uses images of the local terrain for a given fall site to train an artificial neural network, designed to detect meteorite candidates. We have field tested our methodology to show a meteorite detection rate between 75-97%, while also providing an efficient mechanism to eliminate false-positives. Our tests at a number of locations within Western Australia also showcase the ability for this training scheme to generalize a model to learn localized terrain features. Our model-training approach was also able to correctly identify 3 meteorites in their native fall sites, that were found using traditional searching techniques. Our methodology will be used to recover meteorite falls in a wide range of locations within globe-spanning fireball networks.

INTRODUCTION

Fireballs and meteors have been observed since antiquity by Chinese, Korean, Babylonian and Roman astronomers (Bjorkman 1973), while meteorites and their unique metallurgical properties have also been known and used by various cultures around the world from Inuit tools (Rickard 1941) to Egyptian ceremonial daggers (Comelli et al. 2016), their connection to each other and to asteroids as source bodies was not proposed until the 19th century, with the fall of the l'Aigle meteorite (Biot 1803, Gounelle, 2006). Since this link was established, meteorites have, and continue to offer unique insights into the history of the solar system, as well as the contemporary characteristics, both physical and chemical, of asteroids, the Moon and Mars. Unfortunately, the overwhelming majority of these ~60,000 samples have no spatial context since their falls were not observed, leaving their prior orbits uncharacterized. Less than 0.1 % of meteorites in the global collection were observed well enough during their atmospheric entry to properly constrain their orbits (Meier 2017; Borovička et al. 2015; Jenniskens et al. 2020). This ultra-rare subset of meteorites afford some of the most valuable information pertaining to extra-terrestrial geology, since their physical and geochemical properties, along with their orbital histories can be combined to

characterize the nature of asteroid families, and therefore possible parent bodies, that inhabit the same orbital space.

The best methodology for recovering meteorites with corresponding orbits, utilizes fireball camera networks, which use automated all sky camera stations in an overlapping arrangement such that a potential fireball can be imaged by two or more stations. From these observations, scientists can triangulate an atmospheric trajectory, from which a pre-entry orbit and a fall area can also be calculated. The first success of such a system was demonstrated in Czechoslovakia in 1959, with the Příbram meteorite fall (Cepelcha 1961). This event spurred the establishment of the Czech fireball network (Spurný et al. 2006), along with multiple networks across the globe (McCorsky and Boeschstein, 1965; Halliday et al. 1996; Oberst et al. 1998; Bland 2004; Brown et al. 2010; Colas et al. 2014; Gardiol et al. 2016; Devillepoix et al. 2020).

The Desert Fireball Network

An ideal location for one of these networks was determined to be the Nullarbor region in Western and South Australia, due to its low humidity, sparse vegetation and typically clear skies (Bevan and Binns, 1989), thus the Desert Fireball Network (DFN) was born (Bland et al. 2012). Since its inception, the DFN has been responsible for the recovery of four confirmed meteorite falls: Bunburra Rockhole, Mason Gully, Murrili, and Dingle Dell (Spurný et al. 2012; Towner et al. 2011; Bland et al. 2016; Devillepoix et al. 2018), all of which have well constrained orbits. To date, the network covers approximately 30% of the land mass of Australia, with more than 50 camera stations (Howie et al. 2017). On average it observes 300 fireballs per year, typically 5 of which result in a meteorite fall.

For every fireball event observed by multiple camera stations, the bright flight trajectory is triangulated. If a terminal mass (meteorite fall) is predicted, we incorporate wind models into Monte Carlo simulations in order to estimate the likely fall area (Sansom et al. 2015; Howie et al. 2017; Jansen-Sturgeon et al. 2019). Since the fireball appears only as a streak of light, crucial attributes pertaining to the object such as size, mass and shape, are all co-dependent variables. This means that the predicted fall location results in a line, along which all of these parameters vary (Sansom et al. 2019). Inherent uncertainties and gaps in reported wind conditions at altitudes all along the flight, lead to a variation of ~250 m on either side of this fall line. Each predicted fall zone is entirely dependent on the conditions of the fireball, though typical events can result in a fall zone 2-4 km² in area. The decision to search for a particular meteorite is dependent on many factors, from the geometry and confidence of the trajectory triangulation, to local terrain features and geographic accessibility. Once the team has determined the fidelity of the triangulation and conditions of the fall area itself, a searching trip is commissioned to look for the fallen meteorite.

Meteorite Recovery

Traditional methods for meteorite recovery include two main strategies, petal searching and line searching. Petal searching involves sending individuals out from a central point, walking alone or in small groups in a loop, typically a few km long, looking for and collecting meteorites along the way. This method generally covers a larger area but comes with a higher risk of missing meteorites in the area covered. Since this method is usually implemented in strewn fields or in areas with older surface ages

and higher meteorite density, such as the Nullarbor (Bevan 2006), where the objective is to recover older meteorite finds, missing some meteorites is less detrimental.

Alternatively, line searching is more useful when trying to recover a meteorite fall with a well constrained fall line, like those observed through a fireball network. The DFN implements this strategy by assembling searchers in a line, spaced 5-10 m apart, then sweeping the area ~ 250 m on either side of the fall line on foot. This approach is usually able to cover 1-2 km² for each trip, assuming 6 people search 8 hours per day, for 10 days. The Antarctic Search for Meteorites (ANSMET) uses a similar method, only they are not restricted by a fall line, and instead cover the area with greater spacing while mounted on snowmobiles (Eppler 2011). The benefit of the line method is higher fidelity on the area covered, due to overlapping fields of view by the searchers, although generally, less area is covered with this method.

When considering both the number of meteorites found by the DFN, and the number of searching trips it has commissioned (4 and ~ 20 respectively), the success rate remains at $\sim 20\%$. This relatively low rate combined with the cost ($\sim 20,000$ AUD) of sending six people on trips for two weeks at a time necessitates an improvement in the meteorite recovery rate, particularly due to the establishment and expansion of the Global Fireball Observatory (Devillepoix et al. 2019).

Previous Drone-Meteorite Recovery Methods

The gargantuan strides that have been made in the last 10 years in the manufacture of high resolution DSLR cameras and commercial drones capable of carrying them, have opened the possibility of using both to aid in the recovery of meteorites. Previous attempts have been met with mixed to promising results. Moorhouse (2014) in his honors thesis, explored the possibility of using a hyper-spectral camera mounted to a drone to look for the possibly unique spectral signature of meteorites. This approach is unfeasible in our framework since the best hyper-spectral cameras are prohibitively expensive ($>100,000$ AUD), and more importantly, would limit our area coverage rate to little more than 0.1 km² per day, due to low spatial resolution in the camera. Further complications arise from the fact that Moorhouse used a spectral library of meteorite interiors, rather than meteoritic fusion crust, which is what would appear on the surface of fresh meteorite falls. Although meteorite fusion crusts could have a unique spectra compared to typical terrestrial environments, this is not explored in his work. Su (2017) focused on the feasibility of using magnetic sensors suspended from a drone, but this method would preclude us from finding non-magnetic meteorites, and also limited our area coverage to less than 0.1 km² per day. This approach would also be the most susceptible to obstacles on the ground and changes in local elevation, since they prescribe flying at a 2 m altitude.

Citron et al. (2017) relied on an RGB camera to survey an area, and used a machine learning algorithm to identify likely meteorites in the images. Their tests resulted in a meteorite detection rate of 50% and encountered a false positive rate of ~ 4 per 100 m². These results are very promising and seem to be limited mainly by the performance of the drone and camera hardware. The other limitation is false positives, and more importantly, how to separate them from promising meteorite candidates. This is a crucial detail when considering that a typical fall line (>2 km²), analyzed with their model, could have over 100,000 detections, all of which must be examined by a human in one way or another.

The work of Zender et al. (2018) also employed an RGB camera to image meteorites in native backgrounds. They showed the unique reflectance signature of meteorites in each color channel and

created an algorithm to detect these signatures. This approach was able to detect half of their test-meteorites, though it did suffer from a high rate of false positives.

AlOwais et al. (2019) also used an RGB camera, while additionally investigating the utility of a thermal imaging system. They also train a number of neural networks to detect meteorites within images. One of their chief priorities was to create such an image processing system that would fit on-board their surveying drone. With this in mind, they elected to use transfer learning (Pan and Yang 2009) from a handful of smaller pre-trained neural networks, to detect meteorites. Their training resulted in a high model accuracy using images taken from the internet, as well as photo-shopping cropped meteorite-images onto terrain backgrounds. These results are promising and await validation in the form of field tests.

Our previous work on drone-meteorite recovery is described in Anderson et al. (2019). In this previous iteration, we trained a machine learning model on a synthetic dataset. We created it by taking survey images from a drone, splitting them into tiles, then overlaying the tiles with cropped meteorite images. Although training on these tiles yielded a high training accuracy, it was unable to consistently identify real test-meteorites placed on the ground, mostly likely because the training data lacked the native lighting conditions and shadows seen in the real test-meteorite images.

Here, we report on updated methods to achieve a practical system for recovering meteorites using drones and machine learning. Such a system must fulfill the following 6 requirements to be effective:

- 1) Survey at least 1 km²/day,
- 2) Meteorite recovery chance (success rate) greater than 50%
- 3) Portable to different terrains/locations
- 4) Deployable by 3 people or less (1 vehicle)
- 5) Total cost <40,000 AUD (2 traditional searching trips)
- 6) Data processing rate equal to data surveying rate (including model prediction, and false-positive sorting)

METHODS

Drone and Camera Hardware

In recent years, the number of options for consumer and commercial drones has grown dramatically, with many options including fixed-wing, multi-copter, vertical takeoff/landing, and even blimps. The designs with the most flight-proven heritage, at our price range, are fixed-wing and multi-copters. Our previous experience has shown that fixed wing models produce too much image blur and are unable to achieve a meaningful image resolution due to lower limits on most models' cruising altitude. Given these constraints we chose a DJI M600 drone to perform full scale tests as well as surveys of our fall sites. This drone was able to carry our camera and gimbal payload with mass to spare for possible later upgrades. It was also able to perform pre-planned survey flights, with meter-scale GPS precision, for more than 15 min at a time.

We also decided to use an RGB camera, since these systems are both scalable and widespread, as opposed to thermal or hyper-spectral cameras that are more expensive, specialized and are only capable of smaller spatial resolutions. We specifically chose a Sony A7R Mk. 3 (42 MPixel), with a 35 mm Lens, set to take images with a 1/4000 sec exposure, at f/4.5, and an ISO of 320. The total cost of the camera, drone,

batteries, and accessories was 30,000 AUD, well below the 40,000 AUD limit we self-imposed in Criteria (5).

We used the DJI GO 4 app to control the drone manually during training data collection flights, while the survey flights were planned and executed using the DJI GS Pro app. With this equipment, we conducted tests at varying altitudes and determined that an image resolution of 1.8 mm/pixel (15 m altitude) would be sufficient to detect most of our typical meteorite falls (0.3 kg – 4 kg). This would allow meteorites to appear in the image between a size of 18 and 60 pixels in diameter. Using this fixed resolution value, we found that this system could survey approximately 1.3 km²/day, when we flew nearly continuously for 7 hrs per day, easily fulfilling Criteria (1). Although we had 12 hours of daylight at the time of our full scale test, we found that surveying less than 2.5 hours after sunrise or before sunset produced long shadows that resulted in an unacceptably high rate of false positives.

Machine Learning Software

Since a meteorite would appear to be small (18-60 pixels) relative to the total size of the image (42 MPixel), we decided to split each image into 125x125 pixel-tiles with a stride of 70. This allowed a meteorite to fully appear in at least one tile, to maximize the chance of detection and minimize false positives. These tiles were then fed to a binary image classifier, a type of deep convolutional neural network, to separate uninteresting terrain (0) from meteorite suspects (1). We considered any prediction over 0.9 confidence to be a detection, or a possible meteorite.

We implemented our neural network by constructing a model in python using tensorflow (Abadi et al. 2015) and keras (Chollet 2015), the architecture of which is shown in Table 1. Although a sufficiently deep architecture is important when training a neural network, the training data itself is the most important factor, especially in our case where we trained the model from randomized initial weights (from scratch), rather than using a pre-trained network. This means that for a given fall site we needed numerous, diverse examples of both True (meteorite) and False (non-meteorite) tiles. The False tiles were relatively easy to assemble. We simply took a survey of an area without any meteorites, and made all of the images into False tiles.

The True tiles required a bit more effort. Since all the meteorites we would be searching for would have fallen within the last 10 years, they would all have intact, dark fusion crusts covering their surface. Fresh meteorites such as these also tend to be minimally altered, making them more analytically valuable to the meteorite community. This consideration limited the number of real meteorites that were available to us to use in data sets. To artificially bolster the number of True tiles we could generate, we also used stones with desert varnish surfaces, a dark, slightly shiny exterior that develops on some rocks in hot deserts (Engel and Sharp 1958), as ‘synthetic’ meteorites. At each site we also found stones that had a plausible meteoritic shape (non-jagged and without a noticeable elongated axis) and painted them black. Using this combination of fusion-crusted meteorites, desert varnish stones and painted stones, we always had enough samples to make a substantial number of True tiles.

Our procedure for making these tiles is illustrated in Figure 1. Step 1 consisted of laying out the stones in a line at the fall site, spaced more than 1 m apart, and then imaging them with the drone. This line could be either in the fall zone, or just outside of it, in order to train the model on similar backgrounds. We gave a 1 m separation to ensure that two stones would not appear in the same tile, when we augmented

the data later on. We found the best way to accomplish this stone-imaging was for one person to walk ~ 3 m parallel to the line of black stones, and point to each one, while another person manually flew the drone at the prescribed survey altitude, following the first person. Physically pointing out each individual stone allowed us to annotate each stone only once, avoiding a possible double appearance of a particular stone in both the training and validation sets. For Step 2, we drew a tight bounding box around each stone and recorded the box’s height, width and position in the image. These annotations were completed using ImageJ (Schneider et al. 2012). We typically laid out ~ 100 stones at a time; 15% of these stones and their resultant tiles are set aside for validation, not used in training. This ensured that the validation set only consisted of stones that the model had never seen, as opposed to unseen permutations of stones that the model was already familiar with.

Table 1. Meteorite-Detecting Neural Network Architecture

Layer type	Filters (conv.) / neurons (dense)	Kernel size	Stride size	Activation function
Convolutional 2D	30	3	1	Rectified Linear Unit
Batch Normalization				
Max Pooling		2	2	
Convolutional 2D	60	3	1	Rectified Linear Unit
Batch Normalization				
Max Pooling		2	2	
Convolutional 2D	120	2	2	Rectified Linear Unit
Batch Normalization				
Max Pooling		2	1	
Convolutional 2D	240	3	1	Rectified Linear Unit
Batch Normalization				
Max Pooling		2	1	
Flatten				
Dense	100			Rectified Linear Unit
Dropout	30 %			
Dense	50			Rectified Linear Unit
Dense	25			Rectified Linear Unit
Dense	5			Rectified Linear Unit
Dropout	30%			
Dense	5			Rectified Linear Unit
Dense	1			Sigmoid

At Step 3, we took each annotation, in both the training and validation sets, and strode by 15 pixels in both axes over each meteorite, creating a new tile at each stride, while keeping the stone fully in the tile frame. Each of those tiles was then rotated in intervals of 90 degrees and saved for each permutation. These strides and rotations force each rock to appear in nearly every position of a tile, without any preference in local directionality, i.e. shadows and windblown vegetation. We repeat this data-collection process at different times of the day, at different sections of the fall line, to include as much variety as possible. Details like these are crucial when making a widely generalized training set. This process ideally generates ~50,000 True tiles for the training set. To assemble the False tiles, we flew the drone 350 m, parallel to the fall line, taking images all along the way. By splitting the images into tiles, we generated ~2,500,000 False samples. The process of laying out stones, imaging everything, and making the annotations typically took an hour.

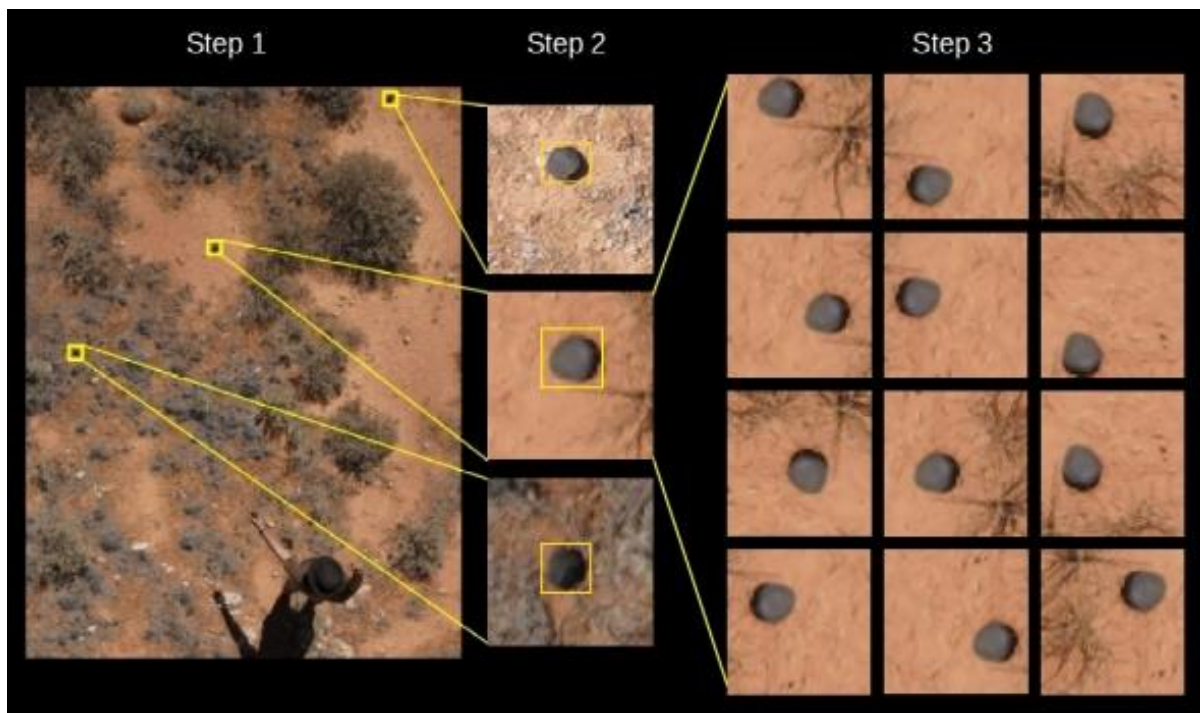


Fig. 1. Our workflow for obtaining meteorite training data. Step 1 consists of laying out the stones on the ground >1 m apart, and imaging them at a 1.8 mm/pixel resolution. Step 2 shows how we record the position, height and width of each rock in the full-sized image, by drawing a bounding box. Step 3 is where we generate the tiles to be used for training and validation.

Since dramatically unbalanced datasets can negatively affect training (Miroslav and Matwin 1997), we could only train with as many False tiles as we had True tiles, to keep a 1:1 ratio. A simplified example of unbalanced datasets is a training set containing 1 True and 99 False samples. Mathematically speaking, the shortest path to the model achieving a high accuracy would be for it to label everything false, resulting in an accuracy of 99%. Obviously, this kind of solution is useless, which is why we must maintain a ratio as close to 1:1 as possible. We did this by randomly selecting 50,000 False tiles from the pool of 2.5 million, and combining them with the 50,000 True tiles, to form the whole training set. We also included ~8,000 False tiles from the 2.5 million into the validation set, ensuring they did not also appear in the training set.

We trained on our dataset for 150 epochs (rounds of training), using a batch size of 250, with 400 steps per epoch, which ensured that each tile is seen by the model once per epoch. The validation set was evaluated at the end of each epoch, also using a batch size of 250 with 64 steps. For smaller datasets, we adjusted our batch size and steps per epoch such that the product of these two values equaled the size of the training set.

Once the model completed training, we judged its utility based on its meteorite detection chance, and rate of false positives. The meteorite detection chance was determined by predicting on each of the True tiles in the validation set, and dividing the number predictions over 0.9 confidence by the total number of tiles. This provided a metric for how well the model could correctly identify new black rocks that it had never seen. For the false positive rate, we wanted to obtain a more widespread and representative value that would reflect model performance across the whole fall zone. So we randomly selected 50 images from the survey of the fall line and predicted on them with the model, recording the average number of detections across all the images.

Model Detection Sorting Interface

An issue we anticipated with any model we would train, was the processing of false positives. Even in best case scenarios, where we assume a model accuracy of 99.999%, with ~8,500 tiles per image and ~650 images per flight, a model would return approximately 5,500 detections per flight, and more than 150,000 per fall line. Thus, we required a tool to help searchers efficiently examine each of these model-detections, and determine which of these were obvious false positives and which ones required further investigation. We created a graphical user interface in python using the Tkinter module to accomplish this task (Figure 2).

The program displays nine detections at a time, in a 3x3 grid pattern. Each grid space is mapped to numbers 1-9 on a standard keyboard's keypad (1 for lower left, 5 for middle-center, 9 for upper right). Each detection is displayed such that the frame is centered on the detection tile, outlined in a yellow box (~25 cm on one side), and extends 70 pixels beyond the target tile, to give the user context of the larger area. Below the grid, 3 images of meteorites are displayed, scaled from the smallest to the largest meteorite possible for that fall site (lowest mass with iron density, to highest mass with chondritic density, respectively). This allows users to easily reference how big a meteorite should appear in the tiles. If the user decides that the tile likely contains a meteorite, they press the number on the keypad corresponding to that grid space, before advancing to the next set. The program also allows the user to remove their responses from the current set of 9 tiles, as well as go back to the previous set.

Through testing trails, we determined that the average user could sort through ~120 tiles per minute. Assuming 150,000 detections per fall line, the task of sorting through this data would take over 20 labor-hours. This problem of staying focused over long periods of time is known as 'vigilance' by human factors psychologists, who have observed decrements in user performance over extended task sessions (See et al. 1995). To mitigate such decrements in vigilance, we ensured that each user would only sort for 20-minute increments. This was chosen as a conservative time limit according to Teichner (1974), who found the vigilance decrement to be fully observed 20-35 minutes into a task. Additionally, to reduce the consequence of individual errors, each tile was inspected by two separate users. We also anticipated that the overwhelming majority of detections would be false positives, thereby counter-productively enticing the users to speed through the tiles, without properly inspecting each one. The resulting consequence of such task parameters has been shown in signal detection literature to result in the missed

detection of such rare signals (Stanislaw & Todorov 1999), in our case the user-detection of a meteorite. To combat this, we added a test function to the program, whereby each set of nine tiles had a uniform probability of containing 0, 1 or 2 test tiles, taken from the training set. This forced the user to slow down and select, on average, one tile per set, thus reducing the rarity of a “hit”.

A final failsafe was included in this sorting task, such that once the user missed two test tiles during a sorting session, the program would shut down, forcing the user to take a break. The user’s score of successfully completed tests, along with the number of meteorite candidates identified, are shown at the top of the display. Both of these strategies; increasing the “hit” rate and providing performance feedback, have been shown to combat the vigilance decrement (Hancock et al. 2016). Once two users sorted through the detections for a flight, we overlaid the original images with bounding boxes around meteorite candidates. We also set aside the false positives, so that we may use them for retraining if needed.

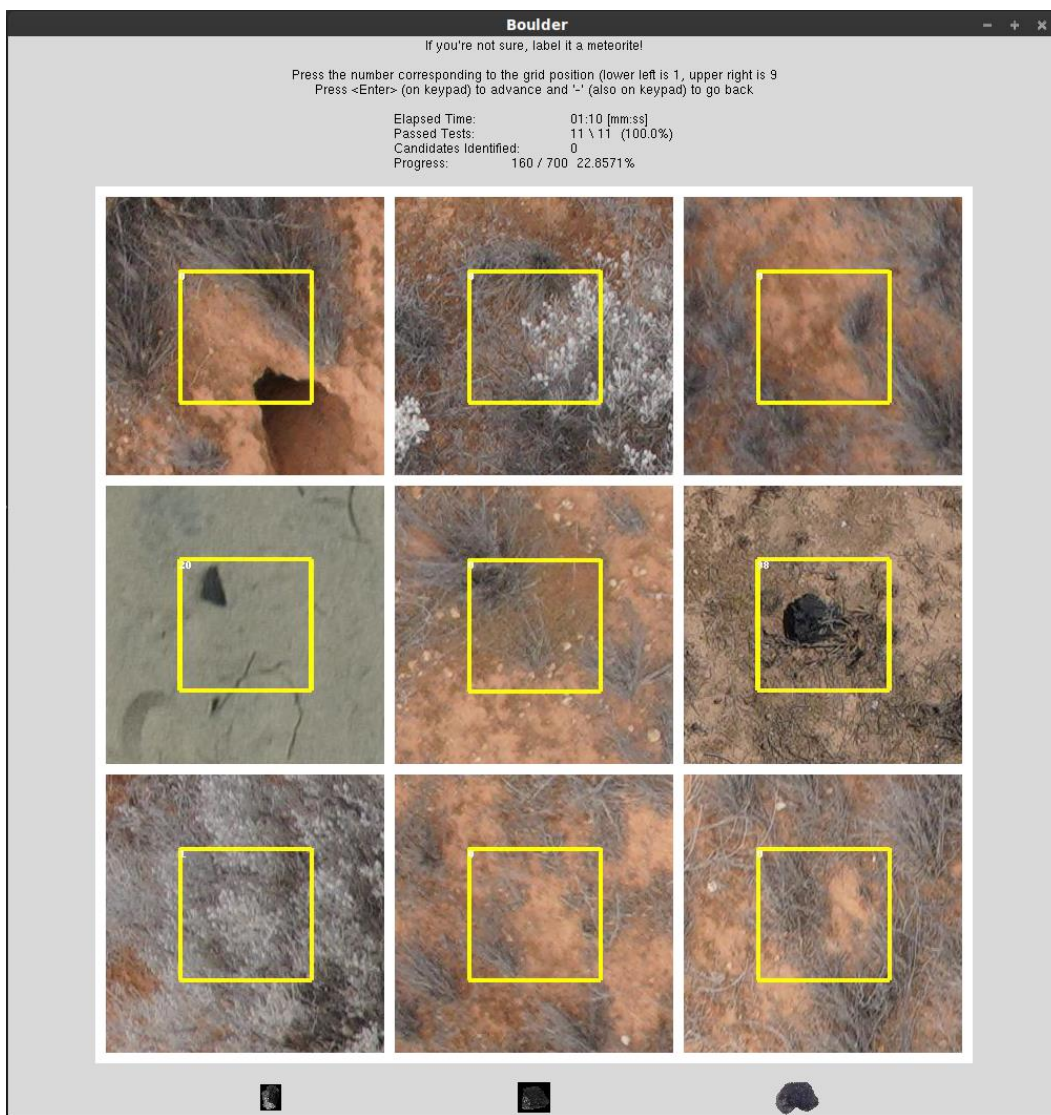


Fig. 2. Our sorting user interface we designed to aid in the separation of candidates from false positives. Test tiles (True tiles from the model training set) for this set appear in the center-left and center-right positions. Users press the corresponding number on the keypad to mark the tile(s) as a likely meteorite.

RESULTS

We conducted small scale tests of our methodology by visiting 4 sites in Western Australia and training a model at each location. Although they were not at real meteorite fall sites, they were all located within the DFN's operational area, and could conceivably be representative of future fall sites. These sites and the results from the models we trained for them are listed in Table 2. For these smaller tests, we only obtained training data for ~30 synthetic and real meteorites, and surveyed less than 0.1 km² at each site.

Table 2. Distinct models at various locations within the DFN. Model performance is dependent on the size of the training dataset.

Location	(Lat, Long)	Total Number of Training Tiles	Training Accuracy	Meteorite Detection	False Positives (per image)
Ledge Point	(-31.151, 115.395)	16,352	92.58 %	68.5 %	21.7
Dalgaranga	(-27.635, 117.289)	30,874	97.03 %	85.6 %	6.5
Lake Kondinin	(-32.496, 118.192)	32,348	96.85 %	86.7%	5.1
Balladonia	(-32.370, 124.790)	98,470	98.73 %	93.2 %	1.3

We also conducted a full test of our methodology by visiting one of our fall sites, DN150413_01, North-East of Forrest Airport, Western Australia (30.764 S, 128.184 E). We obtained training data for this fall site at different times during the day (morning, cloudy mid-morning, midday, early afternoon, and late afternoon). Over the course of two days we also surveyed 2 km² of the fall zone, so that we could identify meteorite candidates for secondary inspection in an upcoming expedition. We also placed 4 painted rocks, unseen by the model, within the survey area and recorded their GPS coordinates. This served as a test of our ability to use the model to correctly identify a meteorite candidate, correctly sort it using the user interface, and accurately correlate the image's GPS coordinates to the those recorded by our handheld unit. During the survey, each of three team-members were assigned a distinct role during survey-flight operations. The first team-member's job was to fly the drone and calibrate the camera, the second oversaw data collection and backups on the computer, and the third was responsible for cooling and charging the batteries.

When we returned from the field, we trained a model on our RTX 2080 Ti (11 GB RAM) GPU, with an Intel i9-9000 CPU for approximately 3 hours (150 epochs). This resulted in a final training accuracy of 99.07% and a validation accuracy of 98.65%. Furthermore, we achieved a meteorite detection chance of 98.71%, and a false positive rate of 2.5 per image. Using the trained model, the detection algorithm was able to process 1 day's images in 22 hours. The model returned a combined total of 92,595 detections for the two-day survey, which we were able to sort through in 12 hours, excluding breaks. Sorting through all of our detections yielded 752 meteorite candidates, some of which are shown in Figure 3. Of the four test rocks we laid out, we successfully located three of them (by comparing GPS coordinates) using our prescribed searching methods, meaning that we successfully met fulfilled criteria: (2).

Four months after this initial trip, when the COVID-19 travel restrictions were lifted in Western Australia, we revisited the same site North-East of Forrest Airport. We began by inspecting ~20 of the 749 candidates in-person and noticed that they generally belonged to one of two populations: dark stones (most likely iron-rich siliceous rock), and small holes in the ground (<7 cm in diameter) most likely made by small animals. The small hole population was far more numerous than the dark stone group and were easy to distinguish in the images, once we knew which features to look for. We then sorted through the remaining ~700 candidate images and narrowed the list to 32 candidates that did not appear to be holes in the ground. Unfortunately after inspecting these remaining candidates, we found that none of them were meteorites

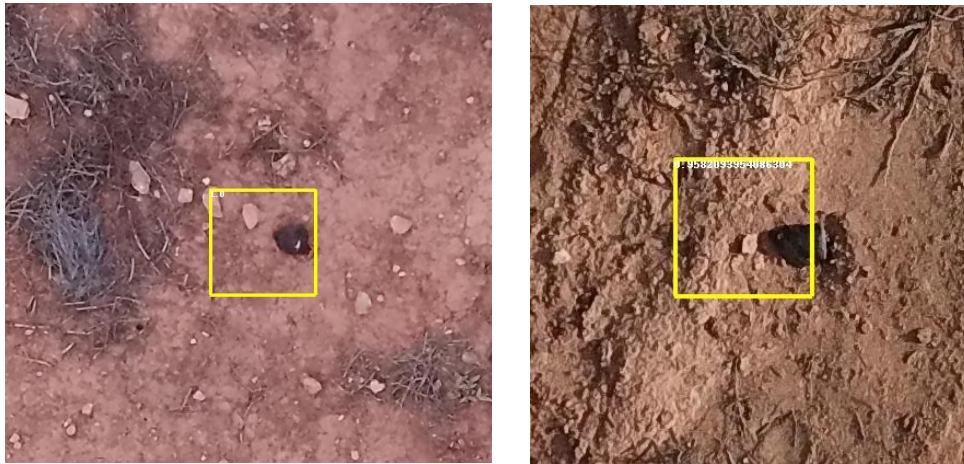


Fig 3. The meteorites recovered North-West of Forrest Airport (left) and South of Madura (Right). Although these two were not recovered using the full surveying methodology, they serve as valuable demonstrations for the feasibility of our approach.

On this same follow-up trip, we also visited a second, separate fall site located North-West of Forrest Airport. For this site, we employed the traditional line searching technique and found the meteorite on the afternoon of the first day. Using our Mavic Pro drone, we took ~100 images of this meteorite (Fig. 3, left) from a top down view, with heights ranging from 1 to 30 m. We also generated training data at this site, trained a model, and used it to predict on 86 of these images (those in which the meteorite was between 10 and 80 pixels in diameter). The model was able to correctly identify the meteorite in 84 of the 86 images, or 97%.

During a separate trip, whereby two members of our research group were scouting a third fall site South of Madura, Western Australia, for an upcoming six-person searching trip, they discovered the meteorite in question (Fig. 3, right), on the dirt road which roughly bisected the predicted fall line. They also used the Mavic Pro to take images of the meteorite from altitudes of 2 to 30 m, and created training data on-site. When they returned from the trip, we trained a model and predicted on the 27 meteorite images, finding that the model correctly identified the meteorite in 24 of the images (88% success rate). At the writing of this manuscript, these two meteorites have not yet been registered with the Meteoritical Bulletin, as their classifications are forthcoming.

An additional and final test of our approach involved using our Forrest-NE model to predict on a drone-image of an older meteorite find, shared with us by a volunteer meteorite hunter who regularly searches in the Nullarbor. We found that the model correctly identified the old chondrite with a prediction value of 1.0: a perfect match.

DISCUSSION AND FUTURE WORK

Our smaller tests (Table 2) show that more training data makes for a more robust model in terms of both meteorite detection and false positives, reinforcing the notion that more training data makes for a better model. These tests also showcase the portability of our methodology, accounting for variations in available training data, which successfully satisfies Criteria (3).

The results of the full test, while not a total success, are a promising prospect for the future of meteorite recovery. Not only is this methodology capable of locating test meteorites analogues, it is able to cover a fall zone nearly 6 times faster than a traditional line-search, when accounting for invested labor. If we assume that in the future we would predict on images and sorting through detections in the field, this rate of data processing can keep pace with data collection through a combination of switching sorting users and simply taking breaks, satisfying the final outstanding Criteria (6).

There are two possibilities as to why the full test did not result in a complete success of recovering the meteorite. The first explanation is that our methodology failed at some stage of the searching, whether the model failed to detect the meteorite, or we failed to label the detection as a candidate. The second possibility was that we did not cover enough of the fall line. Since the initial surveying trip was limited to two days, we were only able to cover 2 km² of the entire 5 km² fall zone. Our other successes with the models correctly identifying two fresh falls and one old find, all in situ, lead us to believe that the second explanation is more likely. For this reason, we plan on returning to the Forrest-NE fall site and surveying the remainder of the fall zone.

We will also embark on an extensive surveying campaign of all of our meteorite fall sites. We initially plan on training a new model for each fall site, using randomly initialized weights. Though as we gain more training data from a range of diverse fall sites, we will investigate the possibility of combining data sets and training a ‘base model’ whose final weights will then be used as the initial weights for each new model we train. This future approach may improve the generalizability of our models and reduce training time on-site.

The python software that we have created, as well as our trained model weights, will be made available to collaborators upon request, so that the entire meteoritics community can benefit from this new method of semi-automated meteorite recovery.

Acknowledgements - We would like to thank Robert Tower for providing us with additional drone-survey images. This work was funded by the Australian Research Council (Grant: DP200102073).

REFERENCES

- Abadi M., Agarwal A., Barham P., Brevdo E., Chen Z., Citro C., Corrado G. S., Davis A., Dean J. Devin, M., Ghemawat S., Goodfellow I., Harp A., Irving G., Isard M., Jozefowicz R., Jia, Y., Kaiser L., Kudlur M., Levenberg J., Mané D., Schuster M., Monga R., Moore S., Murray D., Olah C., Shlens J., Steiner B., Sutskever I., Talwar K., Tucker P., Vanhoucke V., Vasudevan V., Viégas F., Vinyals O., Warden P., Wattenberg M., Wicke M., Yu Y., Zheng X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems, *tensorflow.org* (Accessed 21 September 2020)
- AlOwais A., Naseem S., Dawdi T., Abdisalam M., Elkalyoubi Y., Adwan A., Hassan K. Fernini I. 2019. Meteorite Hunting Using Deep Learning and UAVs. *2nd International Conference on Signal Processing and Information Security (ICSPIS)*. pp. 1-4
- Anderson S. L., Bland, P. A., Towner M. C., Paxman J. P. 2019. Utilizing Drones and Machine Learning for Meteorite Searching and Recovery (Abstract #2426.). *49th Lunar and Planetary Science Conference*.
- Bevan A. W. R. 2006. Desert meteorites; a history. *Geological Society, London, Special Publications 256* (1): 325-343.
- Bevan A. W. R., Binns R. A. 1989. Meteorites from the Nullarbor Region, Western Australia: II. Recovery and classification of 34 new meteorite finds from the Mundrabilla, Forrest, Reid and Deakin areas. *Meteoritics* 24(3):135-141.
- Biot J. B. 1803. *Relation d'un voyage fait dans le department de l'Orne, pour constater la realite d'un meteore observe a l'Aigle le 26 floreal an II*. Baudouin, Imprimeur de L'Institut National, Paris.
- Bjorkman J. K. .1973. Meteors and Meteorites in the Ancient Near East. *Meteoritics* 8(2): 91-132.
- Bland P. A. 2004. The Desert Fireball Network. *Astronomy and Geophysics* 45(5):520-523.
- Bland P. A., Spurny P., Bevan A. W. R., Howard K. T., Towner M. C., Benedix G. K., Greenwood R. C., Shrbeny L., Franchi I., Deacon, G., Borovicka J., Ceplecha Z., Vaughan D., Hough R. M. 2012. The Australian Desert Fireball Network: A new era for planetary science. *Australian Journal of Earth Sciences* 59(2):177-187.
- Bland P. A., Towner M. C., Sansom E. K., Devillepoix H., Howie R. M., Paxman J. P., Cupak M., Benedix G. K., Cox M. A., Jansen-Sturgeon T., Stuart, D. 2016. Fall and recovery of the Murrili meteorite, and an update on the desert fireball network (Abstract #6265). *79th Annual Meeting of the Meteoritical Society*.
- Borovička J., Spurný P., Brown P. 2015. Small Near-Earth Asteroids as a Source of Meteorites. *Asteroids IV*, University of Arizona Press, Tucson, pp. 257-280.
- Brown P., Weryk R.J., Kohut S., Edwards W.N. and Krzeminski Z., 2010. Development of an all-sky video meteor network in Southern Ontario, Canada The ASGARD System. *WGN, Journal of the International Meteor Organization* 38:25-30

Ceplecha Z., 1961. Multiple fall of Pribram meteorites photographed. 1. Double-station photographs of the fireball and their relations to the found meteorites. *Bulletin of the Astronomical Institutes of Czechoslovakia* 12: 21.

Chollet F. 2015. Keras. *keras.io* (Accessed 21 September 2020)

Citron R. I., Shah A., Sinha S., Watkins C., Jenniskens P. 2017. Meteorite Recovery Using an Autonomous Drone and Machine Learning (Abstract #2528). *48th Lunar and Planetary Science Conference*.

Colas F., Zanda B., Bouley S., Vaubaillon J., Vernazza P., Gattacceca J., Marmo C., Audureau Y., Kwon M.K., Maquet, L., Rault J.L., 2014, September. The FRIPON and Vigie-Ciel networks. *33rd International Meteor Conference*. pp. 18-21.

Comelli D., D'orazio M., Folco L., El-Halwagy M., Frizzi T., Alberti R., Capogrosso V., Elnaggar A., Hassan H., Nevin A., Porcelli F. 2016. The meteoritic origin of Tutankhamun's iron dagger blade. *Meteoritics & Planetary Science* 51(7):1301-1309.

Devillepoix H.A.R., Sansom E.K., Bland P.A., Towner M.C., Cupák M., Howie R.M., Jansen-Sturgeon T., Cox M.A., Hartig B.A., Benedix G.K., Paxman J.P. 2018. The Dingle Dell meteorite: A Halloween treat from the Main Belt. *Meteoritics & Planetary Science* 53(10):2212-2227.

Devillepoix H. A. R., Cupak M., Bland P. A., Sansom E. K., Towner M. C., Howie R. M., Hartig B. A. D., Jansen-Sturgeon T., Shober P. M., Anderson S. L., Benedix G. K., Busan D., Sayers R., Jenniskens, P., Albers J., Herd C. D. K., Carlson P., Hill P. J. A., Brown P. G., Krzeminski Z., Osinski G. R., Chennaoui Aoudjehane H., Shisseh T., Benkhaldoun Z., Jabiri A., Guennoun M., Barka A., Darhmaoui H., Daly L., Colline G. S., McMullan S., Suttle M. D., Shaw C., Young J. S., Alexander M., Mardon A. D., Ireland T., Bonning G., Baeza L., Alrefay T. Y., Horner J., Swindle T. D., Hergenrother C. W., Fries M. D., Tomkins A., Langendam A., Rushmer T. A., O'Neill C., Janches D., Hormaechea J. L., 2020. A Global Fireball Observatory. *arXiv preprint arXiv:2004.01069*.

Engel C.G. and Sharp R.P., 1958. Chemical data on desert varnish. *Geological Society of America Bulletin* 69(5):487-518.

Eppler D.B. 2011. Analysis of Antarctic logistics and operations data: results from the Antarctic Search for Meteorites (ANSMET), austral summer season, 2002–2003, with implications for planetary surface operations. *Geological Society of America Special Papers* 483:75-84.

Gardiol D., Cellino A., Di Martino M. 2016. PRISMA Italian network for meteors and atmospheric studies. *Proceedings of the International Meteor Conference 2016*, Egmond, Netherlands, pp. 76-79.

Gounelle M. 2006. The Meteorite Fall at L'Aigle and the Biot Report: exploring the cradle of meteoritics. *Geological Society London Special Publications* 256(1): 73-89.

Halliday I., Griffin, A.A. and Blackwell, A.T., 1996. Detailed data for 259 fireballs from the Canadian camera network and inferences concerning the influx of large meteoroids. *Meteoritics & Planetary Science* 31(2):185-21

Hancock P. A., Volante W. G., Szalma J. L. 2016. Defeating the vigilance decrement. *IIE Transactions on Occupational Ergonomics and Human Factors* 4:151-163.

Howie R.M., Paxman J., Bland P.A., Towner M.C., Cupak M., Sansom E.K., Devillepoix, H.A. 2017. How to build a continental scale fireball camera network. *Experimental Astronomy* 43(3):237-266.

Jansen-Sturgeon T., Sansom E. K., Bland P. A. 2019. Comparing analytical and numerical approaches to meteoroid orbit determination using Hayabusa telemetry. *Meteoritics and Planetary Science* 54:2149-2162.

Jenniskens P. 2020. Review of asteroid-family and meteorite-type links. *Astronomy in Focus, Proceedings of the IAU* 14:9-12.

McCorsky R. E. and Boeschenstein H. 1965. The Prairie Meteorite Network. *Optical Engineering* 3(4):304-127

Meier M. M. M. 2017. Meteoriteorbits.info – Tracking all known meteorites with photographic orbits (Abstract #1178). *49th Lunar and Planetary Science Conference*.

Miroslav K. and Matwin S. 1997. Addressing the curse of imbalanced training sets: one sided selection. *International Conference on Machine Learning* 97:179-186.

Moorhouse D. 2014. Hyper-Spectral Imaging for Airborne Meteorite Detection. *Honors Bachelor Thesis, University of Southern Queensland, Toowoomba, Australia*.

Oberst J., Molau S., Heinlein D., Gritzner C., Schindler M., Spurný P., Ceplecha Z., Rendtel J., Betlem H. 1998. The European Fireball Network: current status and future prospects. *Meteoritics and Planetary Science* 33:49-56.

Pan S. J., Yang Q. 2009. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 22(10):1345-1359.

Rickard, T. A. 1941. The use of meteoric iron. *The Journal of the Royal Anthropological Institute of Great Britain and Ireland* 71:55-56.

Sansom E. K., Bland P., Paxman J., Towner M. 2015. A novel approach to fireball modeling: The observable and the calculated. *Meteoritics and Planetary Science* 50:1423-1435.

- Sansom E.K., Jansen-Sturgeon T., Rutten M.G., Devillepoix H.A., Bland P.A., Howie R.M., Cox M.A., Towner M.C., Cupák M. and Hartig B.A. 2019. 3D meteoroid trajectories. *Icarus* 321:388-406.
- Schneider C. A., Rasband W. S., Eliceiri K. W. 2012. NIH Image to ImageJ: 25 years of image analysis. *Nature Methods* 9:671-675.
- See J. E., Howe S. R., Warm, J. S., Dember W. N., 1995. Meta-analysis of the sensitivity decrement in vigilance. *Psychological Bulletin* 117:230-249.
- Spurný P., Borovicka J., Shrbený L. 2006. Automation of the Czech part of the European fireball network: equipment, methods and the first results. *Proceedings of the International Astronomical Union* 2.S236, pp. 121-130.
- Spurný P., Bland P.A., Shrbený L., Borovička J., Ceplecha Z., Singelton A., Bevan A.W., Vaughan D., Towner M.C., McClafferty T.P., Toumi R., 2012. The Bunburra Rockhole meteorite fall in SW Australia: fireball trajectory, luminosity, dynamics, orbit, and impact position from photographic and photoelectric records. *Meteoritics & Planetary Science* 47:163-185.
- Stanislaw H., Todorov N. 1999. Calculation of signal detection theory measures. *Behavior research methods, instruments, & computers* 31(1):137-149.
- Su D. 2017. Drag-Line Sensor for Drone Assisted Meteorite Detection. *Honors Bachelor Thesis Macquarie University, Sydney, Australia.*
- Teichner W H., 1974. The detection of a simple visual signal as a function of time of watch. *Human Factors* 16:339-352.
- Towner M.C., Bland P.A., Spurný P., Benedix G.K., Dyl K., Greenwood R.C., Gibson J., Franchi I.A., Shrbený L., Bevan A.W.R., Vaughan D. 2011. Mason Gully: The second meteorite recovered by the Desert Fireball Network. *Meteoritics and Planetary Science Supplement* 74, p.5124
- Zender J., Rudawska R., Koschny D., Drolshagen G., Netjes G.J., Bosch M., Bijl R., Crevecoeur R. Bettonvil F. 2018. Meteorite detection with airborne support—a study case. *Proceedings of the International Meteor Conference, Pezinok-Modra, Slovakia*, pp. 145-152

Chapter IV: Meteorite Searching Pt. II

(Published in: *Astrophysical Journal Letters*, 930(2), p.L25)

Successful Recovery of an Observed Meteorite Fall Using Drones and Machine Learning.

Seamus L. Anderson^{1*}, Martin C. Towner¹, John Fairweather¹, Philip A. Bland¹, Hadrien A. R. Devillepoix¹, Eleanor K. Sansom¹, Martin Cupak¹, Patrick M. Shober¹, Gretchen K. Benedix¹

¹Space Science and Technology Centre, School of Earth and Planetary Science, Curtin University, 314 Wark Ave, Bentley, WA, Australia 6102

*seamus.anderson@postgrad.curtin.edu.au

Abstract

We report the first-time recovery of a fresh meteorite fall using a drone and a machine learning algorithm. The fireball was observed on 1st April 2021 over Western Australia by the Desert Fireball Network, for which a fall area was calculated for the predicted surviving mass. A search team arrived on site and surveyed 5.1 km² area over a 4-day period. A convolutional neural network, trained on previously-recovered meteorites with fusion crusts, processed the images on our field computer after each flight. Meteorite candidates identified by the algorithm were sorted by team members using two user interfaces to eliminate false positives. Surviving candidates were revisited with a smaller drone, and imaged in higher resolution, before being eliminated or finally being visited in-person. The 70 g meteorite was recovered within 50 m of the calculated fall line, demonstrating the effectiveness of this methodology which will facilitate the efficient collection of many more observed meteorite falls.

1. Introduction

Besides recording the conditions of the proto-planetary nebula and the early Solar System, meteorites also offer insights into the contemporary physical and chemical compositions of Asteroids and other terrestrial bodies (Cuzzi et al. 2008; Nakamura et al. 2011). Some of these meteorites fall in regions on Earth where fireball observatory networks are active, making it possible to record the trajectory of the fireball as it ablates material from the originating meteoroid. For some fireballs, this data can then be used to simulate both forward and backward in time to predict where the resulting meteorite landed on Earth and where the meteoroid originated in the solar system. Thus, recovering and analyzing these ‘orbital meteorites’ with constrained, prior orbits provides an incredibly unique insight into the geology and dynamic behaviour of the asteroid belt and the nature of mass transfer between the belt and the inner solar system. The Desert

Fireball Network (DFN) (Bland et al. 2012; Howie et al. 2017) is one of many research groups (Oberst et al. 1998; Spurný et al. 2006; Trigo-Rodríguez et al. 2006; Olech et al. 2006; Colas et al. 2015) within the Global Fireball Observatory (Devillepoix et al. 2020) that makes this possible.

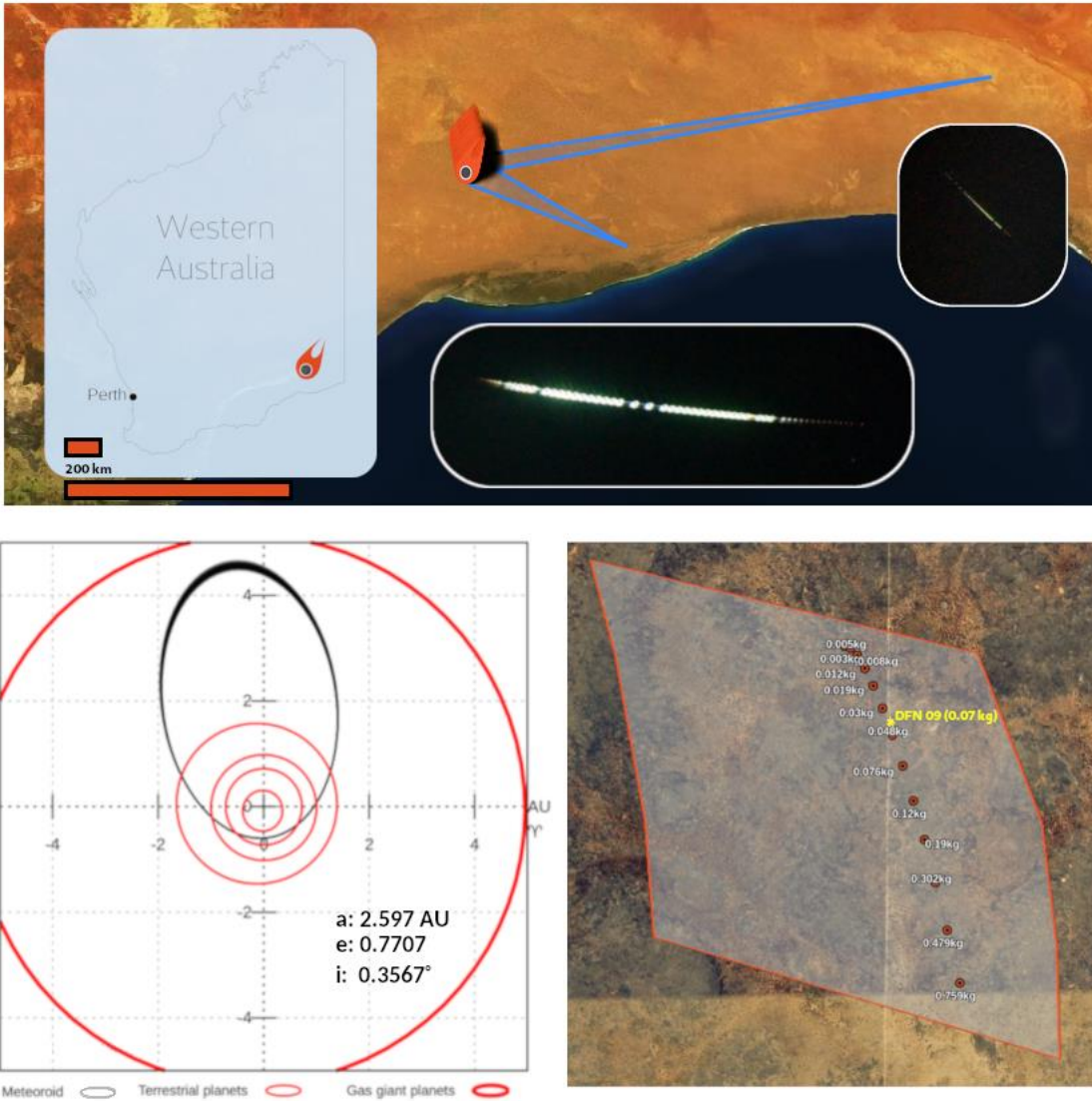


Figure 1. The DFN 09 meteorite fall at Kybo Station, Western Australia. (Clockwise from top) Fireball observations from DFN camera stations at Mundrabilla Station and O'Malley Siding, and their location within WA; The 5 km², 90% certainty searching area (transparent white), the best fit fall line (red markers), and the location of the recovered meteorite (yellow star); Pre-impact orbit for the DFN 09 meteoroid ($a = 2.597 \pm 0.017$ AU, $e = 0.7707 \pm 0.0018$, $i = 0.3567 \pm 0.0053^\circ$, $\omega = 85.862 \pm 0.015^\circ$, $\Omega = 191.90282 \pm 0.00073^\circ$ 39 (1σ uncertainties)).

In the past, recovering meteorites within a predicted area often consisted of 4-6 people walking 5-10 m apart, sweeping the area until the meteorite is found. This was labor intensive and suffered from a relatively low success rate of ~20%, considering how many of our trips (~40) have returned with one of our 8 meteorites. This was partially due to the fact that an entire fall zone could rarely be covered in one trip. This has prompted efforts by multiple groups to recover meteorites using drones and machine learning which massively reduces the time and labor required (Citron et al. 2017; 2021; Zender et al. 2018; AlOwais et al. 2019). Our previous efforts (Anderson et al. 2019; 2020) have shown the ability of our methodology to positively identify already-recovered fresh meteorites, awaiting an opportunity to fully test our searching strategy.

On the night of April 1st 2021 such an opportunity presented itself as a meteorite fell over the Western Nullarbor on the Lintos Paddock of Kybo Station, Western Australia (Figure 1). Unfortunately, only two DFN camera stations, both to the East, were able to capture the event which caused the relatively short fall line to have considerable longitudinal uncertainty, expanding the area with a 90% likelihood of containing the meteorite to a total of 5.1 km² (Figure 1), with a predicted mass of the meteorite being between 150 and 700 g. These fall conditions were promising enough to warrant a fieldtrip to survey the entire fall zone with a drone. The first three days we spent onsite consisted of surveying with a drone, and processing data with our machine learning algorithm. On the fourth and final day we visited meteorite candidates with the drone and in person, and recovered the 70 g meteorite.

2. Methods

2.1 Fireball Observations and Modelling

Although the detailed analysis of the fireball and its orbit will be the subject of a future publication, here we briefly explain the data and methods that allowed us to predict the searching area. These analyses were done prior to the recovery, and have not been revised since. On 1 April 2021, two Desert Fireball Network observatories imaged a bright 3.1 s fireball (Figure 1), which was reported to the DFN team by our automated detection software (Towner et al. 2020), while astrometric calibration is performed following the methodology detailed in Devillepoix et al. (2018).

In total, 78 data points were recorded from just two observatories located at Mundrabilla station and O'Malley siding, 149 km and 471 km from the end point, respectively (Figure 1). The nominal trajectory started at 87 km altitude at 25.4 km/s, and the bolide was observed down to 25 km at 8.4 km/s, on a 64 deg slope. Because of the distance of the viewpoints and a low convergence angle of planes of 28 deg, observation conditions were not ideal, resulting in a poorly constrained trajectory. To estimate the variability in the trajectory, in a similar manner to Devillepoix et al. (2021), we use a Monte Carlo approach, randomising astrometric observations within errors, generating 1000 clones. This analysis shows a 500 m standard deviation on the end point.

For estimating the surviving mass of the object, we use the alpha-beta method from Gritsevich et al. (2012) and Sansom et al. (2019). The noisy velocity data resulted in a poorly constrained surviving mass between 150 and 700 g, assuming a spherical to rounded brick shape, a bulk density of 3.5 g/cm^3 , and a shape change parameter of $2/3$. We model the atmospheric conditions numerically using the Weather Research and Forecasting (WRF) model version 4.0 with the Advanced Research WRF dynamic solver (Skamarock et al. 2019) with 3 model runs starting on 2021-04-01 at 00UT, 06UT, and 12UT, all giving similar models.

Using these atmosphere models, we then propagate bright flight observations to the ground using the dark flight model from Towner et al. (2021). The uncertainty on the mass, and the uncertainty on the positions are the dominant sources of uncertainty. Thanks to the dominant wind being more or less in the same direction as the fireball, the small mass end of the fall line was somewhat compacted. From these simulations, we defined two search areas at different confidence levels: a 90 % confidence level yielding a search area of 5 km^2 (Figure 1), and a 99 % confidence level which gives a significantly larger search area of 8 km^2 .

To compute the pre-encounter orbit, we start by modelling the initial speed with an Extended Kalman Filter/Smother (Sansom et al. 2015), applied to the nominal trajectory. Using the integrator from Jansen-Sturgeon et al. (2019), we then propagate the position of the meteoroid backwards until it is $10\times$ outside the sphere of influence of the Earth-Moon system. The position is then propagated forward to the date of impact, ignoring the influence of the Earth and the Moon. From this point we convert position/velocity to ecliptic orbital elements (J2000). To quantify errors, we repeat this process using Monte Carlo randomization of the initial velocity vector within uncertainties. The resultant pre-impact, orbital parameters for this meteoroid were (Figure 1): Semi-major axis $a = 2.597 \pm 0.017 \text{ AU}$, eccentricity $e = 0.7707 \pm 0.0018$, inclination $i = 0.3567 \pm 0.0053^\circ$, argument of perihelion $\omega = 85.862 \pm 0.015^\circ$, longitude of ascending node $\Omega = 191.90282 \pm 0.00073^\circ$ $39 (1\sigma \text{ uncertainties})$.

2.2 Drone Surveying and Machine Learning

Our field-based work is a continuation of the methodology presented in Anderson et al. (2020). For this trip we used a DJI M300 drone with a Zenmuse P1 camera (44 MP) to survey the 5.1 km^2 fall line at 1.8 mm/pixel with 20% overlap among images in each direction, which took between 2.5 and 3 days to complete. This ground sampling distance would image the meteorite at an apparent size between 15-65 pixels in diameter, due to inherent uncertainties in meteoroid properties during the fall. If the meteorite were predicted to have a significantly larger mass, we would increase the survey height to keep the same apparent size. We processed the data on-site using a desktop computer with an RTX 2080 Ti GPU. Our algorithm could process one flight's worth of images ($\sim 30 \text{ min}$) in approximately 65 min.

Our algorithm works by taking a full 44 MP image and splitting it into tiles ($125 \times 125 \text{ pixels}$) with a 70 pixel overlap in each direction, to ensure the meteorite has a chance to appear fully in at least one tile. Each

tile is fed into the binary image classifier presented in Anderson et al. (2020), constructed using python and keras (Chollet et al. 2015) which scores each tile from 0 (non-meteorite) to 1 (meteorite). To obtain True (meteorite) training data, we relied on our library of meteorite images from previous trips as well as taking new images of the real meteorites: Camel Donga (Eucrite), Mulga North (H6) and Wiluna (H5) (supplied by the Western Australian Museum), at the fall zone. In total, we used 28 individual stones from these 3 meteorite falls, ranging in size from 2 to 14 cm (on their longest dimension). We repeated meteorite image collection each day and for each weather condition that affected illumination. Our True pool was comprised of all the meteorites we imaged on-site and an equal number again sourced from our library, totalling ~100,000 tiles. To create False tiles (non-meteorites) we took images from the surveyed fall zone at random, checked to ensure there were no meteorites in the frame, and split them into tiles. These totalled to more than 1 M depending on the day of the flights being sampled.

To maintain a balanced training set, while also sampling as much of our dataset as possible, we employed what we call ‘rotation training’. To form a training set, we use a constant 80% of our True pool (with 20% kept for validation) and randomly select an equal number from the False pool, then we train for 5 epochs (rounds of training). After this we deselect these False tiles and randomly select a new False set from the pool, then repeat the training process until the model had worked through the False pool twice. The validation set contained only True tiles, so that we could monitor the most important performance metric: meteorite detection chance. We initially trained the model until it rotated through the False pool twice. By the end of training, we achieved a training accuracy of 99.93%, and a validation accuracy (meteorite detection chance) of 91%.

As we predicted on each flight from our first day of surveying, we monitored the quality of the prediction by viewing the distribution of confidence values for a given image, in the form of a histogram (Figure 2). When the distribution for an image resembled that of Figure 2A, we were satisfied, while images that produced a histogram like Figure 2B were flagged as ‘problem images’. We believe they contain physical features from the survey area that were not yet included in the training data, causing the model to infer generously, creating more unnecessary false positives to sort later in the process. We inspected some of these images, split them into tiles and added them to the False pool. We later retrained on this augmented False pool, though only for two epochs and one rotation. Using the retrained model we would re-predict on these problem images, which usually resulted in a more palatable confidence distribution.

We inspected meteorite candidate tiles in 4 stages (Figure 3). The first stages were identified by the model to have a confidence >0.7 , and we inspected them using the 3x3 grid graphical user interface (GUI) described in Anderson et al. (2020), which was ideal for eliminating obvious false positives. The user would be given a 3x3 grid displaying 9 tiles, with some being sourced from the True pool as tests for the user, to characterize their own performance. They identified interesting tiles by typing the corresponding key on the number pad before moving to the next set. Uninteresting tiles were added to the false pool, while those of interest became second stages that were further inspected in a separate image viewer that allowed pan and zoom control over the whole image. If the candidate was still of interest, it became a third stage. To inspect the third stages, we compiled all of their GPS coordinates from one survey flight and planned a new waypoint flight using our DJI Mavic Pro drone and GS Pro app. The Mavic was ideal for this task as it

could lower to ~ 1 m altitude with much lower assumed risk than the more expensive M300. When the autopilot flew the drone to a waypoint (at ~20 m altitude), we paused the mission and manually lowered the drone directly above the candidate, using the original survey image and prediction box as a guide. If the entire team could confidently eliminate that candidate we would remove it, otherwise we took 2-3 pictures for later inspection and proceeded via autopilot to the next waypoint. Since live-feed transmission from the drone provided a lower resolution than its camera, we reviewed the images on our field computer in camp. Any candidates that survived this scrutiny proceeded to the fourth stage: in-person inspection.

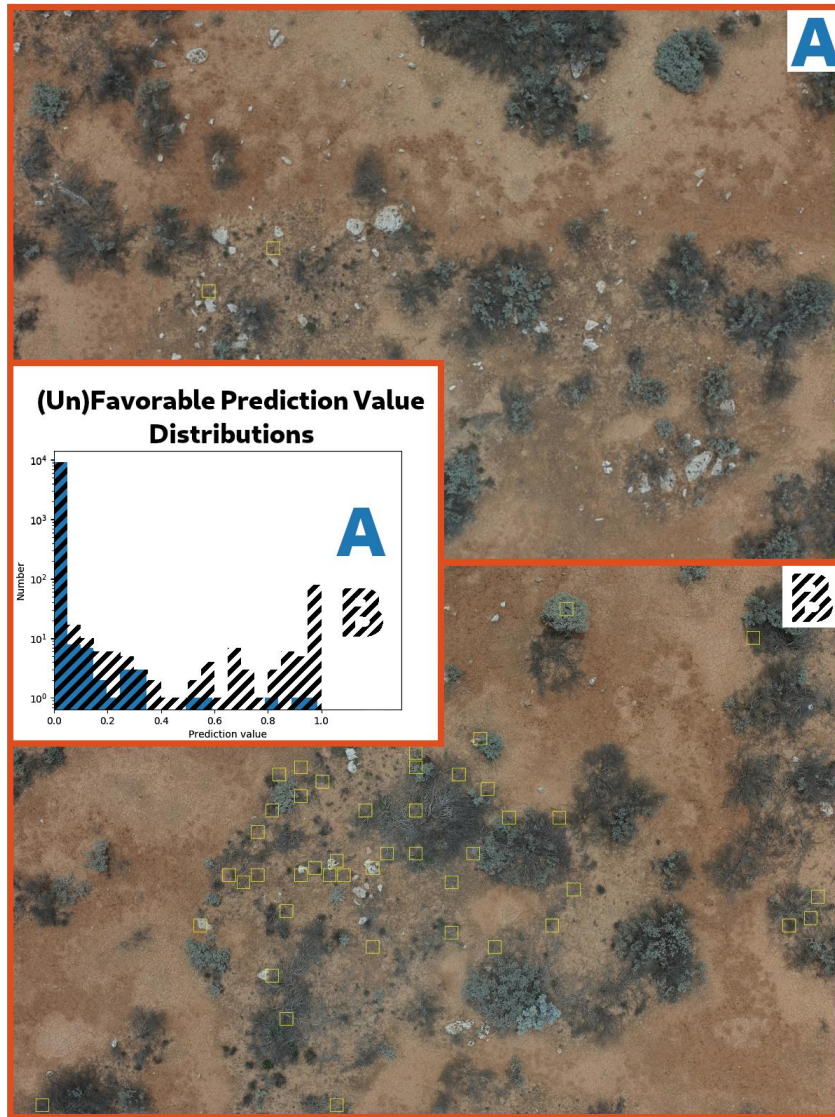


Figure 2. Favorable and unfavorable prediction distributions from two images. Given a 70% confidence threshold, Image/Distribution A will return 3 meteorite candidates, while Distribution B will return >100 candidates. B-like images are later used for retraining. For clarity, the number of detections displayed in image B is capped at 50

3. Results and Discussion

Using the above methodology, we recovered the meteorite <50 m from the best-fit fall line, appearing in the 88th image from the 3rd flight on the 1st day, with an apparent diameter of 27 pixels and a confidence score of 1: a perfect match. Although we have not yet classified the meteorite, its fusion crust (Figure 5) resembles that of other chondrites. It consists of one 70 g piece, approximately 5x4x3 cm, with a preferentially smoothed side. Initial CT scans show chondrules along with a mixture of metal and sulfide grains indicative of an equilibrated chondrite. Analysis of our two thick section samples via Scanning Electron Microscope and Electron Microprobe, which can definitively determine meteorite type, is forthcoming.



Figure 3. The four stage process for eliminating false positives and verifying meteorite candidates. (From Left to Right) 1) Grid GUI. 2) Zoom-pan GUI. 3) Drone visit. 4) In-person visit.

While this meteorite is a great discovery that will hopefully spur the efficient collection of further meteorites, we also made some important discoveries about our methodology. Our data processing rate for both the machine learning algorithm and manual candidate sorting must be improved in the future, or the length of the trips must be extended as the meteorite appeared on the 3rd/43 survey flights. We were able to process 4 flights, totalling 5096 images, which produced 46,501,000 tiles for our algorithm, identifying 56384 first stage candidates (Figure 3). Using our candidate elimination process, we produced 259 second stage candidates, visiting 38 of these as third stages with the Mavic Pro, finally searching for 4 candidates in person (fourth stage). These 4 candidates (including the recovered meteorite) are shown in Figure 5 along with 4 training tiles for comparison. If we were instead required to process all 57,255 images, we would not have been able to complete it onsite, and would have to return on a later trip to follow up on meteorite candidates. That being said, the meteorite was located <50 m from our ‘ideal’ fall line, which is surprising considering the side-to-side uncertainty in the fall line. With this in mind, we may in the future prioritize searching the area immediately around the ideal fall line.



Figure 4. The recovered meteorite as seen in person (top two), and from the survey drone (bottom one). For scale, a 15 cm long felt pen is placed next to the meteorite (top right). The yellow box in the bottom image is 22 cm on one side.

Before we embark on our next searching trip to one of our 35 unvisited, logged meteorite falls, we must alleviate some logistical bottlenecks in our system. This includes locally networking laptops to our main processing computer on site to allow more than one team member to sort meteorite candidates from false positives at a time. For each new fall that we visit we will also bring meteorites to retrain our model on generated, local training data to enhance meteorite detection

chance and mitigate false positives. In lieu of real meteorites, images of black-painted rocks collected on site can also work (Anderson et al. 2020).

Although we recovered the meteorite we did not really train a meteorite detection algorithm, instead, we created an anomaly detector, in this case trained for the Nullarbor. During the course of devising this strategy, we encountered false positives such as tin cans, bottles, snakes, kangaroos, and piles of bones from multiple animals. We also notice that when we predict on survey images taken directly over our campsite, the algorithm ferociously identifies our items and equipment, none of which is represented in the training data. We hope that our findings and methodology prove useful for training neural networks in other low-occurrence or anomaly detection problems, such as wildlife monitoring, or search and rescue.

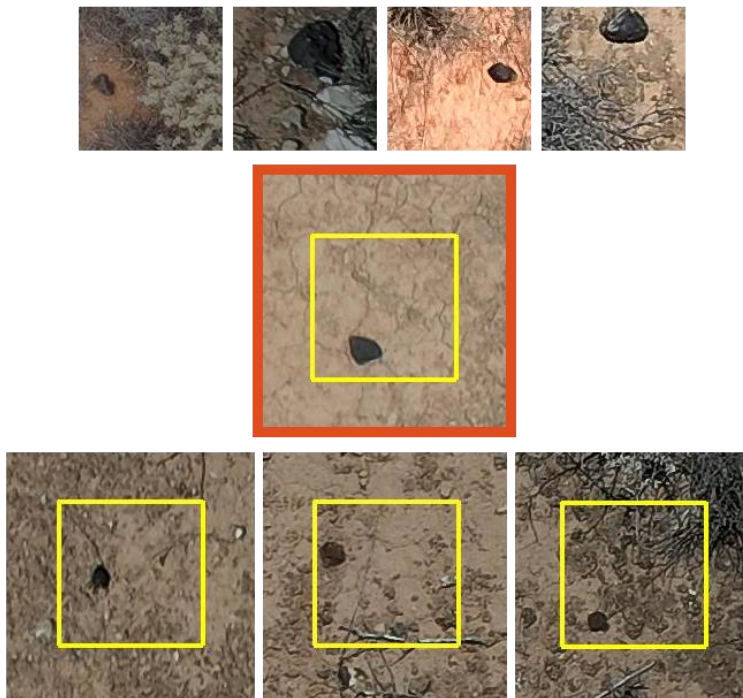


Figure 5. (Top row) 4 meteorites tiles selected from our training data, (Middle) The recovered meteorite, (Bottom) the other 3 meteorite candidates visited for a second time with the Mavic Pro, and later inspected in-person.

Acknowledgements

This work was funded by the Australian Research Council as part of the Australian Discovery Project scheme (DP170102529, DP200102073), and receives institutional support from Curtin University. Fireball data reduction is supported by resources provided by the Pawsey Supercomputing Centre with funding from the Australian Government and the Government of Western Australia. The DFN data reduction pipeline makes intensive use of Astropy, a community-developed core Python package for Astronomy (Price-Whelan et al. 2018). We would like to thank Geoff Deacon and Peter Downes at the Western Australian Museum for loaning us meteorites from the collection.

References

- AlOwais, et al. 2019, *2nd International Conference on Signal Processing and Information Security (ICSPIS)* 1.
- Anderson, S. L. et al. 2019, *50th Annual Lunar and Planetary Science Conference* 2426
- Anderson, S. et al. 2020, *Meteoritics & Planetary Science*, 55(11), 2461-2471.
- Bland, P. A. et al. 2012, *Australian Journal of Earth Sciences* 59(2), 177.
- Chollet, F. et al. 2015, Keras. Available at: <https://github.com/fchollet/keras>.
- Citron, R. I. et al. 2017, *48th Annual Lunar and Planetary Science Conference*, 2528.
- Citron, R. I. et al. 2021. *Meteoritics & Planetary Science*, 56(6), 1073.
- Colas, F. et al. 2015, *Proceedings of the International Meteor Conference*, 37
- Cuzzi, J. N., Dobrovolskis, A. R., & Champney J.M. 1993, *Icarus* 106(1), 102.
- Devillepoix, H. A., et al. 2018, *Meteoritics & Planetary Science*, 53(10), 2212.
- Devillepoix, H. A., et al. 2022, arXiv:2202.06641.
- Devillepoix, H. A., et al. 2020, *Planetary and Space Science*, 191, p.105036.
- Gritsevich, M.I., Stulov, V.P. & Turchak, L.I. 2012, *Cosmic Research*, 50(1) 56.

Howie, R. M., 2017, *Experimental Astronomy* 43(3), 237.

Skamarock, W.C., et al. 2019, *National Center for Atmospheric Research: Boulder, CO, USA*, 145.

Spurný, P., Borovička J., & Šrbený L., 2006 *Proceedings of the International Astronomical Union 2*, S236 121.

Towner, M.C., et al. 2020, *Publications of the Astronomical Society of Australia*, 37.

Towner, M. C. et al. 2021, arXiv:2108.04397

Trigo-Rodríguez, J. et al. 2006, *Astronomy & Geophysics*, 47(6), 6.

Zender, J., et al. 2018, *Proceedings of the International Meteor Conference* 145.

Chapter V: The Murrili (H5) Meteorite

(Published in: *Meteoritics and Planetary Science*, 56(2), pp.241-259)

Mineralogy, petrology, geochemistry, and chronology of the Murrili (H5) meteorite fall: The third recovered fall from the Desert Fireball Network.

S. Anderson^{*1}, G.K. Benedix^{1,2}, L.V. Forman¹, L. Daly^{1,3}, R.C. Greenwood⁴, I.A. Franchi⁴, J.M. Friedrich^{5,6}, R. Macke⁷, S. Wiggins⁸, D. Britt^{8,9}, J.M. Cadogan¹⁰, M.M.M. Meier^{11, 12}, C. Maden¹¹, H. Busemann¹¹, K.C. Welten¹³, M.W. Caffee^{14, 15}, F. Jourdan^{1,16}, C. Mayers^{1,16}, T. Kennedy^{1,16}, B. Godel¹⁷, L. Esteban¹⁷, K. Merigot¹⁸, A.W.R. Bevan², P.A. Bland^{1,2}, J. Paxman¹, M. C. Towner¹, M. Cupak¹, E.K. Sansom¹, R. Howie¹, H. Devillepoix¹, T. Jansen-Sturgeon¹, D. Stuart¹⁹, and D. Strangway¹⁹

¹ *School of Earth & Planetary Sciences, Curtin University, GPO Box U1987, Perth,*

WA 6845, Australia

² *Dept. Earth and Planetary Science, Western Australia Museum, Locked Bag 49, Welshpool, WA 6986, Australia*

³ *School of Geographical and Earth Sciences, University of Glasgow, Glasgow, G12 8QQ, UK.*

⁴ *PSSRI, Open University, Milton Keynes, UK MK7 6AA*

⁵ *Dept. of Chemistry, Ford-ham University, Brooklyn, NY, USA*

⁶ *Dept. Of Earth and Planetary Sciences, American Museum of Natural History, 79th Street at Central Park West, New York, NY 10024 USA*

⁷ *Vatican Observatory V-00120 Vatican City-State*

⁸ *University of Central Florida Department of Physics, 4111 Libra Dr, Orlando FL 32816 USA*

⁹ *Center for Lunar and Asteroid Surface Science, 12354 Research Pkwy, Suite 214, Orlando FL 32826 USA*

¹⁰ *School of Science, UNSW Canberra, BC2610, Canberra, ACT, Australia*

¹¹ *ETH Zurich, Institute for Geochemistry and Petrology, Zurich, Switzerland*

¹² *Naturmuseum St. Gallen, St. Gallen, Switzerland*

¹³ *Space Sciences Laboratory, University of California, Berkeley, CA 94720, USA*

¹⁴ *Dept. of Physics and Astronomy, Purdue University, West Lafayette, IN 47907, USA*

¹⁵ *Dept. Of Earth, Atmospheric and Planetary Sciences, Purdue University, West Lafayette, IN 47907, USA*

¹⁶ *Western Australian Argon Isotope Facility, JdL Centre, Curtin University, GPO Box U1987, Perth, WA 6845, Australia*

¹⁷ *CSIRO Earth Sci. and Resource Engineering, ARRC, Kensington, WA, Australia*

¹⁸ *John de Laeter Centre, Bldg 301, Curtin University, Bentley, WA, 6845, Australia*

¹⁹ *23 Main Street, Port Augusta, South Australia*

*Corresponding author: Email: seamus.anderson@postgrad.curtin.edu.au

ABSTRACT

Murrili, the third meteorite recovered by the Desert Fireball Network, is analyzed using mineralogy, oxygen isotopes, bulk chemistry, physical properties, noble gases, and cosmogenic radionuclides. The modal mineralogy, bulk chemistry, magnetic susceptibility, physical properties and oxygen isotopes of Murrili point to it being an H5 ordinary chondrite. It is heterogeneously shocked (S2-S5), depending on the method used to determine it, although Murrili is not obviously brecciated in texture. Cosmogenic radionuclides yield a cosmic ray exposure age of 6-8 Ma, and a pre-atmospheric meteoroid size of 15-20 cm in radius. Murrili's fall and subsequent month-long embedment into the salt lake Kati Thanda, significantly altered the whole rock, evident in its Mössbauer spectra, and visual inspection of cut sections. Murrili may have experienced minor, but subsequent impacts after its formation 4475.3 ± 2.3 Ma, which left it heterogeneously shocked.

INTRODUCTION

Meteorites are an important resource for understanding the origin and evolution of our solar system and come to us for free. Pieces of this precious material shower Earth every year but falls are rarely witnessed. This means that important context is missing from the meteoritic record; that is, what is the general geologic origin of these precious rocks? The Desert Fireball Network (Bland et al. 2012; Howie et al. 2017) provides a basic framework to determine that context for these rocks, allowing their source region in the solar system to be constrained. Every meteorite that has characterized orbit provides unique information that can be used to better interpret the structure of the solar system. Murrili (pronounced moo-ree-lee) is the 3rd meteorite recovered by the Desert Fireball Network (Bland et al. 2016), after Bunburra Rockhole (anomalous achondrite), and Mason Gully (H5) (Spurný et al. 2012; Dyl et al. 2016).

Sansom et al. (2020) reported observations of a fireball lasting 6.1 s, entering the atmosphere with a speed

of $\sim 13.7 \text{ km s}^{-1}$ at an altitude of 85 km, before slowing to $\sim 3 \text{ km s}^{-1}$ at an altitude of 18 km at the end of its luminous flight phase. From these observations they determined that the meteoroid's pre-entry orbit had a semi major axis of $2.521 \pm 0.075 \text{ AU}$, an eccentricity of 0.609 ± 0.012 and an inclination of $3.32 \pm 0.060 \text{ deg}$. Orbits like this, with low inclination and near the 3:1 mean-motion resonance with Jupiter, are not uncommon for other H5 chondrites with determined orbits (Jenniskens 2013; Meier 2017).

The meteorite fell in Kati Thanda (Lake Eyre) National Park, South Australia, the land of the Arabana people, on 27 November 2015. The rock specifically fell into South Lake Eyre, in a region referred to as 'Murrili' by the Arabana peoples. Kati Thanda (Lake Eyre) is one of the largest landlocked lakes ($>9,500 \text{ km}^2$) in the world. The lake rarely fills completely, but even during dry seasons there is usually some water remaining in smaller sub-lakes. The surface is comprised of a thin layer of halite and gypsum salts on top of brine-saturated, fossiliferous, thick clay mud (Habeck-Fardy and Nanson 2014).

This fall site posed a serious challenge for recovery. The calculated fall site was 6 km from the nearest "shore" (Figure 1). Initial reconnaissance from light-aircraft identified what appeared to be an impact feature in the surface of the lake close to the predicted fall site (Figure 1 inset). Our expedition to recover the meteorite was guided by members of the Arabana people, the traditional custodians of the land. In the time between the fall and the expedition (roughly 1 month), rain had obscured the original impact site. However, searching on foot and with quad bikes, in tandem with aerial and drone surveys, led to the successful recovery of the stone from a depth of 43 cm in the lake. It was recovered on 31st December 2015, $<50 \text{ m}$ away from the predicted fall line (Sansom et al. 2020).

Here, we present details of the classification, physical attributes, chronology and geochemistry of this meteorite.

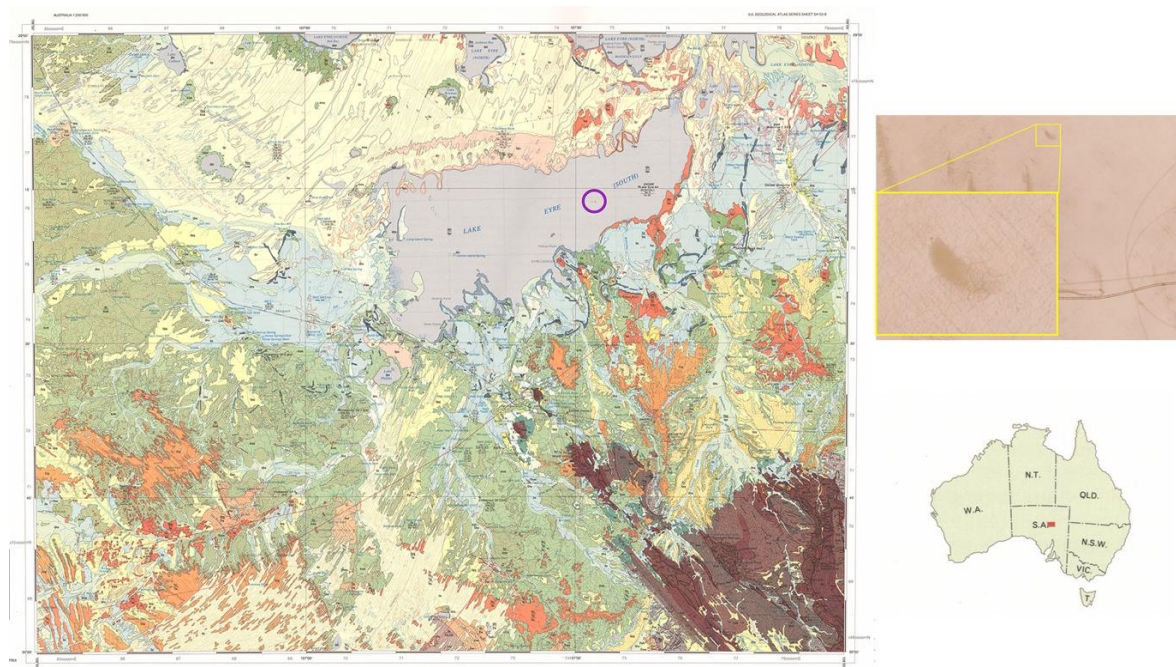


Figure 1. Excerpted from the Curdimurka map sheet of the Kati Thanda area of South Australia (Krieg et al., 1992). Purple circle shows where Murrili was recovered. Inset shows aerial photo of the lake showing visible 'impact' made by the meteorite.

SAMPLE ALLOCATION AND ANALYTICAL METHODS

A single, fusion-crust, heart-shaped, stone, measuring $\sim 13 \times 7 \times 6$ cm, weighing 1.68 kg, was retrieved (Figures 2 and 3). From this main mass, two small wedges and a thin slab were cut for examination and analyses (Figure 4). The cut surfaces reveal extensive alteration, due to the interaction with the salty lake clays, despite there being no obvious broken surfaces. Samples of altered and unaltered meteorite were distributed to a consortium of international researchers. Details of the allocated materials and methods are summarized in Table 1. We examined both altered and unaltered regions using optical microscopy and computed tomography. Various laboratories analyzed wedges, chips and powders for oxygen isotope and bulk composition, cosmogenic nuclides, porosity/density, and Mössbauer spectroscopy. Details of each method are described below.



Figure 2. Full mass of the Murrili meteorite, just after recovery from Lake Eyre South

Optical Mineralogy

We examined two thin sections of Murrili using a Nikon Eclipse LV100 POL microscope at Curtin University in both transmitted (plane and polarised) and reflected light.

Computed Tomography

Two separate labs carried out X-ray Computed Tomography (CT)– CSIRO (Perth, Western Australia) and the American Museum of Natural History (New York, USA).

CSIRO – The sample was scanned in 3D by X-ray computed tomography (CT) using a Siemens SOMATOM definition AS medical scanner installed at the Australian Resources Research Centre (ARRC, Kensington, Western-Australia) allowing the rapid 3D scanning of drill cores. The instrument was calibrated using air as well as a set of five in-house rock standards of known density which are suitable for mineral resources applications (standards have densities varying from 2.7 to 4.3 g/cm³). The energy of the beam was set-up to have maximal phase contrast between the different minerals of interest (accelerating voltage of 140 kV and x-ray beam current of 1000 mA). The voxel size (a pixel in 3D) for this overview CT scan was 220 x 220 x 100 μ m. The voxel size is dictated by the size of the sample – in this case the large meteorite meant that the overall resolution was relatively low. The XCT data was processed and analysed using workflows developed across scale for mineral resource applications (Godel et al. 2006; Godel 2013).



Figure 3. Full mass of the Murrili meteorite – ruler scale is cm. Inset shows close up of fusion crust texture – cracks are infilled with what is most likely lake sediments.

Fordham/AMNH– X-ray microtomography data was collected on one of the smaller wedges (a ~50g pyramid-shaped chunk of Murrili) at the American Museum of Natural History using a GE phoenix v|tome|x s240 μ CT system operating with a polychromatic x-ray tube. Data was collected at several resolutions. The entire wedge was imaged at a resolution of $50.5 \mu\text{m}^3/\text{voxel}$ edge. A subsection was imaged at a resolution of $10.0 \mu\text{m}^3/\text{voxel}$. The latter conditions are known to be adequate for observing morphology and size distributions of metal and sulfide grains in ordinary and other chondrites as well as examining inter-granular porosity structures in partially compacted samples (Friedrich et al. 2008; 2013; 2017). From the 3D datasets, features can be visibly identified and digitally isolated for quantitative examination, and 2D slices (tomograms) can be extracted. Typical tomographic slices of this volume are shown in Figure 5.

We used the method presented in Friedrich et al. (2008; 2013) to quantify the magnitude of foliation of metal grains within Murrili. Our method produces a numerical value for the strength factor, C (Woodcock 1977; Woodcock and Naylor, 1983). The higher the numerical strength factor, the more pronounced the common orientation of the metal grains and the greater the foliation.

Table 1. Sample allocations and methods used in this study, listed by institution.

Method	Institution(s)	Sample type
Optical Microscopy (Shock Analysis)	Western Australian Museum / Curtin University	Thin Section
X-ray Computed Tomography (Density and Porosity)	CSIRO – Kensington / American Museum of Natural History	Whole Rock / 50 g Wedge
Gas Pycnometry (Density and Porosity)	Vatican Observatory / University of Central Florida (UCF)	50 g Wedge (same sample used for CT)
Scanning Electron Microscopy/Electron Microprobe Microanalysis (Modal Mineralogy and Mineral Composition)	Curtin University / University of Western Australia	Thick Section
Laser-assisted Fluorination (Oxygen Isotope Analysis)	Open University	30-400 g chips
Inductively Coupled Mass Spectrometry (Bulk Chemistry)	Fordham University	4 x 120 mg chips
Mössbauer Spectroscopy (Terrestrial Alteration)	University of New South Wales – Canberra	300 mg powder
Noble Gas Analysis (Cosmic Ray Exposure age (CRE) and 74hour gl size)	Swiss Federal Institute of Technology – Zurich	100 mg chip
Cosmogenic Radionuclides (CRE and meteoroid size)	University of California – Berkeley	1.1 g chip
⁴⁰ Ar- ³⁹ Ar Dating / ³⁸ Ar CRE (Impact Heating Age/ CRE)	Curtin University	2 g (fragments)

Density, volume, porosity, and magnetic susceptibility

We measured bulk volume and density on the same 50g piece that was CT scanned by the AMNH (see above), using a NextEngine ScannerHDPro laser scanner at high resolution (Macke et al. 2015) located at UCF, with a digital scan analysis performed at the Vatican Observatory. The laser scanner produced a 3-dimensional computer model of the meteorite, from which the volume enclosed by the outer surface could be calculated in the software.

Grain volume and density was determined using helium ideal gas pycnometry using a Quantachrome Ultrapyc 1200e following the procedure outlined in Macke (2010). Porosity is calculated from these two densities: $P = 1 - \rho_{\text{bulk}}/\rho_{\text{grain}}$, where ρ_{bulk} is the density of the whole rock, including pore spaces, while ρ_{grain} excludes pore spaces. Magnetic susceptibility was measured using an SM-30 handheld device, with volumetric and shape corrections according to Gattacceca et al. (2004) and Macke (2010).

Mineral compositions and modes

We determined modal mineralogy on a thick section of Murrili using a Tescan Integrated Mineral Analyzer

(TIMA) housed in the Digital Mineralogy Hub (DMH) of the John de Laeter Centre at Curtin University. The automated modal analysis of the TIMA instrument was not optimized for meteorites, therefore, we generated mineral modes from element maps, using the methods outlined in Ford et al. (2008).

Mineral compositions were determined on the same thick section. Silicon, Mg, Ti, Cr, Fe, Ca, Al, and Na were measured in olivine, orthopyroxene, clinopyroxene, and chromite, with a JEOL 8530F electron microprobe based in the Centre of Microscopy, Characterization and Analysis (CMCA) at the University of Western Australia. The probe was operated with an accelerating voltage of 20kV and beam current of 20nA. Well-known standards were used for calibration of the elements.

Oxygen Isotopic Analysis

Oxygen isotope analysis was carried out at the Open University using an infrared laser-assisted fluorination system (Miller et al. 1999; Greenwood et al. 2017). All analyses were obtained on approximately 2 mg aliquots drawn from a larger homogenized sample powder, with a total mass of approximately 100 mg, prepared by crushing clean, interior, whole rock chips. Oxygen was released from the sample by heating in the presence of BrF₅. After fluorination, the oxygen gas released was purified by passing it through two cryogenic nitrogen traps and over a bed of heated KBr. Oxygen gas was analyzed using a MAT 253 dual inlet mass spectrometer. Overall system precision, as defined by replicate analyses of our internal obsidian standard is: ±0.053‰ for δ¹⁷O; ±0.095‰ for δ¹⁸O; ±0.018‰ for Δ¹⁷O (2σ) (Starkey et al. 2016).

We report oxygen isotopic analyses in standard δ notation, where δ¹⁸O has been calculated as: δ¹⁸O = [(¹⁸O/¹⁶O)_{sample} / (¹⁸O/¹⁶O)_{ref} - 1] × 1000 (‰) and similarly for δ¹⁷O using the ¹⁷O/¹⁶O ratio (where ref is VSMOW: Vienna Standard Mean Ocean Water). For comparison with the ordinary chondrite analyses of Clayton et al. (1991) Δ¹⁷O, which represents the deviation from the terrestrial fractionation line, has been calculated as: Δ¹⁷O = δ¹⁷O - 0.52 × δ¹⁸O.

Bulk composition

We determined bulk trace and major element compositions using Inductively Coupled Mass Spectrometry (ICPMS) at Fordham University, with methods described in Friedrich et al. (2003) and Wolf et al. (2012). We analyzed four individual chips of Murrili (126.1, 136.6, 120.5, 114.8 mg). In each aliquot, the material used for the analyses was a combination of altered and unaltered material, since the alteration was too intermingled to separate. We used a combination of concentrated HF and HNO₃ in a high-pressure microwave digestion system at 185 °C, for 15 min, to initially dissolve each chip, then we took the resulting solution to incipient dryness on a hotplate. This was followed by treatment with concentrated HClO₄ and a further drying to completely dissolve the samples. The dissolved samples were taken up in 1% HNO₃ and analyzed with a ThermoElemental X-Series II ICPMS with the methods outlined in Friedrich et al. (2003) for trace elements and Wolf et al. (2012) for major and minor elements. In addition to the Murrili samples, a procedural blank and the Allende Standard Reference Meteorite were simultaneously analyzed for calibration purposes.

Mössbauer Spectroscopy

In the scope of meteorite analyses, we use Mössbauer spectroscopy to measure the oxidation states of Fe in the sample. In ordinary chondrites, Fe⁰ and Fe²⁺ are of extraterrestrial origin, occurring in metallic iron,

troilite and silicates, while Fe^{3+} in iron oxides forms solely due to terrestrial contamination, for ordinary chondrites. By measuring the relative abundance of Fe^{3+} to all other iron oxidation species in the sample, we can obtain a quantitative measurement of the meteorite's weathering state (Bland et al. 1998).

The Mössbauer spectra were obtained at room temperature using a standard transmission spectrometer with a $^{57}\text{CoRh}$ source. The spectrometer's drive system was calibrated using a $6\mu\text{m}$ $\alpha\text{-Fe}$ foil. The spectra were fitted using a non-linear, least-squares, full-hamiltonian method.

Noble gas analyses

He and Ne were measured on an in-house-built sector field noble gas mass spectrometer at ETH Zurich, according to a protocol most recently described in detail by Meier et al. (2017). Two fusion-crust-free samples with masses of 45.9 and 22.6 mg, respectively, were wrapped in Al foil, loaded into the sample holder and pumped to $\sim 10^{-10}$ mbar while being heated to 80°C for one week to desorb atmospheric gases. For gas extraction, the samples were dropped into a crucible and heated in a single step to ca. 1700°C by electron bombardment. Both sample analyses were bracketed by blank (empty Al foil) analyses. For analysis of all He and Ne isotopes as well as the potentially interfering species ($\text{H}_2^{18}\text{O}^+$, $^{12}\text{C}^{16}\text{O}_2^{++}$, $\text{H}^{35,37}\text{Cl}^+$), we separated He, Ne from Ar using a cryotrap cooled with liquid nitrogen. Interferences were negligible, and blank levels contributed $<0.01\%$ for He and $<5\%$ for Ne isotopes. For Ar, the blank contribution was inexplicably high ($>50\%$), which eventually led us to discard the Ar data completely, for this approach. Noble gas concentrations and ratios were then used to determine the cosmogenic (cosmic-ray produced) ^3He and ^{21}Ne concentrations using multi-component deconvolutions with radiogenic (for He only), cosmogenic and phase Q (or air) endmembers. We used both model-based ("L&M09", Leya & Masarik, 2009) and empirically determined (Dalcher et al., 2013) production rates to calculate the cosmic-ray exposure ages from the cosmogenic concentrations. Uranium-Thorium-Helium radiogenic gas retention ages (^4He was corrected for cosmogenic contributions) were calculated based on assumed Th, U concentrations of 42 ppb and 13 ppb, respectively, typical for H chondrites (Kallemeyn et al. 1989).

Cosmogenic Radionuclide analysis

We used a chip of ~ 1.13 g for analysis of the cosmogenic radionuclides ^{10}Be (half-life= 1.36×10^6 y), ^{26}Al (7.05×10^5 y) and ^{36}Cl (3.01×10^5 y). We crushed the sample in an agate mortar and separated the magnetic (metal) from the non-magnetic (stone) fraction. The magnetic fraction was purified by ultrasonic agitation in 0.2N HCl at room temperature, to remove attached troilite. After rinsing the metal four times with MilliQ water and once with ethanol, the separation yielded 146 mg of relatively clean metal, corresponding to 13 wt% bulk metal. The metal fraction was further purified by ultrasonic agitation in concentrated, room temperature HF for 15 min to dissolve attached silicates, yielding ~ 140 mg of purified metal. We dissolved ~ 66 mg of the purified metal in 1.5N HNO_3 along with a carrier solution containing 1.47 mg Be, 1.64 mg Al, and 4.85 mg Cl. After dissolution we took a small aliquot ($\sim 3.6\%$) of the dissolved sample for chemical analysis (Mg, Fe, Co, Ni) by ICP-OES and used the remaining solution for radionuclide separation following procedures described in Welten et al. (2011). The chemical analysis of the dissolved sample yields a Mg concentration of 0.074 wt%, indicating that the purified metal contained $\sim 0.4\text{-}0.5$ wt% of silicate contamination.

After separating and purifying the Be, Al and Cl fractions, the $^{10}\text{Be}/\text{Be}$, $^{26}\text{Al}/\text{Al}$ and $^{36}\text{Cl}/\text{Cl}$ ratios of these samples were measured by accelerator mass spectrometry (AMS) at Purdue University's PRIME Lab (Sharma et al. 2000). The measured ratios were normalized to those of well-known AMS standards (Nishiizumi 2004; Nishiizumi et al. 2007; Sharma et al. 1990) and converted to concentrations in disintegrations per minute per kg (dpm/kg). The ^{10}Be and ^{26}Al concentrations in the metal fraction were corrected for small contributions of ^{10}Be (1%) and ^{26}Al (5.5%) from the stone fraction.

We also dissolved 106 mg of the stone fraction of Murrili, along with 2.93 mg of Be carrier and 3.68 mg of Cl carrier, in concentrated HF/HNO₃ by heating the mixture for 24 h inside a Parr 77horou digestion bomb at 125 °C. After cooling off, we separated the Cl fraction as AgCl, and removed Si as SiF₄ by repeated fuming with ~0.5 ml concentrated HClO₄. The residue was dissolved in diluted HCl and a small aliquot was taken for chemical analysis by ICP-OES. After adding ~5.2 mg of additional Al carrier to the remaining solution, we then separated and purified the Be and Al fractions and measured the isotopic ratios of the Be, Al and Cl fractions by AMS. Results of the chemical analysis and AMS measurements of both the metal and stone fraction are shown in Tables 2 and 3, respectively.

³⁸Ar age

Cosmic ray exposure (CRE) ages were also calculated based on the spallation of Ca to ³⁸Ar due to the interaction of cosmic rays with the sample. To determine the amount of ³⁸Ar in the sample per gram of Ca (Hennessy and Turner; 1980) we use the cosmochron approach which consists of an isotope correlation diagram with ³⁸Ar/³⁶Ar vs. ³⁷Ar/³⁶Ar (Levine et al., 2007). The exposure age is calculated assuming a nominal production rate of ³⁸Ar from spallation of Ca of 1.81 x 10⁻⁸cm³ STP ³⁸Ar/gram of Ca/Ma for plagioclase, 2.30 x 10⁻⁸cm³ STP ³⁸Ar/gram of Ca/Ma for pyroxene and a half-way value of 2.05 x 10⁻⁸cm³ STP ³⁸Ar/gram of Ca/Ma for matrix which comprises similar quantities of comminuted plagioclase and pyroxene, best suited for the asteroid belt (Eugster and Michel, 1995; Korochantseva et al. 2005). As plagioclase has very low concentrations of Fe and Ti and as the ³⁸Ar_C production rate from Ca is ca. 90 and 10 times that of Fe and Ti, respectively, the contribution of the latter elements to the total ³⁸Ar_C is negligible (Turner et al., 2013). For a full description of the approach, see Kennedy et al. (2013).

⁴⁰Ar-³⁹Ar ages

To determine the Ar-Ar age of Murrili, we analyzed 2g of material without fusion crust, which we prepared by disaggregating a larger slice to extract fragments of mineral (pyroxene and plagioclase) as well as whole chondrules. Most of our results are derived from pyroxene grains.

Samples were irradiated for 40 hours and subsequently analyzed at the Western Australian Argon Isotope Facility (WAAIF) using a MAP 215-50 Mass Spectrometer equipped with a 10.4 μm CO₂ laser for stepped heating for 60 seconds, in accordance with Jourdan et al. (2020).

RESULTS

In this section we present the results of each previously listed method. We start with physical properties, proceeding to chemistry and finishing with chronology. In the discussion section we combine these results to weave together the history of the Murrili meteorite and its subsequent fall to Earth.

Physical Properties

As discussed in the introduction, Murrili consists of a single stone roughly 15 cm in longest dimension. It is heart-shaped with a continuous fusion crust, which shows small cracks filled with terrestrially weathered material (Figures 3 and 4). The exterior also shows several large indentations, but there are no obvious broken surfaces.

Visual inspection of the cut surfaces of Murrili shows a heavily altered interior with a heterogeneously distributed reddish stain in the hand specimen (Figure 4). The whole rock CT scan does not reveal obvious

boundaries that would indicate brecciation, although a brecciated texture cannot be entirely ruled out (Figure 5). Cracks are evident cutting through the fusion crust and running across whole slices. The weathering, however, is not linearly distributed away from such cracks and the alteration does not seem to affect metal uniformly, as grains both within and away from altered areas appear to be unoxidized (Figure 4).



Figure 4. The Wedge used for He pycnometry, magnetic susceptibility and μ CT scans.

Alteration

Both of the ^{57}Fe Mössbauer spectra for the ‘unaltered’ and ‘altered’ materials in Murrili spectra are well-fitted with five components. The paramagnetic doublets of olivine and pyroxene account for 70 to 76 % of the spectral area. Troilite (FeS), Fe^{3+} and FeNi metal comprise the remainder of the spectra. The most prominent difference between the two spectra is the relative spectral area contribution of the paramagnetic Fe^{3+} component. The ‘unaltered’ sample has an Fe^{3+} area of slightly more than 3 % whereas the ‘altered’ sample has around 12% relative area. This dramatic increase in the relative amount of the paramagnetic Fe^{3+} component comes at the expense of the olivine, pyroxene and metal components; the relative amount of troilite remains constant. This increase in Fe^{3+} points to an aggressive weathering process (Bland et al. 1998), in this case likely related to 30-day storage in warm, brine-saturated mud at the Lake Eyre fall site.

Another indicator of aggressive weathering may be drawn from the relative area of the silicates (olivine and pyroxene) compared to the metal. The ‘unaltered’ spectrum has 76.1 ± 6 % relative area for the silicates and 6.8 ± 4 % FeNi metal, yielding a silicate to metal ratio of 11.2 ± 8 . For the ‘altered’ spectrum these numbers are 70.0 ± 6 %, 4.6 ± 4 % and 15.2 ± 14 , respectively. The olivine to pyroxene ratios are 1.59 and 1.57 for the ‘altered’ and ‘unaltered’ spectra, respectively, putting Murrili firmly in the H-classification according to the work of Wolf et al. 2012. However, the classification work of Verma^[10] suggests that our silicon content should be accompanied by about twice as much FeNi metal than we measured, to place Murrili in the ‘H’ region. We^[10] suggest^[10] that aggressive weathering of the FeNi component produced the paramagnetic ferric component which is possibly goethite or akaganéite.

Density and Porosity

We determined bulk density and porosity for Murrili, using both computed tomography and helium pycnometry. Course-resolution (220x220x100 $\mu\text{m}/\text{voxel}$) CT gave a bulk volume of 467.5 cm^3 for the whole rock, which combined with the total mass of Murrili (1.68 kg), yielded a bulk density of 3.6 g cm^{-3} . Although the course-resolution CT was not detailed enough to resolve micro-cracks to obtain a high-fidelity value for porosity, we were able to deduce a porosity of 3.4%. CT imaging at the highest resolution in this study (10.0 $\mu\text{m}^3/\text{voxel}$) did not reveal any visible porosity, which suggests that the main source of Murrili's porosity is due to microcracks rather than larger vugs or intergranular porosity (Friedrich and Rivers 2013). The bulk density recorded by helium pycnometry of the 50 g wedge is $3.47 \pm 0.01 \text{ g cm}^{-3}$, consistent with the results from the CT imaging and data extraction. Helium pycnometry also yielded a value of $3.59 \pm 0.01 \text{ g cm}^{-3}$ for grain density, which is a measure of the rock's density excluding interior void spaces. Combining bulk and grain density from helium pycnometry allowed us to determine a porosity of $3.4 \pm 0.4\%$ for Murrili.

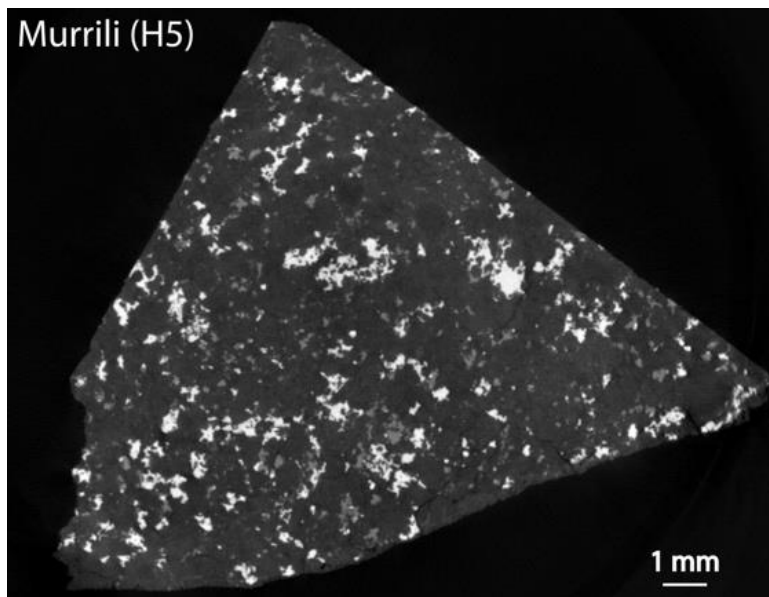


Figure 5. Typical x-ray μCT tomogram or “slice” of the middle portion of a ~50g wedge (Figure 4a) of the Murrili H chondrite collected at a resolution of 10.0 $\mu\text{m}/\text{voxel}$. The higher the greyscale intensity, the higher the density of the material. The brightest material is reduced Fe-Ni metal. The slightly darker spots are troilite. The silicates, making up the majority of the meteorite, are the medium grey material. The black area around the rock is air. Occasional round silicate shapes (chondrules), often outlined by troilite, can be distinguished perceived in the tomogram. The area toward the bottom is a fusion crusted surface. The straight surfaces of the rock are saw-cut surfaces.

Shock

The shock state of Murrili differs significantly between the results of the optical microscopy and fine-resolution CT. We investigated two thin sections, taken from both altered and unaltered regions, using optical microscopy. We examined 25 olivine grains in both thin sections, and found their extinction features to be straight to slightly undulatory, indicating a low, S2 shock state (Stöffler et al., 1991, 2018).

Using reflected light microscopy, we also examined the Fe,Ni metal and sulfide in the thin sections. The

metallography is typical of an undisturbed, slowly cooled ordinary chondrite. In a polished mount briefly etched with a 5% Nital solution, kamacite (bcc Fe,Ni metal) shows a single set of slightly annealed and dilated Neumann bands as the only indication of mild (<130 kb), mechanical shock-loading at low temperature. Many grains of taenite (fcc Fe,Ni metal) are polycrystalline, frequently comprising up to three juxtaposed taenite crystallites each delineated by tetrataenite rims and cloudy etching zones. Large grains (several hundred μm) of zoned taenite show internal decomposition to plessite (kamacite + taenite). Troilite (FeS) is generally fractured and, under crossed polarized reflected light, shows undulose, feathery, extinction. This feature is also typical of low-temperature, mechanical strain. No grains of native copper appeared in our analysis; however, grains of chromite are abundant. In terms of shock level, the metallographic features observed in the opaque phases in Murrili would equate to undulose extinction in the ferro-magnesian silicates. In the polished mount examined, no indication of foliation, or any other directional fabric was observed in the metallic phases.

The high-resolution CT images taken from the 50 g wedge show significant metal-grain foliation. The collective orientation of metal grains observed in the 50 g wedge have a foliation strength factor of 0.75 (Figure 6), which is consistent for a chondrite having experienced significant (S4-S5) shock-related compaction, contrary to the thin section microscopy analysis.

It is important to note that the two thin sections were taken from a separate area than the 50 g wedge. Upon further inspection of the course-resolution CT of the whole rock, we do not see evidence of metal foliation in the thin section-sampled region, while the wedge's original area does reveal foliation. This non-uniformity in the foliation texture, along with low-shock features in the thin sections may suggest a heterogenous shock state across Murrili.

Murrili - collective metal grain orientations

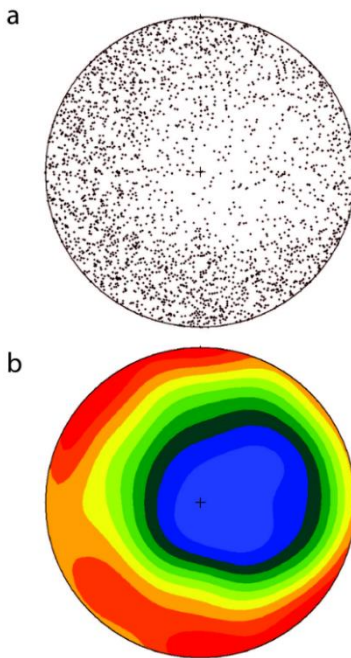


Figure 6. a) Orientation of reduced metal grains and b) stereogram density diagram for the Murrili H chondrite. These are lower hemisphere equal area projections. The orientation strength factor (see text) of 0.755 suggests Murrili experienced significant impact-related compaction.

Modal Mineralogy

The mineralogy of Murrili is typical for ordinary chondrites, being dominated by olivine and pyroxene. The backscattered-electron (BSE) image and elementally-derived mineral maps of Murrili's thick section are shown in Figure 7. The modal mineralogy of Murrili is plotted in Figure 8, shown with published values for H Chondrite meteorites (McSween et al. 1991; Dunn et al. 2010), for comparison. Relative to other H chondrites, there is a slight increase in orthopyroxene relative to olivine which is generally associated with reduction processes. However, there is a coupled decrease in the abundance of plagioclase. This variation could be related to the distribution of Ca in the sample.

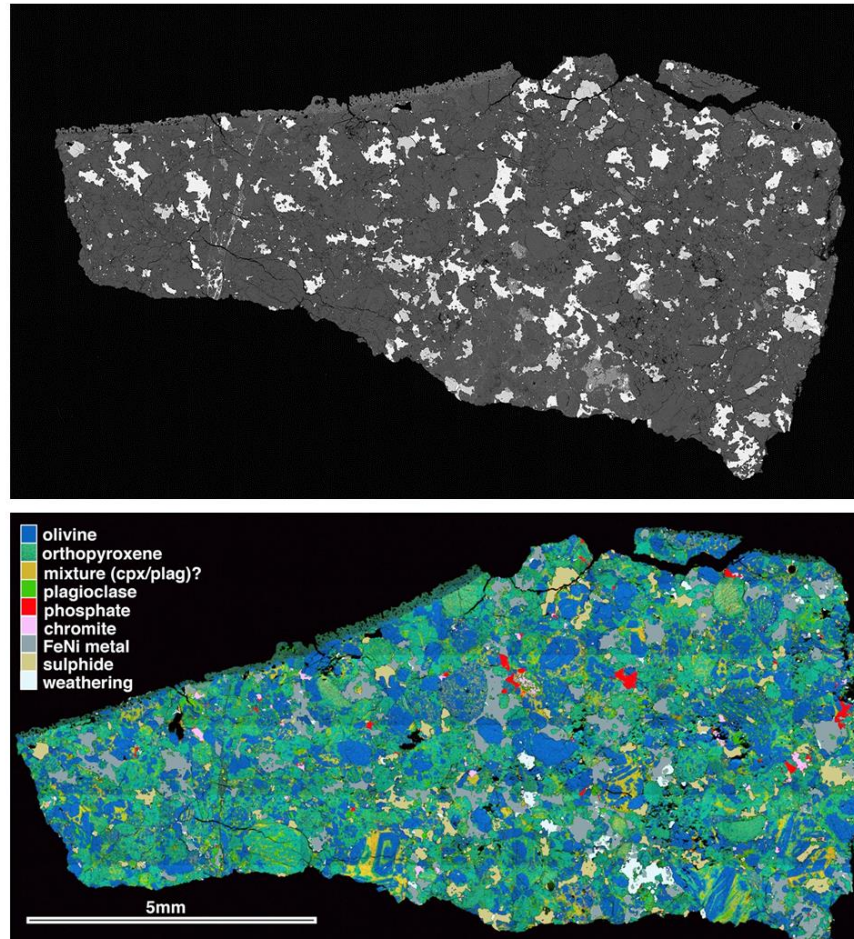


Figure 7. Images of thin section TS1 of Murrili. (a) Backscattered-electron (BSE) image; (b) Chemistry map created by combining maps for 7 elements – Ca, Si, Mg, Fe, Ni, S, and Cr. This map distinguishes 9 distinct chemistries that are tied to the mineralogies listed in the legend. CPX/Plag mix is a very fine-grained mixture of these two phases and are not distinguished at the resolution of the map. Scale bar is the same for both images.

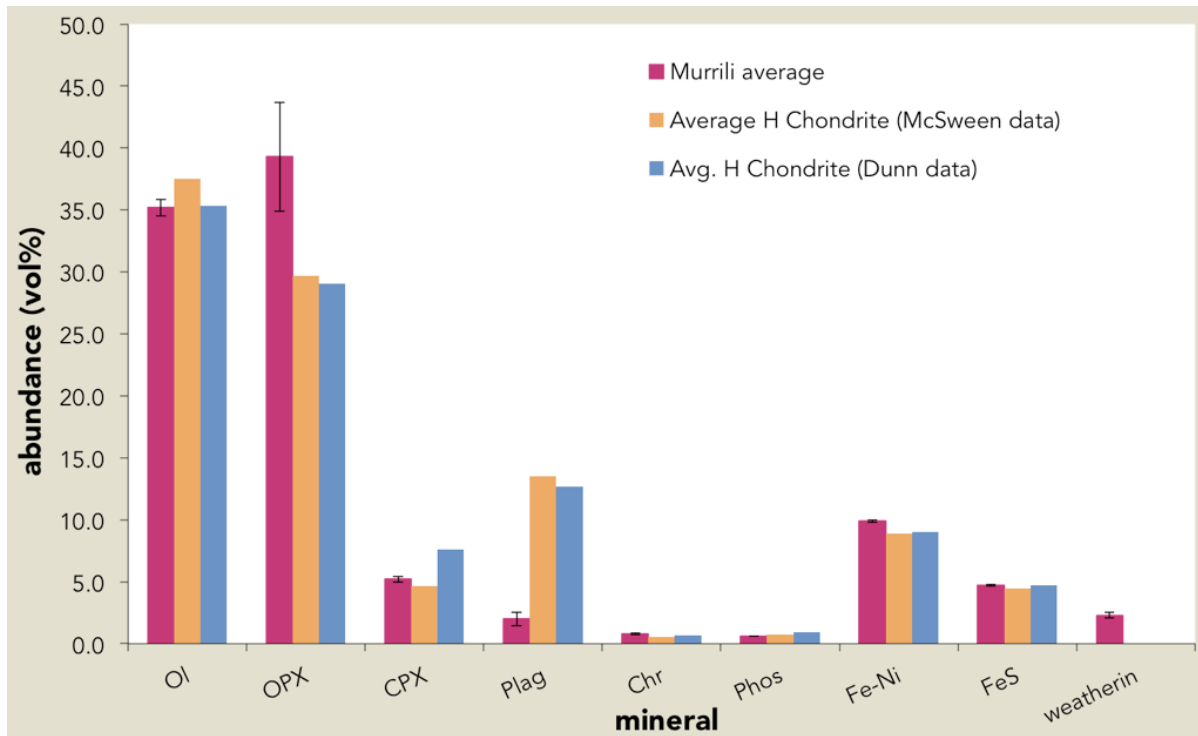


Figure 8 – Murrili’s modal mineralogy based on TIMA and point counting, compared to average H chondrite values.

Composition

Mineral Compositions

Average elemental compositions of olivine, orthopyroxene and chromite, obtained using Electron Probe Micro-Analysis (EPMA) are shown in Table 2. Analysis of 15 olivine grains results in an average fayalite value of 18.8 ± 0.5 , indicating a significant amount of equilibration has occurred (consistent with the textural features of the sample). Orthopyroxene (N = 7) has an average Fs value of 16.3 ± 0.3 and an average Wo value of 1.1 ± 0.3 . Additionally, Figure 9 plots the abundance of fayalite against ferrosilite in Murrili Fe-Mg silicates, which suggests that, chemically speaking, Murrili is an H chondrite.

Bulk chemistry

We collected data on 53 major, minor, and trace elements (Figure 10). Typically, errors are <12% RSD. Lithophiles (Zr-Ba, n=29) have a mean CI and Mg normalized abundance of 0.84 ± 0.08 (1σ). Kallemeyn et al. (1989) demonstrated that ordinary chondrites possess a mean CI and Mg normalized abundance of 0.9, indicating that Murrili is an ordinary chondrite. Within the ordinary chondrites, total siderophile element content increases in the order: LL→L→H. Siderophile elements in Murrili (Re-Pd, n=9; Figure 10) have a mean CI and Mg normalized abundance of 1.30 ± 0.15 (1σ), within errors of the H chondrite range of values.

It is likely that the slight enrichment in calcium, barium and sodium relative to other lithophiles are related to terrestrial contamination, due to the month spent buried in Kati Thanda.

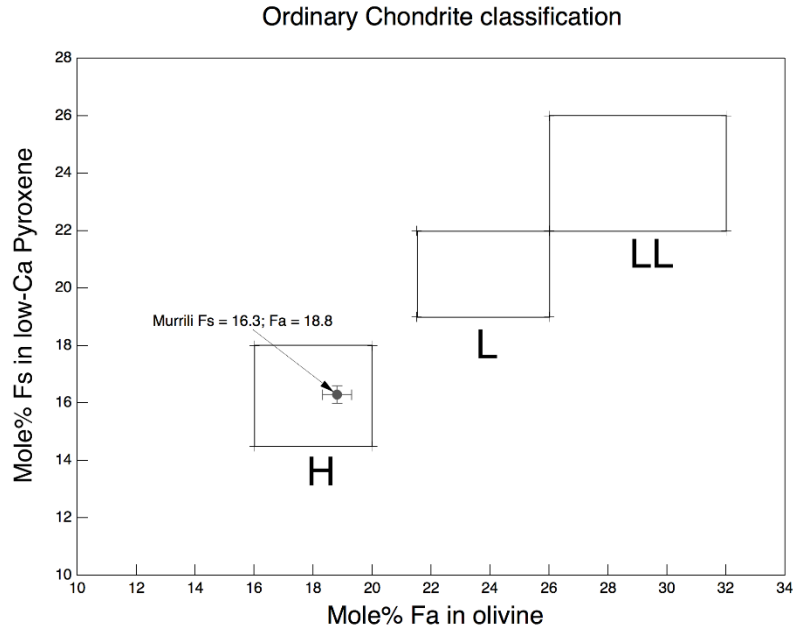


Figure 9 – Ferrosilite (Fs) value of orthopyroxene vs Fayalite (Fa) value of olivine for Murrili.

Table 2. Average olivine, orthopyroxene and chromite mineral compositions in Murrili. Number of analysed grains in parenthesis. Fa, Fs, Wo, and Chromite endmembers calculated EPMA data. Low total for chromite likely due to missing elements from analysis – most likely Mn and V which can be up to 1 wt % each in ordinary chondrites (Bunch et al, 1967) which were not included in the EPMA analysis set up.

	Olivine (N=15)		Orthopyroxene (N=8)		Chromite (N=7)	
	average	stdev	average	stdev	average	stdev
SiO ₂	38.46	0.37	55.34	0.73	0.09	0.13
TiO ₂	0.02	0.02	0.13	0.05	1.92	0.22
Al ₂ O ₃	0.01	0.03	0.15	0.09	6.27	0.26
Cr ₂ O ₃	0.02	0.03	0.13	0.06	56.95	0.75
FeO	17.77	0.44	11.10	0.15	28.92	0.25
MgO	42.92	0.36	31.21	0.28	2.85	0.13
CaO	0.03	0.03	0.59	0.15	0.04	0.04
Na ₂ O	0.05	0.06	0.05	0.03	0.17	0.27
Total	99.29	0.49	98.69	0.85	97.21	0.59
Fa	18.85	0.47				
Fs			16.45	0.30		
Wo			1.12	0.29		
Fe/Fe+Mg					85.07	0.57
Cr/Cr+Al					85.91	0.55

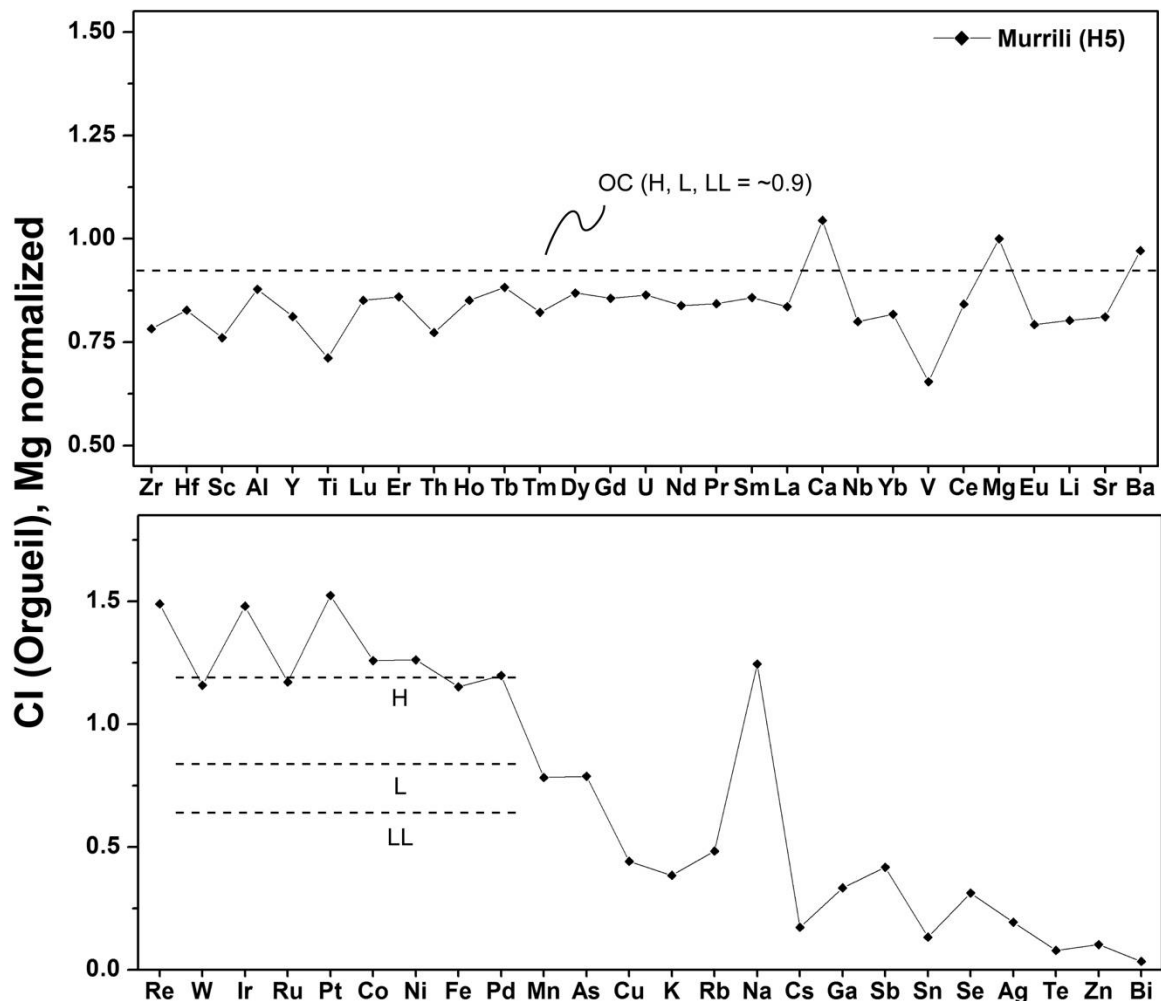


Figure 10. CI and Mg normalized elemental abundances in the Murrili H5 chondrite.

Oxygen isotopic composition

The compositional classification of Murrili as H ordinary chondrite is further supported by its oxygen isotopic composition. We analyzed a minimally-altered piece of Murrili ($\delta^{17}\text{O} = 2.764 \pm 0.016\text{‰}$; $\delta^{18}\text{O} = 3.988 \pm 0.056\text{‰}$; and $\delta^{17}\text{O} = 0.691 \pm 0.013\text{‰}$) as well as an altered region ($\delta^{17}\text{O} = 2.848 \pm 0.016\text{‰}$; $\delta^{18}\text{O} = 4.182 \pm 0.039\text{‰}$; and $\delta^{17}\text{O} = 0.673 \pm 0.004\text{‰}$) (errors 1σ), shown in Figure 11. Both isotope analyses indicate consistent H chondrite classification for Murrili, falling well within the restricted range for H chondrites defined by McDermott et al. (2016) ($\delta^{18}\text{O} = 4.16 \pm 0.42 \text{‰}$; $\delta^{17}\text{O} = 0.73 \pm 0.08 \text{‰}$ ($n=20$)). There is a slight shift to a lower $\delta^{17}\text{O}$ value and a higher $\delta^{18}\text{O}$ value in the altered piece compared to the unaltered piece. It is however, important to note that in terms of its oxygen isotope composition, the degree of alteration is very limited when taking into account the 2σ errors on the respective analyses.

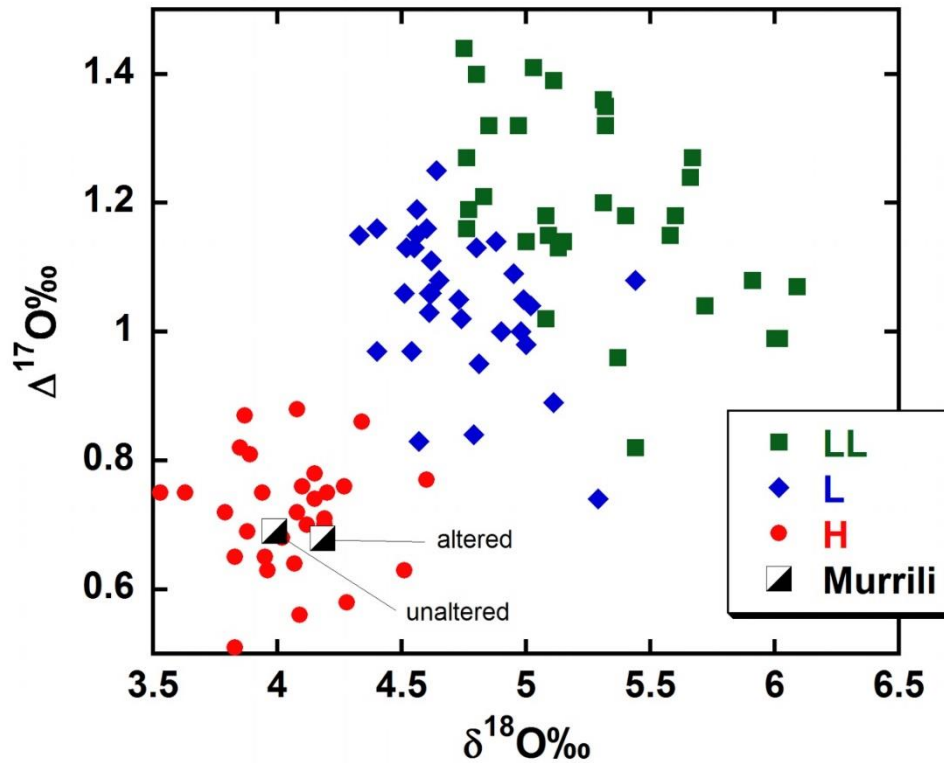


Figure 11. Oxygen isotopic composition ($\Delta^{17}\text{O}$ vs $\delta^{18}\text{O}$) of Murrili (both altered and unaltered) relative to the Ordinary Chondrite groups (from Clayton et al., 1991). $\Delta^{17}\text{O}$ is a measure of the relative difference from the Terrestrial Fractionation line (TFL) which is defined to be 1. Positive numbers are on parallel slope $\frac{1}{2}$ lines above the TFL and negative numbers are below. Murrili plots unambiguously within the H grouping of ordinary chondrite meteorites.

Chronology and Meteoroid Size

He and Ne components

The Ne isotopic composition is almost identical with the cosmogenic endmember (i.e., nearly all ^{21}Ne is cosmogenic; see Table 4). A minor trapped component, theoretically allowed within uncertainties and contributing no more than ca. 2% of the total Ne, might be of Earth atmospheric (air) or phase Q origin – the calculated concentration of cosmogenic ^{21}Ne does not depend on the choice of the trapped endmember. Similarly, the He isotopic composition of Murrili is best explained by a simple combination of cosmogenic He (all ^3He , and a fraction of the ^4He , determined by $^4\text{He}_{\text{cos}}/^3\text{He}_{\text{cos}} = \sim 6$) and radiogenic He (the remaining ^4He after subtraction of $^4\text{He}_{\text{cos}}$). While some atmospheric He might theoretically be present too, it would likely be accompanied by corresponding amount of atmospheric Ne, which already has to be very minor as found above. In combination, air He (with $^4\text{He}/^{20}\text{Ne} = \sim 0.3$) cannot contribute more than 0.003% of the measured total He.

Meteoroid Size

Using the cosmogenic noble gas and radionuclide results, we are able to estimate the pre-atmospheric size of the meteoroid that delivered Murrili to Earth. We assume that the radionuclides are saturated, i.e., that the meteorite was exposed as a small object in space for >5 Ma. The $^{36}\text{Cl}/^{10}\text{Be}$ ratio of 4.33 ± 0.18 in the metal fraction is in good agreement with the $^{36}\text{Cl}/^{10}\text{Be}$ - ^{10}Be correlation that was obtained from a large set of meteorites with long exposure ages (Lavielle et al. 1999). The $^{26}\text{Al}/^{10}\text{Be}$ ratio of 0.71 ± 0.02 in the metal fraction also matches the average saturation ratio of 0.71 ± 0.05 that was found in a large set of meteorites. We thus conclude that the ^{36}Cl , ^{26}Al , and ^{10}Be concentrations in Murrili were all produced by a single cosmic-ray exposure stage >5 Ma (as independently confirmed by cosmogenic noble gases, see below).

The measured ^{10}Be and ^{26}Al concentrations in the stone and metal fraction of Murrili (Table 3) and the stone/metal ratio of 87/13, yield bulk ^{10}Be and ^{26}Al concentrations in Murrili of 19.1 ± 0.3 and 51.4 ± 1.2 dpm/kg, respectively. These values are consistent with production rates in a relatively small object with a radius $R = 10$ -20 cm (Figure 12). The concentration of cosmogenic ^{10}Be (5.6 dpm/kg) in the metal sample of Murrili is $\sim 15\%$ higher than the maximum calculated ^{10}Be production rates in the center of an object of 10 cm radius (L&M09). This suggests that the calculated ^{10}Be production rates in the metal fraction are too low. This is not implausible, since the uncertainty in the absolute production rates are estimated at 10-15% in the model calculations of L&M09. We increased the ^{10}Be production rates of L&M09 in the metal phase – somewhat arbitrarily – by 15% to obtain better agreement with measured ^{10}Be concentrations of up to ~ 6.5 dpm/kg in the metal fraction of small chondrites. The measured ^{10}Be concentrations in the stone and metal phase yield a radius of 15-20 cm for Murrili and a relatively shallow shielding depth of 3-4 cm (Figure 13). This depth is somewhat lower than the one of >20 cm derived from the $^{22}\text{Ne}/^{21}\text{Ne}$ ratio, but the inferred size overlaps (see below).

Finally, the measured ^{36}Cl concentration of 6.7 dpm/kg in the stone fraction is consistent with calculated ^{36}Cl production rates in objects of 20-65 cm in radius (Figure 13). However, these calculated production rates only include spallation reactions on K, Ca, Ti, Fe and Ni, while objects larger than ~ 30 cm in radius also have a significant contribution of ^{36}Cl from neutron-capture on Cl, which can increase the total ^{36}Cl production rates by up to a factor of 3-5 (Welten et al. 2001). Since the measured ^{36}Cl concentration in Murrili shows no evidence of neutron-capture produced ^{36}Cl , this result is also consistent with a relatively small pre-atmospheric size. Based on the cosmogenic radionuclide data, we conclude that the Murrili meteorite probably came from an object with a pre-atmospheric radius of 15-20 cm.

Assuming a simple, single stage exposure, we can also use the model of L&M09 to derive the production rates of ^3He and ^{21}Ne for Murrili, either using the $^{22}\text{Ne}/^{21}\text{Ne}$ ratio as shielding parameter or the ^{26}Al concentration, which is more sensitive to shielding and shows a good correlation with the ^{21}Ne production rate in objects <50 cm in radius.

Based on the Ne isotopic composition of Murrili ($^{22}\text{Ne}/^{21}\text{Ne} = 1.10$), only the smallest modelled meteoroid given by L&M09 (with $R = 10$ cm) can be excluded, since all shielding depths have $^{22}\text{Ne}/^{21}\text{Ne} > 1.10$. Since no $R = 15$ cm model is provided by L&M09, we technically cannot exclude the radius suggested by the radionuclide and fireball results (ca. 15 cm). The next-larger meteoroid provided by L&M09, with $R = 20$ cm, has a compatible zone at a shielding depth of 14 ± 1 cm. For even larger modelled meteoroids, the compatible zone moves to slightly more shallow shielding depths (e.g., 10 ± 1 cm within a $R = 50$ cm meteoroid). The measured (average) $^3\text{He}/^{21}\text{Ne}$ ratio of 5.2 is close to the range of values expected under these shielding conditions (ca. 5.2 to 5.6, depending on R), providing further support for a small pre-atmospheric size and minimal to no gas loss during its transfer to Earth.

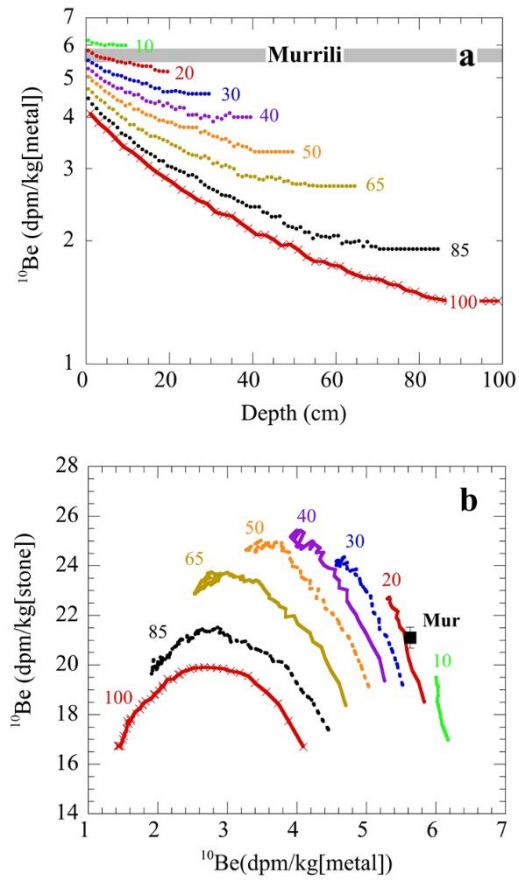


Figure 12. Comparison of measured ^{10}Be concentrations in the metal phase (a) and in stone and metal phase (b) in Murrili with calculated production rates of ^{10}Be in metal and stone fraction of H-chondrites using the model of Leya and Masarik (2009). The ^{10}Be production rates in the metal fraction were increased by 15%.

Table 3. Measured concentrations of major/minor elements (in wt%) and of cosmogenic radionuclides (in dpm kg⁻¹) in the purified metal and non-magnetic (“stone”) fraction of the Murrili H5 chondrite fall. The Mg concentration of 0.074 wt% in the metal fraction indicates that the metal contains ~0.45 wt% of silicate contamination.

Element	Metal	Stone
	65.9 mg	106.0 mg
Mg	0.074	16.2
Al	nd	1.21
K	nd	0.12
Ca	nd	1.37
Ti	nd	0.067
Mn	nd	0.23
Fe	90.7	15.0
Co	0.48	0.011
Ni	7.9	0.64
¹⁰ Be	5.63 ± 0.05	21.1 ± 0.3
²⁶ Al	3.99 ± 0.12	58.4 ± 1.4
³⁶ Cl	24.4 ± 0.4	6.70 ± 0.11

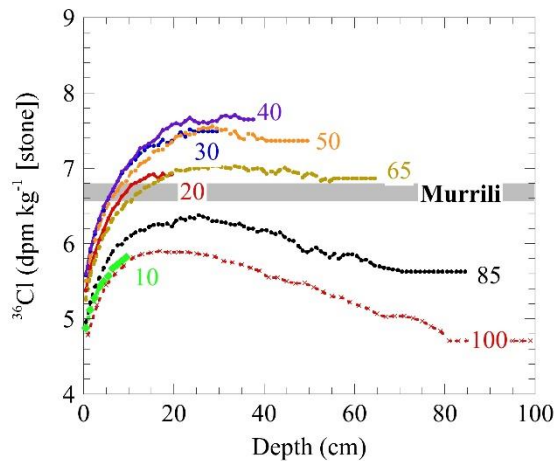


Figure 13. Comparison of measured ³⁶Cl concentration in the stone fraction of Murrili with calculated depth profiles from the model of Leya and Masarik (2009). Production rates include spallation reactions on K, Ca, Ti, Fe and Ni.

Table 4: Noble gas concentrations and derived values. All concentrations are given in units of $10^{-8} \text{ cm}^3 \text{ STP/g}$. Given uncertainties include sample mass uncertainties. The last column gives the inverse-variance weighted average of the two samples.

Sample	Murrili NG-1	Murrili NG-2	Adopted Value
Mass (mg)	45.9	22.6	-
Noble gas concentrations			
^3He	11.3 ± 0.4	12.6 ± 0.8	11.5 ± 0.3
^4He	976 ± 32	1420 ± 94	1020 ± 30
^{20}Ne	1.91 ± 0.06	2.39 ± 0.16	1.98 ± 0.06
^{21}Ne	2.01 ± 0.07	2.54 ± 0.17	2.08 ± 0.06
^{22}Ne	2.22 ± 0.07	2.80 ± 0.19	2.29 ± 0.07
Derived values			
$^{21}\text{Ne}_{\text{cos}}$	2.00 ± 0.12	2.53 ± 0.21	2.07 ± 0.17
$^3\text{He}_{\text{cos}}/^{21}\text{Ne}_{\text{cos}}$	5.62 ± 0.01	4.97 ± 0.01	5.18 ± 0.01
$^{22}\text{Ne}_{\text{cos}}/^{21}\text{Ne}_{\text{cos}}$	1.105 ± 0.002	1.103 ± 0.003	1.104 ± 0.002
$^4\text{He}_{\text{rad}}$	917 ± 42	1360 ± 130	961 ± 40

Cosmic Ray Exposure (CRE)

In addition to extrapolating a pre-atmospheric entry size for the meteoroid, the cosmic-ray produced ^3He and ^{21}Ne concentrations are used to calculate the CRE age. For the noble gas interpretation (excluding Ar), we used the inverse-variance-weighted concentrations of 11.5 (^3He) and 2.07 (^{21}Ne) $\times 10^{-8} \text{ cm}^3 \text{ STP/g}$. The empirically derived ^3He and ^{21}Ne production rates of $(1.65\text{-}1.78) \times 10^{-8} \text{ cm}^3 \text{ STP/(g Ma)}$ and $(3.2\text{-}3.4) \times 10^{-9} \text{ cm}^3 \text{ STP/(g Ma)}$ (L&M09; Dalcher et al. 2013), yield a CRE age of 6.1-7.0 Ma. This determines the time elapsed since Murrili was separated as a small object from its parent body in the asteroid belt, and overlaps with the main CRE age peak at ~ 7 Ma (Marti and Graf 1995) or $\sim 6\text{-}10$ Ma (Herzog & Caffee, 2014) for H chondrites.

Based on the measured ^{26}Al concentration of 51.4 dpm/kg for the Murrili H chondrite, the model of L&M09 yields ^3He and ^{21}Ne production rates of 1.65 and $0.27 \times 10^{-8} \text{ cc STP/g/Ma}$, respectively, for Murrili. This method yields CRE ages of 7.0 Ma for ^3He and 7.7 Ma for ^{21}Ne , which still overlap within error with the main H chondrite CRE age cluster at 6-10 Ma (Figure 14). Radiogenic ^4He (corrected for a cosmogenic contribution with $^4\text{He}_{\text{cos}}/^{3}\text{He}_{\text{cos}} = 5.2$) yields a U,Th-He age of 2.7 ± 0.1 Ga.

Using the ^{38}Ar data, gathered from 9 samples of Murrili, we calculate a weighted mean CRE age of 7.12 ± 0.41 Ma. The relative agreement of values obtained from ^3He , ^{21}Ne and ^{38}Ar , suggest minimal noble gas loss during atmospheric entry, or during its terrestrial residence.

$^{40}\text{Ar}\text{-}^{39}\text{Ar}$ Age

Our $^{40}\text{Ar}\text{-}^{39}\text{Ar}$ dating analysis yielded three plateaus and two smaller plateaus, all concordant with an average age of 4475.3 ± 2.3 Ma. Across our 17 samples there was little variation in age results, although 2 samples did yield ages up to 1 Ga younger than the rest. This suggests minor subsequent impact heating events, that had different results on the Ar system, depending on the crystal size and type, along with local porosity.

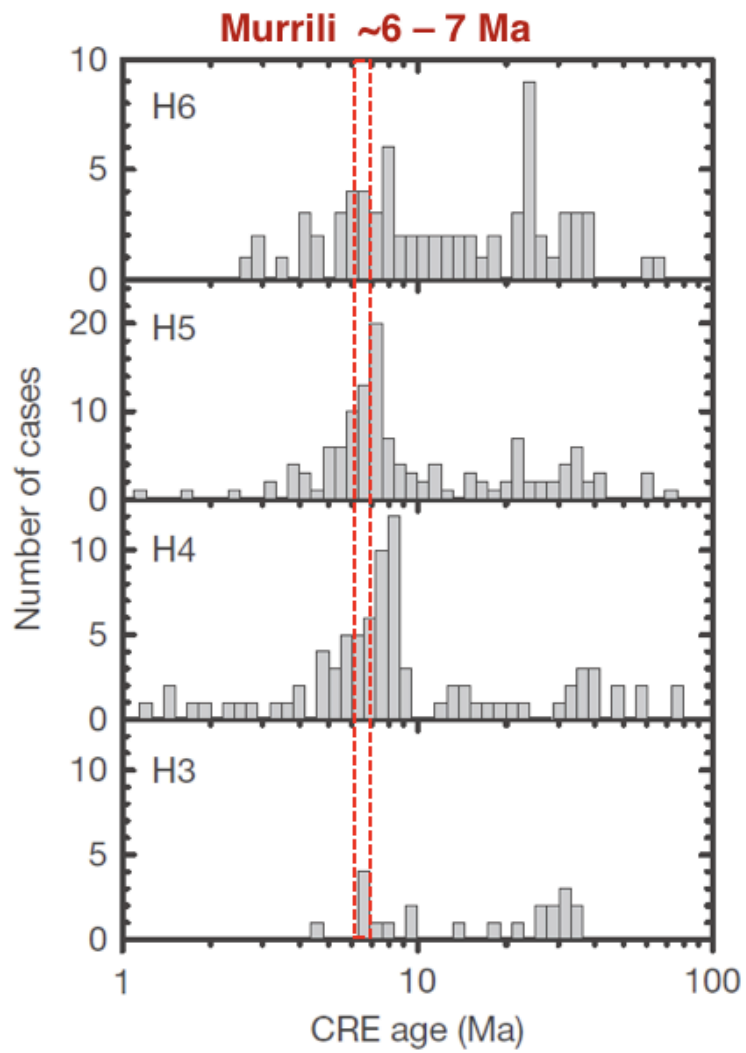


Figure 14. Cosmic Ray Exposure (CRE) ages for other H chondrites. Murrili falls into the (possibly double) peak around 6-8 Ma.

DISCUSSION

Classification

The bulk olivine and pyroxene compositions of Murrili fall squarely within the H-chondrite fields (Figures 7, 8, 9). The siderophile elements are more diagnostic than the lithophile elements (Figure 10), in line with the metal/sulfide ratio. The oxygen isotopic composition also confirms that Murrili is an H-type ordinary chondrite. Its magnetic susceptibility ($\log \chi = 5.30$) too, is consistent with other, weathered, H chondrites. The modal mineralogy of Murrili is broadly consistent with ordinary chondrites, being dominated by Fe-Mg silicates and containing metal and sulfide in a proportion roughly similar to that of H chondrites (McSween et al, 1991; Dunn, et al, 2010). The modal mineralogy derived from the thin section supports the estimate derived from CT scan density percentages. Of the whole rock, 73% has a density consistent with silicate mineralogy – the remaining 27% comprises metal, sulfide and minor minerals (phosphates and chromite, etc.). It is interesting to note that the modal mineralogy of the thin section shows a high abundance of orthopyroxene relative to olivine and a significant decrease in abundance of plagioclase relative to other chondrites – which may indicate a slight variation in oxidation states compared to other H-chondrites, or potentially a redistribution of Ca. Mason Gully, another fall recovered by the DFN, showed this modal mineralogical anomaly as well (Dyl et al 2016).

The lack of distinct chondrule boundaries, the absence of striated pyroxene, the lack of chondrule glass, an overall recrystallised texture and the equilibrated silicate mineral compositions ($\text{Fa}_{18.8 \pm 0.5}$; $\text{Fs}_{16.4 \pm 0.3}$; $\text{Wo}_{1.1 \pm 0.3}$; Scott et al 1986), indicate that the petrographic type of this meteorite is most consistent with type 5. The morphologies of the CT scans of both the main mass and the 50 g piece support this.

The above modal mineralogy, mineral chemistry, morphologies, and isotopic compositions, indicate that Murrili is an H5 chondrite with extensive weathering. We will discuss shock features in detail below but based on a difference between features in thin section and CT scans, impact affected this rock heterogeneously and Murrili is a likely a breccia with indistinct lithic clasts.

Terrestrial Alteration

Cut surfaces reveal pervasive alteration with rusty staining heterogeneously distributed in a wormy pattern (Figure 4). Results from Mossbauer spectroscopy (Figure 16) point to an aggressive weathering process. Both of these results are consistent with the length of time Murrili resided in the salt lake environment, allowing alteration to occur throughout the low-porosity rock. There is no difference in mineral composition between the altered and unaltered regions, though there are discrepancies between the two regions in the Mössbauer analyses, and the oxygen isotope measurements.

Physical properties

Density and Porosity

For a fresh fall with an assumed low shock state (proposed by thin section analysis), Murrili's porosity of 3.4% is very low. The average H fall is about 10% porous while most S1s within the H falls are between about 7-14 % porous, with porosity decreasing as shock increases (Consolmagno et al. 1998).

Shock

The thin sections we examined showed an overall low (S2) shock state apparent in undulose extinction in numerous olivine grains, while the fine resolution tomographs of the 50 g wedge show metal foliation, indicative of moderate (S4-S5) shock loading (Friedrich et al. 2008). The coarse resolution tomographs of the whole meteorite show that metal foliation is present throughout the most of the rock, including the region that the wedge was taken from, but excluding the locality where the thin sections were sampled.

Although we did not observe brecciation at the micro scale in either of our thin sections or in our CT tomographs, larger cracks are apparent in the wedge we used for both helium pycnometry and fine resolution CT (Figure 4a). Due to the heterogeneous nature of its shock features, as well as a lack of micro-scale brecciated texture, Murrili is most likely brecciated at the cm-dm scale.

Chronology

The ^{40}Ar - ^{39}Ar age of Murrili is dated to 4475.3 ± 2.3 Ma which fits well with other H Chondrites (Trieloff et al. 2003). Although we recorded some minor variation in 2 of samples, which may suggest post-formation impact events, we do not see apparent evidence of major brecciation. This most likely suggests minor impact events just after recrystallization. This chronological anomaly, along with the heterogeneous nature of Murrili's shock state will be investigated further in a forthcoming publication.

Murrili has a cosmic-ray exposure age (CRE) that falls within the broad 6-10 Ma peak in the CRE age histogram of the H chondrites (Graf & Marti, 1995). Graf and Marti (1995) suggested that this peak, which contains about ~50% of the H-chondrites, might be a double peak, with ages around ~6 Ma more prominent for H5 chondrites than for other petrographic types of H chondrites, which typically have CRE ages around ~8 Ma (Figure 15). The CRE age of Murrili, which is between 6.1 and 7.7 Ma depending on the method chosen, cannot provide further support for this pattern. The relatively high $^3\text{He}/^{21}\text{Ne}$ ratio suggests that Murrili is derived from a rather small meteoroid with a radius of ~20 cm (certainly >10 cm). This is compatible with the estimated radius of ~14 cm inferred by Sansom et al. (2020) from the photographic observations of the fireball, and the estimated radius of 15-20 cm from the cosmogenic radionuclide data.

CONCLUSIONS

Murrili formed as an H5 chondrite on its parent body 4475.3 ± 2.3 Ma, experiencing subsequent but minor impacts which left it heterogeneously shocked and brecciated at the cm-scale. Approximately 6-8 Ma, Murrili's precursor meteoroid was separated from its parent body, at a size of 15-20 cm in radius. Just prior to its collision with Earth, the meteoroid had an orbit with a semi-major axis of ~2.5 AU with a low inclination, near the 3:1 mean resonance with Jupiter, which is not uncommon for other orbitally-determined H5s.

The meteoroid entered the Earth's upper atmosphere at a speed of ~13 km/s, over South Australia on the night of November 27th, 2015 at 9:15 pm (local time). After it stopped ablating, it eventually fell into the salt lake: Kati Thanda (South Lake Eyre), punching through the upper crust of the lake, embedding itself 42 cm below the surface. Although there was no standing water on the surface of the lake during the fall, the clay soil below the surface was saturated with a salt brine, which heavily weathered some portions of the meteorite before it was recovered a month later. The main characteristics of Murrili, and how we measured them are listed in Table 4.

Although Murrili has been thoroughly characterized here, one anomaly remains unexplored with this meteorite: its shock history. Most of the ^{40}Ar - ^{39}Ar measurements for Murrili indicate an age of 4475.3 ± 2.3 Ma, except for two readings that yield ages up to 1 Ga younger. This, combined with Murrili's heterogeneous shock state requires further study into its shock history.

Table 4. Summary of the characteristics appearing in the Murrili meteorite.

Property	Results	Method
Petrologic Type	H5	Optical Microscopy, Computed Tomography, Laser Assisted Fluorination, TIMA, Electron Microprobe, Magnetic Susceptibility
Porosity	3.4 % 3.4 ± 0.4%	Computed Tomography Helium Pycnometry
Bulk Density	3.6 g cm ⁻³ 3.47 ± 0.01 g cm ⁻³	Computed Tomography Helium Pycnometry
Grain Density	3.59 ± 0.01 g cm ⁻³	Helium Pycnometry
Magnetic Susceptibility (log χ)	5.30	Magnetic Susceptibility
Fe(III)/Total Fe	3-12 %	Mössbauer Spectroscopy
Shock Classification	S2 S4-S5	Optical Microscopy Computed Tomography
⁴⁰ Ar- ³⁹ Ar Age	4475.3 ± 2.3 Ma	⁴⁰ Ar- ³⁹ Ar Dating
Cosmic Ray Exposure Age	6.1-7 Ma 7 Ma 7.7 Ma 7.12 ± 0.41 Ma	Empirical Derivation L&M09 Model (³ He) L&M09 Model (³ He) ³⁸ Ar concentration
Meteoroid Size (radius)	15-20 cm 14 cm	Noble Gas concentrations Kalman Filter (Sansom et al. 2020)

Acknowledgments- We thank the Arabana people, the traditional custodians of the land, for granting us permission and assisting us to recover Murrili. We also thank our two anonymous reviewers for their inputs, which helped to improve the quality of this manuscript. This work was funded by the Australian Research Council (Grant: DP200102073).

References

Bland P. A., Berry F. J., Pillinger C. T. 1998. Rapid weathering in Holbrook: An iron-57 Mössbauer spectroscopy study. *Meteoritics & Planetary Science* 33(1):127-129.

Bland P.A., Spurný P., Bevan A.W.R., Howard K.T., Towner M.C., Benedix G.K., Greenwood R.C., Shrubeny L., Franchi I.A., Deacon G. and Borovička J. 2012. The Australian Desert Fireball Network: a new era for planetary science. *Australian Journal of Earth Sciences* 59(2):177-187.

Bland P. A., Towner M. C., Sansom E. K., Devillepoix H., Howie R. M., Paxman J. P., Cupak M., Cox M. A., Jansen-Sturgeon T., Stuart, D. 2016. Fall and recovery of the murrili meteorite, and an update on the desert fireball network. *79th Annual Meeting of the Meteoritical Society* (Vol. 1921).

Bunch T. E., Keil K., Snetsinger K. G. 1967. Chromite composition in relation to chemistry and texture of ordinary chondrites. *Geochimica et Cosmochimica Acta* 31(10):1569-1582.

Clayton R.N., Mayeda T.K., Goswami J.N., Olsen, E.J. 1991. Oxygen isotope studies of ordinary chondrites. *Geochimica et Cosmochimica Acta* 55(8):2317-2337.

Consolmagno S.J., G. J., Britt D. T., Stoll C. P. 1998. The porosities of ordinary chondrites: Models and interpretation. *Meteoritics & Planetary Science* 33(6):1221-1229.

Dalcher N., Caffee M. W., Nishiizumi K., Welten K. C., Vogel N., Wieler R., Leya I. 2013. Calibration of cosmogenic noble gas production in ordinary chondrites based on ³⁶Cl-³⁶Ar ages. Part 1: Refined produced rates for cosmogenic ²¹Ne and ³⁸Ar. *Meteoritics & Planetary Science* 48(10):1841-1862.

Dunn T.L., Cressey G., McSween Jr H.Y., McCoy T.J., 2010. Analysis of ordinary chondrites using powder X-ray diffraction: 1. Modal mineral abundances. *Meteoritics & Planetary Science* 45(1):123-134.

Dyl K.A., Benedix G.K., Bland P.A., Friedrich J.M., Spurný P., Towner M.C., O'Keefe M.C., Howard K., Greenwood R., Macke R.J., Britt D.T., 2016. Characterization of Mason Gully (H5): the second recovered fall from the Desert Fireball Network. *Meteoritics & Planetary Science* 51(3):596-613.

Eugster O. and Michel T. 1995. Common asteroid break-up events of eucrites, diogenites, and howardites and cosmic-ray production rates for noble gases in achondrites. *Geochimica et Cosmochimica Acta* 59(1):177-199.

Ford R. L., Benedix G. K., McCoy T. J., Rushmer T. 2008. Partial melting of H6 ordinary chondrite Kernouvé: constraints on the effects of reducing conditions on oxidized compositions. *Meteoritics & Planetary Science* 43(8):1399-1414.

Friedrich J. M., Wang M. S., Lipschutz M. E. 2003. Chemical studies of L chondrites. V: Compositional patterns for 49 trace elements in 14 L4-6 and 7 LL4-6 falls. *Geochimica et Cosmochimica Acta* 67(13):2467-2479.

Friedrich J. M., Macke R. J., Wignarajah D. P., Rivers M. L., Britt D. T., & Ebel D. S. 2008. Pore size distribution in an uncompacted equilibrated ordinary chondrite. *Planetary and Space Science* 56(7):895-900.

Friedrich J.M., Ruzicka A., Rivers M.L., Ebel D.S., Thostenson J.O., Rudolph R.A. 2013. Metal veins in the Kernouvé (H6 S1) chondrite: Evidence for pre- or syn-metamorphic shear deformation. *Geochimica et Cosmochimica Acta* 116:71-83.

Friedrich J. M., Ruzicka A., Macke R. J., Thostenson J. O., Rudolph R. A., Rivers M. L. and Ebel D. S. 2017 Relationships among physical properties as indicators of high temperature deformation or post-shock thermal annealing in ordinary chondrites. *Geochimica et Cosmochimica Acta* 203:157-174.

Friedrich J.M. and Rivers M.L. 2013. Three-dimensional imaging of ordinary chondrite microporosity at 2.6 µm resolution. *Geochimica et Cosmochimica Acta* 116:63-70

Gattacceca J., Eisenlohr P., Rochette P. 2004. Calibration of in situ magnetic susceptibility measurements. *Geophysical Journal International* 158(1):42-49.

Godel B. 2013. High-resolution X-ray computed tomography and its application to ore deposits: From data acquisition to quantitative three-dimensional measurements with case studies from Ni-Cu-PGE deposits. *Economic Geology* 108(8): 2005-2019.

Godel B., Barnes S. J., Maier W. D. 2006. 3-D distribution of sulphide minerals in the Merensky Reef (Bushveld Complex, South Africa) and the JM Reef (Stillwater Complex, USA) and their relationship to microstructures using X-ray computed tomography. *Journal of Petrology* 47(9):1853-1872.

Graf T. and Marti K. 1995. Collisional history of H chondrites. *Journal of Geophysical Research: Planets* 100(E10):21247-21263.

Greenwood R.C., Burbine T.H., Miller M.F., Franchi I.A. 2017. Melting and differentiation of early-formed asteroids: The perspective from high precision oxygen isotope studies. *Geochemistry* 77(1):1-43.

Habeck-Fardy A., Nanson G. C. 2014. Environmental character and history of the Lake Eyre Basin, one seventh of the Australian continent. *Earth-Science Reviews* 132: 39-66.

Hennessy J. and Turner G. 1980. ^{40}Ar — ^{39}Ar ages and irradiation history of Luna 24 basalts. *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences* 297(1428):27-39.

Herzog G.F. and Caffee M.W. 2014. Cosmic-ray exposure ages of meteorites. *Mcp* 1:419-454.

Howie R. M., Paxman J., Bland P. A., Towner M. C., Cupak M., Sansom E. K., Devillepoix H. A. 2017. How to build a continental scale fireball camera network. *Experimental Astronomy* 43:237-266.

Jenniskens, P. 2013. Recent documented meteorite falls, a review of meteorite–asteroid links. *Meteoroids 2013: proceedings of the astronomical conference held at AM University, Poznan* 57-68.

Jourdan F., Kennedy T., Benedix G., Eroglu E., Mayer C. 2020. Timing of the magmatic and upper crustal cooling of differentiated asteroid 4 Vesta. *Geochimica et Cosmochimica Acta* 273: 205-225.

Kallemeyn G. W., Rubin A. E., Wang D., Wasson J. T. 1989. Ordinary chondrites: Bulk compositions, classification, lithophile-element fractionations and composition-petrographic type relationships. *Geochimica et Cosmochimica Acta* 53(10):2747-2767.

Kennedy T., Jourdan F., Bevan A.W., Gee M.M., Frew A., 2013. Impact history of the HED parent body (ies) clarified by new $^{40}\text{Ar}/^{39}\text{Ar}$ analyses of four HED meteorites and one anomalous basaltic achondrite. *Geochimica et Cosmochimica Acta* 115:162-182.

Korochantseva E.V., Trieloff M., Buikin A.I., Hopp J., Meyer H.P. 2005. $^{40}\text{Ar}/^{39}\text{Ar}$ dating and cosmic-ray exposure time of desert meteorites: Dhofar 300 and Dhofar 007 eucrites and anomalous achondrite NWA 011. *Meteoritics & Planetary Science* 40(9-10):1433-1454.

Lavielle B., Marti K., Jeannot J.-P., Nishiizumi K. and Caffee M. W. 1999. The ^{36}Cl - ^{36}Ar - ^{40}K - ^{41}K

records and cosmic-ray production in iron meteorites. *Earth and Planetary Science Letters* 170:93-104.

Levine J., Renne P.R., Muller R.A. 2007. Solar and cosmogenic argon in dated lunar impact spherules. *Geochimica et Cosmochimica Acta* 71(6):1624-1635.

Leya I. and Masarik J. 2009. Cosmogenic nuclides in stony meteorites revisited. *Meteoritics & Planetary Science* 44(7):1061-1086.

Macke R. J. 2010. Survey of meteorite physical properties density, porosity and magnetic susceptibility. *Doctoral Thesis: University of Central Florida*

Macke R. J., Kent J. J., Kiefer W. S., Britt D. T. 2015. 3D-Laser-Scanning Technique Applied to Bulk Density Measurements of Apollo Lunar Samples.

Masarik J. and Beer J. 2009. An updated simulation of particle fluxes and cosmogenic nuclide production in the Earth's atmosphere. *Journal of Geophysical Research: Atmospheres* 114:11.

Marti K. and Graf T. 1995. Collisional history of H chondrites. *Journal of Geophysical Research: Planets* 100(E10):21247-21263.

McDermott, K. H., Greenwood, R. C., Scott E. R. D., Franchi, I. A., Anand M. 2016. Oxygen isotope and petrological study of silicate inclusions in IIE iron meteorites and their relationships with H chondrites. *Geochimica et Cosmochimica Acta* 173:97-113.

McSween Jr H. Y., Bennett III M. E., Jarosewich E. 1991. The mineralogy of ordinary chondrites and implications for asteroid spectrophotometry. *Icarus* 90(1):107-116.

Meier, M. M. M. "Meteoriteorbits. Info-tracking all known meteorites with photographic orbits." *LPI 1964* (2017): 1178.

Meier M.M., Welten K.C., Riebe M.E., Caffee M.W., Gritsevich M., Maden C., Busemann H. 2017. Park Forest (L5) and the asteroidal source of shocked L chondrites. *Meteoritics & Planetary Science* 52(8):1561-1576.

Miller M. F., Franchi I. A., Sexton A. S., Pillinger C. T. 1999. High precision $\delta^{17}\text{O}$ isotope measurements of oxygen from silicates and other oxides: method and applications. *Rapid Communications in Mass Spectrometry* 13(13):1211-1217.

Nishiizumi K. 2004. Preparation of ^{26}Al AMS standards. *Nuclear Instruments and Methods in Physics Research*. B223–224:388–392.

Nishiizumi K., Imamura M., Caffee M. W., Southon J. R., Finkel R. C., McAninch J. 2007. Absolute calibration of ^{10}Be AMS standards. *Nuclear Instruments and Methods in Physics Research* B258:403–413.

Sansom, E. K., Bland, P. A., Towner, M. C., Devillepoix, H. A. R., Cupak, M., Howie, R. M., Jansen-Sturgeon, T., Cox, M. A., Hartig, B. A. D., Paxman, J., Benedix, G. Forman, L. V. (in review). Murrili meteorite's fall and recovery from Kati Thanda. *Meteoritics & Planetary Science*.

Scott E. R., Taylor G. J., Keil, K. 1986. Accretion, metamorphism, and brecciation of ordinary chondrites: Evidence from petrologic studies of meteorites from Roosevelt County, New Mexico. *Journal of Geophysical Research: Solid Earth* 91(B13):E115-E123.

Sharma P., Kubik P. W., Fehn U., Gove G. E., Nishiizumi K. and Elmore D. 1990. Development of ^{36}Cl standards for AMS. *Nuclear Instruments and Methods* B52:410-415.

Sharma P., Bourgeois M., Elmore D., Granger D., Lipschutz M. E., Ma X., Miller T., Mueller K., Rickey F., Simms P., Vogt S. 2000. PRIME lab AMS performance, upgrades and research applications. *Nuclear Instruments and Methods in Physics Research* B172:112–123.

Spurný P., Bland P.A., Shrubný L., Borovicka J., Ceplecha Z., Singleton A., Bevan A.W., Vaughan D., Towner M.C., McClafferty T., Toumi R. 2012. The Bunburra Rockhole meteorite fall in SW Australia: fireball trajectory, luminosity, dynamics, orbit, and impact position from photographic and photoelectric records. *Meteoritics and Planetary Science* 47(2):163-185.

Starkey N.A., Jackson C.R.M., Greenwood R.C., Parman S., Franchi I.A., Jackson M., Fitton J.G., Stuart F.M., Kurz M., Larsen L.M. 2016. Triple oxygen isotopic composition of the high- $^3\text{He}/^4\text{He}$ mantle. *Geochimica et Cosmochimica Acta*, 176:227-238.

Stöffler D., Keil K., RD S. E. 1991. Shock metamorphism of ordinary chondrites. *Geochimica et Cosmochimica Acta* 55(12), 3845-3867.

Stöffler D., Hamann C., Metzler K. 2018. Shock metamorphism of planetary silicate rocks and sediments: Proposal for an updated classification system. *Meteoritics & Planetary Science* 53(1):5-49.

Trieloff M., Jessberger E.K., Herrwerth I., Hopp J., Fiéni C., Ghéllis M., Bourot-Denise M., Pellas P. 2003. Structure and thermal history of the H-chondrite parent asteroid revealed by thermochronometry. *Nature* 422(6931):502-506.

Turner G., Crowther S.A., Burgess R., Gilmour J.D., Kelley S.P., Wasserburg G.J., 2013. Short lived ^{36}Cl and its decay products ^{36}Ar and ^{36}S in the early solar system. *Geochimica et Cosmochimica Acta* 123:358-367.

Welten K.C., Nishiizumi K., Masarik J., Caffee M.W., Jull A.J.T., Klandrud S.E., Wieler, R. 2001. Cosmic-ray exposure history of two Frontier Mountain H-chondrite showers from spallation and neutron-capture products. *Meteoritics & Planetary Science* 36(2):301-317.

Welten K. C., Caffee M. W., Hillegonds D. J., McCoy T. J., Masarik J., Nishiizumi K. 2011. Cosmogenic radionuclides in L5 and LL5 chondrites from Queen Alexandra Range, Antarctica: Identification of a large L/LL5 chondrite shower with a preatmospheric mass of approximately 50,000 kg. *Meteoritics & Planetary Science* 46(2):177-196.

Wolf S. F., Compton J. R., Gagnon C. J. 2012. Determination of 11 major and minor elements in chondritic meteorites by inductively coupled plasma mass spectrometry. *Talanta* 100:276-281.

Woodcock N. H. 1977. Specification of fabric shapes using an eigenvalue method. *Geological*

Society of America Bulletin 88(9):1231-1236.

Woodcock N. H., Naylor M. 1983. Randomness testing in three-dimensional orientation data.
Journal of Structural Geology 5(5):539-548.

Chapter VI: The Silicate Sulfuric Acid Process

(Published in: *Acta Astronautica*, 188, p.57-63)

The Proposed Silicate-Sulfuric Acid Process: Mineral Processing for In Situ Resource Utilization (ISRU)

Seamus L. Anderson*¹, Eleanor K. Sansom¹, Patrick M. Shober¹, Benjamin A.D. Hartig¹, Hadrien A.R. Devillepoix¹, Martin Towner¹

¹Space Science and Technology Center, Curtin University, GPO Box U1987, Perth, WA 6845, Australia

*Corresponding author: seamus.anderson@postgrad.curtin.edu.au

Abstract

Volatile elements and compounds found in extra-terrestrial environments are often the target of In Situ Resource Utilization (ISRU) studies. Although water and hydroxide are most commonly the focus of these studies as they can be used for propellant and human consumption; we instead focus on the possible exploitation of sulfur and how it could be utilized to produce building materials on the Moon, Mars and Asteroids. We describe the physical and chemical pathways for extracting sulfur from native sulfide minerals, manufacturing sulfuric acid in situ, and using the produced acid to dissolve native silicate minerals. The final products of this process, which we call the Silicate-Sulfuric Acid Process (SSAP), include iron metal, silica, oxygen and metal oxides, all of which are crucial in the scope of a sustainable, space-based economy. Although our proposed methodology requires an initial investment of water, oxygen, and carbon dioxide, we show that all of these volatiles are recovered and reused in order to repeat the process. We calculate the product yield from this process if it were enacted on the lunar highlands, lunar mare, Mars, as well as an array of asteroid types.

1. Introduction

Humanity's renewed interest in deep-space exploration will bring to bear countless challenges we as a species have not yet faced. As humans plan to depart from Earth for longer durations and further distances than ever before, they must be equipped with an increasingly large stockpile of resources in order to survive and thrive on their voyages. An alternative to this 'bring everything' approach, is to instead make use of the resources present at the various destinations, such as the Moon and Mars. This concept is known as in situ resource utilization (ISRU), and although the basic principle is not a novel idea [1], its numerous

demonstrations and theoretical implementations are emerging at a faster rate than ever before [2,3,4,5].

Previous ISRU studies have investigated a wide range of possibilities for extracting and utilizing materials from celestial bodies across the solar system, from rare earth elements on certain asteroids [6], to the constructional uses of regolith for radiation shielding on the Moon and Mars [7]. Much of this current literature regarding ISRU focuses on extracting volatile elements, chiefly oxygen and hydrogen, which are often locked away in the form of non-volatile minerals. The importance of these two resources cannot be understated, since they can be used not only as breathable air and potable water for human consumption but could also serve as propellant for rocket engines [5,8,9]. To that end, carbon also plays an important role in martian ISRU, since it can be combined with hydrogen to make fuel for methane-based rocket engines [10,11,12]. Extracting volatile elements and refining propellants on the surfaces of the Moon and Mars will significantly reduce the mass required to launch from Earth, and therefore the cost of spacecraft for interplanetary missions. That being said, we focus on an often overlooked volatile element that is present on the Moon, Mars and numerous asteroids: sulfur.

A study by Vaniman et al. [13] illustrated possible uses for lunar sulfur, from sealants to electricity generation and storage. A series of experiments have also evaluated the mechanical properties and durability of concrete made from lunar soil simulant and elemental sulfur [14,15]. Unfortunately the deterioration of such concretes due to simulated lunar thermal cycling and degassing under vacuum has reduced the interest in this application in recent years.

Here, we will instead explore the physical and chemical pathways for manufacturing sulfuric acid (H_2SO_4) from native sulfur in order to dissolve silicate minerals, also native to these bodies. Doing so would produce considerable amounts of iron metal, silica, oxygen as well as other useful building materials for permanent human settlement in space. We call this methodology: The Silicate-Sulfuric Acid Process (SSAP). Although mineral processing for ISRU using sulfuric acid has been examined in the past [16, 17], harvesting oxygen from the less abundant mineral ilmenite ($FeTiO_3$) was the main focus. In this paper, a variety of highly-abundant silicate minerals are the principle focus for refinement into building materials.

Using a relatively small investment of other volatile elements including carbon, oxygen, and hydrogen, this process could be carried out on the Moon, Mars, and many Asteroids. Although this initial investment could be costly, many of these elements can be found in situ especially for mission profiles where the primary goal is to extract water, such as on C-type asteroids, the lunar poles, and high-latitude locations on Mars. The other main invested volatile, carbon, is also found on C-type asteroids, as well as on the surface and atmosphere of Mars. Furthermore, we will show that the SSAP allows for inherent recycling of the invested volatile elements, for continuous reuse.

1.1 Sulfur Availability

In the ordinary and carbonaceous chondrite meteorite groups, which originated from S and C-type asteroids, respectively [18,19,20], sulfur is fairly abundant (~2.5 wt%). In ordinary chondrites it is almost entirely

contained in the mineral troilite (FeS) [21], while carbonaceous chondrites usually contain pyrrhotite (Fe_{1-0.8}S) and pentlandite ((Fe, Ni)₉S₈) instead [22]. Martian dust can contain 2.5 wt% sulfur [23], while sulfates (e.g. MgSO₄, FeSO₄) are regularly detected in martian soils [24,25]. On the Moon, troilite exists across most of the surface, although in comparatively low abundance (<0.5 wt%) [26]. In the permanently shadowed crater regions near the poles, the sulfur content could be as high as 1 wt% in the form of SO₂ ice [27]. Although sulfur may not be one of the most abundant elements on these bodies, its presence as a minor element still offers an opportunity to utilize it.

1.2 Silicate-Bound Resources

Silicate minerals are an obvious target for acid-driven dissolution, since they are abundant on essentially all terrestrial bodies. They usually exist in the general formula: $\alpha_i\text{Si}_j\text{O}_k$, where α can represent Mg, Fe, Ca, Al, Na, K, or other metals present within the crystal lattice. On the Moon, silicate minerals mostly consist of plagioclase (CaAl₂Si₂O₈), pyroxene ([Mg,Fe]₂Si₂O₆) and some olivine ([Mg,Fe]₂SiO₄), totaling more than 70 vol% of the regolith [28]. Similarly, most soils on Mars have a total silicate abundance near 80 wt%, where it is also mostly composed of plagioclase, pyroxene and olivine [29]. On C-type asteroids, with similar mineralogies to CM and CI meteorites, silicates in the form of olivine and water-bearing phyllosilicates, collectively account for between 75 and 95 % of the total weight [22]. Meanwhile S-type asteroids contain mostly olivine and pyroxene with some plagioclase, totaling between 75 and 90 wt% of these bodies [30, 19]. Once the metal components in these minerals are liberated they would be extremely valuable and useful for building large structures and other essential hardware off-world.

Iron is a prime example since it is already widely used for building large structures here on Earth, particularly when it is combined with other elements to form steel. Currently, the only way to obtain steel off-world is to launch it into orbit from Earth, a very expensive means of construction. Alternatively, producing iron metal at the desired destination could significantly reduce mission costs when the goal is to make large, permanent structures and equipment in space.

Silica (SiO₂), another main product of the SSAP, can serve as the precursor for fused quartz, which has been used as spacecraft windows on the Space Shuttle orbiters and the International Space Station [31]. Fused quartz is made by melting silica grains (~1650 °C) either under vacuum or in an inert atmosphere. Due to this high melting temperature, fused quartz can be used in some high temperature environments. This additionally makes it a possible candidate for constructing some of the equipment required for the processes we describe below.

As previously discussed, oxygen has obvious applications for human space exploration, since it can be used as breathable air. When carefully combined with hydrogen it can also be used as rocket propellant; otherwise it forms pure water for both human consumption and industrial processes.

The various oxides formed by the SSAP have an extensive variety of niche uses, possibly the most useful of which is the molten electrolysis of aluminum oxide to form pure metal. Although this refinement is not a focus of this paper, aluminum metal could feed into the fabrication of lightweight, high-strength alloys.

2. Physical and Chemical Pathways of the SSAP

The proposed Silicate-Sulfuric Acid Process consists of four main stages. The first stage contains optional pre-processing steps so that the SSAP can be compatible with other resource extraction methods such as water harvesting. The second stage entails the synthesis of sulfuric acid, either from indigenous minerals or from recycled sulfur, water and oxygen. The third stage then uses this sulfuric acid to dissociate the silicate minerals into silica and sulfate minerals; the former of which is a final product and is removed from further processing. The sulfates are then thermally decomposed, and some of their products reduced to form metal and simple oxides in the fourth and final stage. Each of these stages is discussed below in further detail. Fig. 1 provides an overview of the SSAP in the form of a flow diagram.

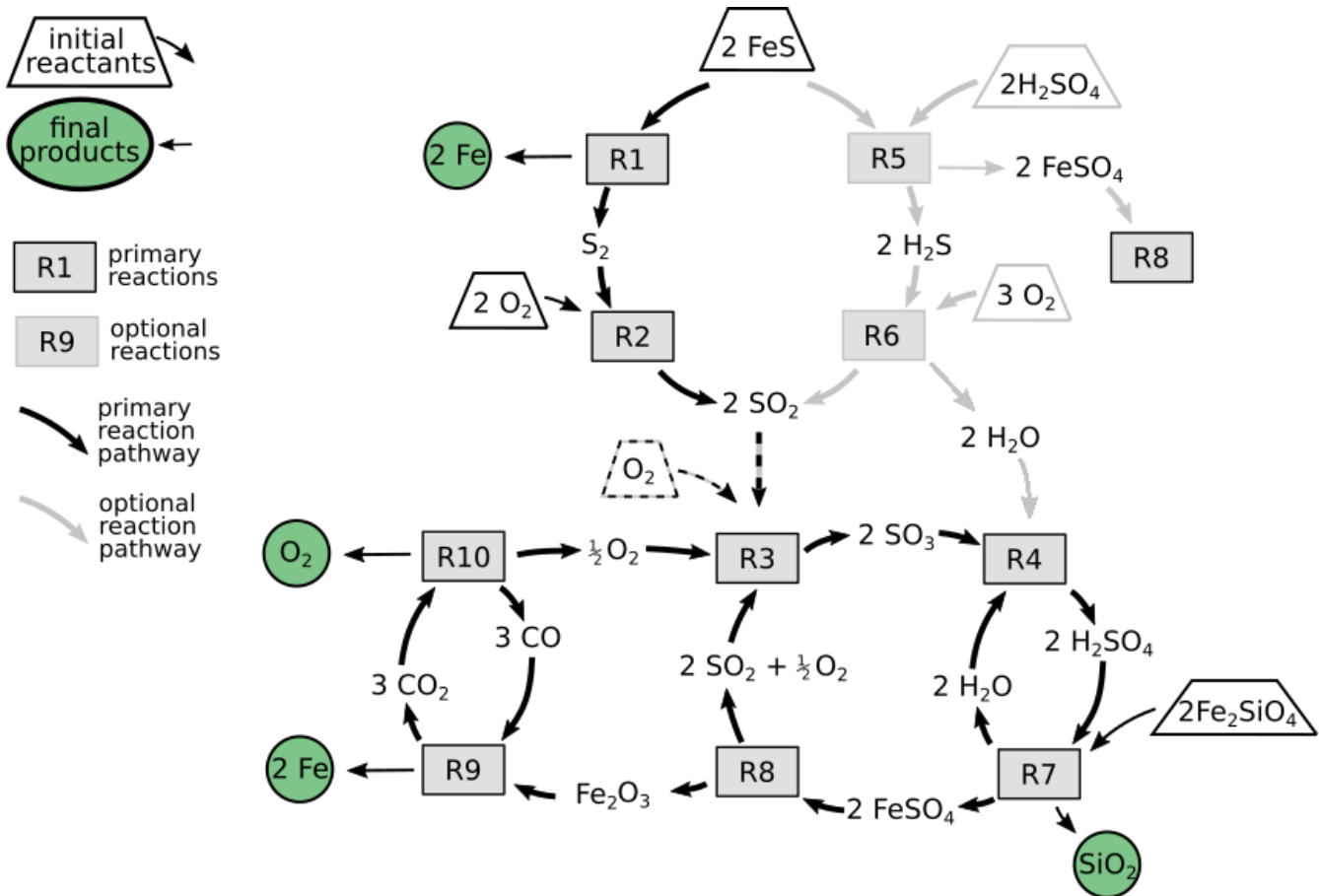


Fig. 1. This flowchart illustrates the chemical reaction pathways for the SSAP, while also highlighting the recycling of oxygen, sulfur dioxide, carbon monoxide and water. For simplicity, this shows only the processing of fayalite (Fe_2SiO_4). Final products are signified by a green circle, input reactants are outlined by trapezoids, and named reactions steps (see text) appear in rectangles. The upper half of this figure shows two pathways for processing troilite, either through thermal decomposition (left) or acid-dissociation (right) when sulfuric acid is available.

For each of the reaction steps, we calculate the change in Gibbs free energy (ΔG), at 20 °C and 1 bar, from reported values in the NIST Standard Reference Database [32]. Eq. (1) shows this relation to enthalpy (H), entropy (S) and temperature (T). For the minerals that did not appear in this database, we compiled their thermodynamic properties from individual sources (listed in supplementary materials). Although this approach of combining multiple databases for comparison and calculation is not ideal, it does provide an estimate for the energy requirements at each step.

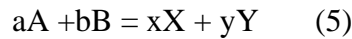
$$\Delta G = \Delta H - T\Delta S \quad (1)$$

We also calculate Gibbs free energy at pressures of 10^{-2} , 10^{-5} , and 10^{-8} bar (G_p) for each compound we assess, using Eq. (2), where R is the ideal gas constant, n is number of moles of gas, and P is pressure. From these G_p values, we also calculate the temperature-dependent equilibrium constant (K_{eq}) at those pressures via Eq. (3). This enables us to predict the equilibrium composition of each reaction using Eq. (4) and (5), where a compound's concentration is signified by square brackets (e.g. [A]). This is especially important for reactions that involve thermal decomposition, as we will show that performing these steps under vacuum can significantly reduce the temperatures required for the reaction to proceed.

$$G_p = G + nRT \ln(P) \quad (2)$$

$$K_{eq} = e^{\left(\frac{-\Delta G_p}{RT}\right)} \quad (3)$$

$$K_{eq} = \frac{[X]^x [Y]^y}{[A]^a [B]^b} \quad (4)$$



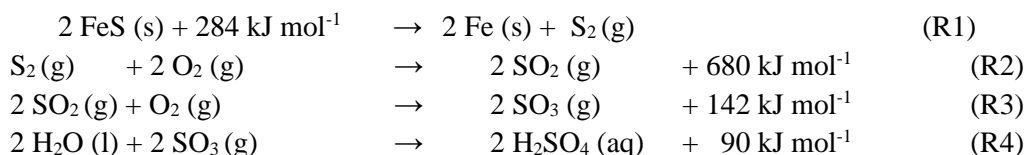
2.1 Pre-processing

Ideally the first step in the SSAP consists of mechanically crushing larger silicate rocks to reduce the average particle size, which would allow for a quicker reaction with the sulfuric acid. This can be bypassed if regolith is used instead of large boulders or rocks. The bulk material should also be magnetically separated to extract native Fe-Ni metal grains present on the Moon and some asteroids [21, 28]. Although these metal grains could be processed along with the silicates, doing so would be redundant as iron metal is an end product of the SSAP. Magnetic separation on the Moon could also separate the weakly magnetic mineral ilmenite from the silicates, which can separately be reduced into iron metal, titanium oxide and oxygen [33,34,35].

As we discussed earlier, water is possibly the most valuable resource to be harvested off-world. To avoid further complicating its extraction, water-bearing minerals and regoliths can be heated, and their evolved vapors collected in a cold trap, prior to the rest of the material being subjected to the SSAP. On the Moon, this means that water-ice rich regoliths [36] should be heated to 150 °C in order to sublime the ice into vapor [37]. For C-type asteroids, the phyllosilicates should be heated to liberate the lattice-bound OH molecules [38], which will also recrystallize much of the phyllosilicates into olivine [39]. Alternatively, if sulfuric acid were applied to water-bearing ores prior to heating, the silicates would still dissociate but the aqueous solution will be more dilute, which would likely slow the reaction rate.

2.2 Sulfuric Acid Synthesis

The sulfuric acid synthesis stage of the SSAP consists of thermal decomposition of sulfide minerals, followed by the industry-proven wet sulfuric acid process (WSA) [40], which creates sulfuric acid from sulfur-bearing gases. The main ore for sulfur on the Moon and S-type asteroids comes in the form of troilite (FeS), whereas for C-type asteroids, the slightly more sulfur-rich pyrrhotite (Fe_(1-0.8)S) is the dominant sulfide mineral [22]. For simplicity in our description we assume that the sulfides appear in their stoichiometric flavor troilite (FeS). For Mars, the main sulfur-ore consists of sulfates (e.g. MgSO₄, FeSO₄) which can be heated to release SO₂ gas and form solid oxides (discussed later in Reaction 8).

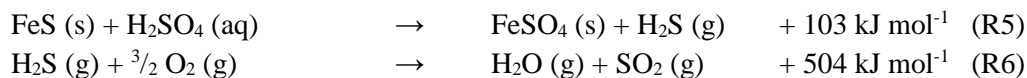


Reaction 1 shows the thermal decomposition of troilite and yields both iron metal and sulfur gas by heating to approximately 1250 °C in vacuum. More sulfur-rich minerals such as pyrrhotite and pyrite will begin releasing their sulfur component before this temperature [41]. This decomposition has been explored in meteorite heating experiments [38] which show that CM chondrites undergo a minor sulfur volatilization event around 550°C, followed by a major outgassing event at 1200 °C. In these experiments, the sulfide minerals were not separated from the rest of the meteorite sample when they were heated and formed SO₂ gas rather than pure S₂. It is unclear how much, if any of the iron in the sulfides was oxidized into Fe₂O₃ or Fe₃O₄ in this more oxygen-available environment. If the sulfide minerals were instead isolated from the rest of the bulk material before they were heated, the resultant gas would more likely be comprised of pure sulfur, while the iron within the sulfides would not likely be oxidized, resulting in pure iron metal. This iron metal is the first product harvested from the SSAP.

If pure sulfur gas is produced in Reaction 1, it must be exothermically combined with oxygen to yield sulfur dioxide (Reaction 2). If the thermal decomposition of the sulfide minerals results instead in sulfur dioxide, then Reaction 2 can be bypassed. The sulfur dioxide is then subjected to the WSA process (Reactions 3-4), whereby the gas is oxidized in the presence of a vanadium oxide catalyst between 400 and 620 °C (Reaction

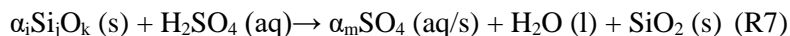
3), in order to form sulfur trioxide. It is important to note that this catalyst is not depleted during the reaction and can be reused. The resulting sulfur trioxide is then exothermically hydrated before being condensed to form highly concentrated sulfuric acid in Reaction 4. Although Reactions 2-4 require an investment of oxygen and hydrogen, we will show in later stages that they will be recovered and can be reused, such that no volatiles are lost or wasted.

Alternatively, pre-existing sulfuric acid either brought to location or created in earlier processing, can be reacted with troilite to produce iron sulfate and hydrogen sulfide gas (Reaction 5). The resulting gas can then be burned with oxygen to form water and sulfur dioxide (Reaction 6), both of which are used in the above reactions to produce sulfuric acid. The processing of the iron sulfate in Reaction 5, will be elaborated on further in the next subsection. This alternative approach may be more logistically feasible since the sulfide and silicate minerals would not need to be separated prior to their reaction with the acid. This approach would also be ideal on the martian surface, as the main sulfuric ores are various sulfates [29] that should dissolve in sulfuric acid.



2.3 Silicate Dissolution and Silica Extraction

Once the sulfuric acid is synthesized, the SSAP proceeds to the next stage: silicate dissolution. By combining the acid with the silicate minerals (Reaction 7), this stage produces silica, water, and sulfate minerals; the last of which will be broken down further in final stage of the SSAP. The reactions between the silicates and the acid are listed in Table 1.



Although we list the pure end-members of these minerals, nearly every silicate grain native to terrestrial bodies is actually a solid solution, with a varying proportion of the appropriate metal cation coexisting in the same crystal lattice. It is for simplicity that we examine the reactions of the pure end members with sulfuric acid.

A suite of previous experiments describe in detail, the acid-silicate reactions listed in Table 1 [42,43,44,45,46]. The results of these experiments show the general trend that the silicate minerals are broken down into a hydrated amorphous silica gel, while the cations (Fe, Mg, etc.) are released into the water-acid solution and eventually precipitate into their hydrated sulfate counterparts. Minor amounts of iron oxides also form from olivine when the initial aqueous sulfuric acid solution is less concentrated [43]. The mixture should be mechanically perturbed or mixed to prevent a nonreactive product layer to form on the surface of unreacted silicate grains. Since previous experiments did not mix or perturb the rock-acid mixtures, it is unclear exactly how quickly this step will progress.

Table 1. The generalized reactions between sulfuric acid and the end-members of each silicate mineral. Change in Gibbs free energy was calculated for 20 °C at 1 bar. The silica and sulfate products will both precipitate and be dissolved in solution depending the conditions of the reaction chamber. The negative values for ΔG in this table indicate that each reaction is exothermic and will proceed under standard temperature and pressure.

Reactants			Products				
Silicates (s)		Sulfuric Acid (aq)	Water (l)	Silica (s/aq)	Sulfates (s/aq)	ΔG° [kJ mol ⁻¹] (20 °C, 1 bar)	
Mineral	Endmember	Formula					
Olivine	Fayalite	Fe ₂ SiO ₄	2 H ₂ SO ₄	2 H ₂ O	SiO ₂	2 FeSO ₄	-292
	Forsterite	Mg ₂ SiO ₄	2 H ₂ SO ₄	2 H ₂ O	SiO ₂	2 MgSO ₄	-258
Pyroxene	Ferrosilite	Fe ₂ Si ₂ O ₆	2 H ₂ SO ₄	2 H ₂ O	2 SiO ₂	2 FeSO ₄	-1220
	Enstatite	Mg ₂ Si ₂ O ₆	2 H ₂ SO ₄	2 H ₂ O	2 SiO ₂	2 MgSO ₄	-270
	Wollastonite	Ca ₂ Si ₂ O ₆	2 H ₂ SO ₄	2 H ₂ O	2 SiO ₂	2 CaSO ₄	-2080
Plagioclase	Anorthite	CaAl ₂ Si ₂ O ₈	4 H ₂ SO ₄	4 H ₂ O	2 SiO ₂	CaSO ₄ , Al ₂ (SO ₄) ₃	-2250

Once all of the initial silicates have reacted with the acid, the fluid can be evaporated such that any excess water or acid can be collected for later use. The evaporation will cause the ions in the solution to precipitate into the sulfates listed in Table 1. At this point most of the solid products will still likely contain water and can be dehydrated by heating to 100 °C under vacuum. This released water can be immediately reused in Reaction 4. This step of separating water and unreacted acid from dissolved components is one of the key factors that will determine the overall efficiency of the entire SSAP in terms of energy. Adding more acid-water solution at the beginning of Reaction 7 will cause it to progress more quickly; however this also requires more energy to evaporate the remaining liquid. In later sections we estimate discuss efficiency bottlenecks for the SSAP.

2.4 Metal and Oxide Production

The final stage of the SSAP produces iron metal, oxygen, as well as metal oxides, via thermal decomposition and carbothermal reduction. The sulfates previously produced in Reaction 7 (Table 1) can be intermixed for this next step, since calcium, aluminum, magnesium and iron(II) sulfate each have distinct thermal decomposition temperatures. Heating the sulfates will decompose the iron(II) sulfate into iron(III) oxide (Fe₂O₃), sulfur dioxide, and oxygen, as shown in Reaction 8 (Table 2). Simultaneously, the aluminum sulfates will decompose into aluminum oxide, sulfur dioxide and oxygen. Since iron(III) oxide is

ferromagnetic, while aluminum oxide, calcium and magnesium sulfate are diamagnetically susceptible, the iron(III) oxide can be magnetically separated before further heating. The remaining magnesium and calcium sulfates will thermally decompose into their corresponding metal oxides at higher temperatures. The equilibrium compositions for each of these reactions is shown in Fig. 2. Further refining these oxides into pure metals (Al, Mg, Ca) is not addressed in this work; we will only describe the extraction of iron metal from iron(III) oxide.

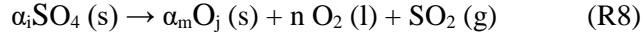
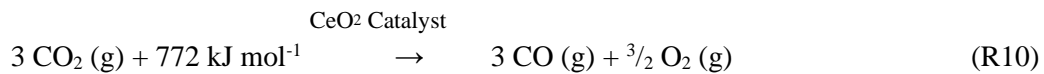
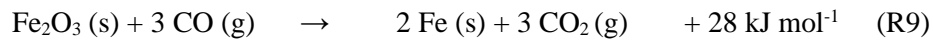


Table 2. Generalized thermal decomposition reactions for produced sulfates. The Oxides described here are in their simple form (MgO:Magnesia, Al₂O₃:Alumina). The positive values attained for ΔG° indicate that heat-energy is required for the reaction to proceed (see Fig. 2). These reactions are collectively referred to as Reaction 8.

Reactants		Products			
Sulfate (s)	→	Sulfur Dioxide (g)	Oxygen (g)	Oxide (s)	ΔG° [kJ/mol] (20 °C, 1 bar)
2 FeSO ₄		2 SO ₂	½ O ₂	Fe ₂ O ₃	306
2 MgSO ₄		2 SO ₂	O ₂	MgO	557
2 CaSO ₄		2 SO ₂	O ₂	CaO	868
Al ₂ (SO ₄) ₃		3 SO ₂	³ / ₂ O ₂	Al ₂ O ₃	621

The gases produced in Table 2 can be collected and reused in Reactions 2 and 3 to produce more sulfuric acid. Although not all the invested oxygen is recovered from iron(III) oxide the final steps will net a 1 mole surplus of O₂.

Iron metal is produced by reducing iron(III) oxide using carbon monoxide in Reaction 9. Although Chen et al. [47] show that this process consists of multiple steps: Fe₂O₃ → Fe₃O₄ → FeO → Fe, oxidizing the ambient CO atmosphere all along the way, we forgo these intermediates and represent the reaction more concisely.



The final step of the SSAP is to recover the oxygen locked away in the carbon dioxide at the end of Reaction 9, while also replenishing the supply of carbon monoxide for repeating the very same reaction. This is done by electrolyzing CO₂ with a cerium oxide catalyst near 500 °C [48], shown in Reaction 10. We would like to note that the energy required that we list in Reaction 10 is likely an overestimate since we do not consider the effects of the catalyst in our calculations. Now that the CO gas has been replenished for repeat use, 1 mole of O₂ has also been created for every 2 moles of iron metal produced, and the SSAP is complete.

3. Results and Discussion

We have calculated the theoretical yield of the SSAP for each terrestrial body listed in Table 3. We assume an initial silicate mass of one ton, with mineral chemistry and abundance representative of each particular body. By taking the product of each mineral's abundance and its end member molar percentage, we calculated the total number of moles of each silicate end member. With this, we use Reactions 7-10 to

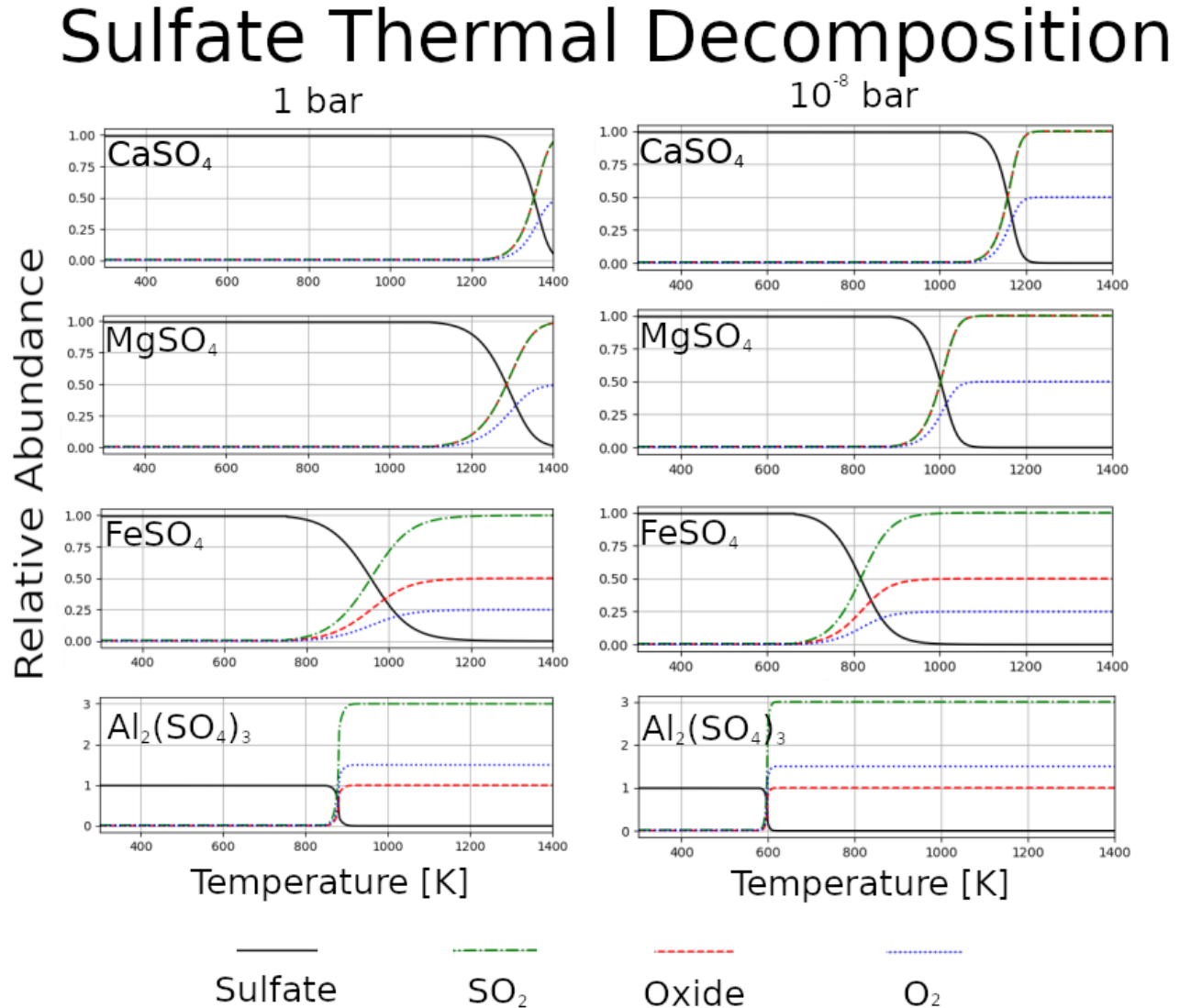


Fig. 2. The calculated equilibrium compositions for each major sulfate species at 1 bar (left column) and 10⁻⁸ bar (right column). This shows the general trend that reducing the ambient pressure lowers the temperatures required for the reactions to proceed. Note the legend at the bottom of the figure. The Y axes represent molar abundance of each compound.

calculate how many moles, and by extension kilograms, of each resource could be produced from the SSAP. For H, L, and LL chondrites (S-type asteroids) we obtained average mineral abundances from [30], and mineral compositions from [49,50,51]. For C-type asteroids we used mineral data reported by [22] for the meteorites: Murchison, Orgueil and Allende (CM, CI, CV chondrites respectively). The C and S-type asteroid calculations include contributions from native sulfides. For the lunar highlands and mare calculations, we averaged the bulk chemical compositions reported in [28] including the ilmenite and sulfide contributions. This calculation assumes that the ilmenite cannot be separated by pre-processing and is included in the silicate dissolution step (Reaction 7). For the martian calculations we averaged the values reported in [29] across both models and localities (Gusev and Meridiani), but we did not include contributions from native sulfates, which would increase the total yield of iron metal and oxygen.

Table 3. Theoretical yield for various solar system bodies, assuming 100% efficient processing of 1000 kg of native silicates. The Oxides described here are in their simple form (e.g. MgO:Magnesia, Al₂O₃:Alumina, etc.)

Body	Recoverable Resource [kg]								Reference
Luna	Fe (metal)	O₂	SiO₂	Al₂O₃	MgO	CaO	H₂O	TiO₂	
Mare	143	31	492	143	100	123	-	46	[28]
Highlands	53	13	475	237	85	142	-	8	[28]
Mars	121	35	514	124	77	96	-	-	[29]
Asteroids									
C-type (CI-Orgueil mineralogy)	65	3	430	-	436	-	118	-	[22]
C-type (CM-Murchison mineralogy)	375	125	260	-	174	-	87	-	[22]
C-type (CV-Allende mineralogy)	267	53	393	4	359	2	-	-	[22]
S-type (H Chondrite mineralogy)	132	27	509	25	323	20	-	-	[30,49,50,51]
S-type (L Chondrite mineralogy)	170	35	490	22	308	20	-	-	[30,49,50,51]

S-type (LL Chondrite mineralogy)	164	40	473	21	304	18	-	-	[30,49,50,51]
--	-----	----	-----	----	-----	----	---	---	---------------

3.1 Products and Uses

Our calculations show that a significant quantity of building material can be obtained on the Moon, Mars, and asteroids (Fig. 2). Table 3 shows that CM-like C-type asteroids are the most fruitful candidate for the SSAP, as it would produce the most iron metal and oxygen, with harvested water being an added benefit. Resource utilization on S-type asteroids would particularly benefit from the SSAP, as they would be otherwise considered relatively resource-poor, due to their lack of native water, while also hosting a relatively high abundance of sulfur. For some C-type asteroids, we list water as a SSAP-product only because their silicates contain significant native water that is incidentally released, otherwise water is not a product of the SSAP. For lunar operations, the lunar mare would be preferable to the lunar highlands in terms of a more useful product yield, since it is more highly concentrated in iron which can be used in 3D printing (discussed below). As for the longevity of SSAP operations at a destination, Mars may be the best candidate since it has abundant water ice as well as the highest relative abundances of carbon and sulfur. This would allow for higher tolerances in volatile loss.

On the lunar surface, the SSAP could considerably contribute to the Artemis program’s goals of establishing a sustainable presence on the surface on the Moon, by providing some building materials in situ. The produced iron metal will likely take the form of small particles, which can be used as the feedstock for direct metal laser sintering to 3D print components or structures including landing pads, radiation shields, or electrical wire. The silica produced here will also likely be in granular form, which can be melted in a cast to produce windows for future habitats. Alumina is a natural insulator of electricity, making it ideal for encasing power cables for extra-terrestrial solar power plants. As we mentioned earlier, alumina could be further reduced into aluminum metal, although this pathway is beyond the scope of the SSAP.

We envision a logistical framework, whereby a cargo spacecraft could deliver to a planetary surface: a metal 3D printer, a casting furnace, and a SSAP-refinery including some initial sulfuric acid, water, and carbon dioxide. The refinery could begin producing silica for the furnace and iron for the printer, to construct the hull or main body of a habitat in situ. Later cargo missions can deliver robotic workers and more specialized components such as airlock doors or life support systems to be installed. Although the mission architecture for the Artemis program prescribes an Earth-fabricated habitat, any attempts at permanent human settlement will likely require a process to create infrastructure in situ.

3.2 Engineering and Logistical Considerations

Although most of the steps within our proposed process are supported by a suite of previous experiments including some industry-proven methodologies, its effectiveness should be validated by performing these techniques on lunar, martian, and asteroid regolith simulants. These experiments will help to determine reaction rates and will characterize some of the engineering challenges that will inevitably become apparent.

The first experimental validation should be focused on the feasibility of processing the bulk material, not solely the silicates, for a given locale. For instance, the magnetic force required for collecting native iron metal and iron oxide must be determined. This is especially important for martian regolith, which contains considerable iron oxide. Once the separation is complete, the remaining non-magnetic material should be dissolved in acid (Reaction 7) to determine what problems, if any, might arise from insoluble impurities while also characterizing the reaction rate. Previous experiments [46] also indicate that Reaction 7 progresses more quickly if the temperature is slightly elevated. Keeping Reaction 7 in thermal contact with Reaction 8 (thermal decomposition) may be an efficient way to conserve energy.

The overall efficiency of the SSAP will be heavily influenced by the efficiency of Reaction 7. The less water and mechanical perturbations required to fully react the silicates (or non-magnetic material) will reduce the overall energy requirements for this step. Unfortunately we cannot accurately calculate the total energy required to enact the SSAP from start to finish, as it is unclear which hydrated sulfates would form (monohydrate, pentahydrate, etc.) in Reaction 7. Additionally, an accurate calculation would require knowledge of the power and duration needed to operate the vacuum systems in Reaction 8. These reasons underline the need for experimental investigation in the future.

As we briefly mentioned in the previous subsection, the long term viability of the SSAP will also depend on how much of the volatile compounds can be retained, especially when trapping evolved gases in Reaction 8. Having the ability to replenish volatile elements (sulfur, oxygen, carbon, hydrogen) in situ will alleviate strict leak tolerances for the processing equipment. This makes Mars and C-type asteroids more forgiving in terms of volatile loss, while the Moon and S-type asteroids are relatively volatile poor and therefore less forgiving. Striking a balance between chamber pressure and heating in Reaction 8 will also influence energy efficiency and volatile loss. As Fig. 2 shows, decreasing overbearing pressure during sulfate decomposition will also decrease the temperatures required to drive off the sulfur-bearing gases, but will also require a more robust and energy intensive vacuum system. Validation experiments should explore this balance to determine which pressure and temperature profiles are most efficient to fully decompose the sulfates.

3.3 Comparison to other ISRU Methodologies

Although we cannot accurately calculate the energy requirements of the SSAP, we can qualitatively compare its inherent strengths and weaknesses to a collection of other ISRU techniques: molten salt electrolysis (modified FCC Cambridge Process), vapor phase pyrolysis, and hydrogen reduction. An inherent advantage that all of these methods have over the SSAP, is their experimental characterization.

Molten salt electrolysis [52,53], specifically one modified for lunar surface operations [54] can essentially reduce all oxides and silicates into metallic alloys, while also releasing nearly all the oxygen present. This benefit comes at the cost of the required operating temperature. While the SSAP will need to reach similar operating temperatures as this approach (~950 °C), this is only the peak temperature required, while salt electrolysis will need to maintain this temperature for the duration of the process. Special consideration will also need to be taken when choosing which salt and anode to use such that they are not depleted or corroded respectively, though this concern is relatively minor.

Like salt electrolysis, vapor phase pyrolysis [55,56,57] can also reduce the native minerals into metals while also liberating all the oxygen. A major benefit of pyrolysis is its relative simplicity: heating up the rocks they vaporize. This heating however is also the major drawback, since the temperatures required are in excess of 2000 °C.

Another approach for reducing minerals into metals, while also harvesting oxygen involves using hydrogen [34] on the bulk material. This process works mostly on ilmenite, reducing it into titanium oxide and iron metal, while releasing some oxygen (~5 wt% [58]) at the same time. Unfortunately this approach has a lesser effect on silicates. This approach also requires operating temperatures which are slightly higher than the SSAP's peak temperature. An advantage to this approach is its simplicity compared to the SSAP and could conceivably be performed as a pre-processing step for the SSAP.

4. Conclusions

In this paper, we have presented the physical and chemical pathways for the proposed Silicate-Sulfuric Acid Process, which aims to manufacture building materials from abundant resources found in situ on major planetary bodies such as the Moon, Mars and asteroids. Although this approach has not yet been tested, it allows for inherent recycling of volatile elements, such that little to no material must be supplied after the initial investment. This proposed ISRU approach could provide substantial building materials including iron metal for 3D printing, silica for window construction, as well as modest amounts of oxygen gas. The next step for further investigating the utility of the SSAP lies in experimental validation.

5. Acknowledgments

We would like to thank our anonymous reviewers for their insightful suggestions that considerably improved the scope and quality of this manuscript. This work was funded by the Commonwealth Scientific and Industrial Research Organisation's Top Up Scholarship (No. 50077118).

6. References

- [1] B. O’Leary, Mining the Apollo and Amor asteroids, *Science* 197 (1997) 363-366.
- [2] F.E. Meyen, M.H. Hecht, J.A. Hoffman and MOXIE Team. Thermodynamic model of Mars ISRU experiment (MOXIE), *Acta Astronautica* 129 (2016) 82-87.
- [3] M. Anand, I.A. Crawford, M. Balat-Pichelin, S. Abanades, W. Van Westrenen, G. Péraudeau, R. Jaumann, W. Seboldt, A brief review of chemical and mineralogical resources on the Moon and likely initial in situ resource utilization (ISRU) applications, *Planetary and Space Science* 74 (2012) 42-48.
- [4] P.T. Metzger, A. Muscatello, R.P. Mueller, J. Mantovani, Affordable, Rapid bootstrapping of the space industry and solar system civilization, *Journal of Aerospace Engineering*, 26 (2013) 18-29.
- [5] D. Kornuta, A. Abbud-Madrid, J. Atkinson, J. Barr, G. Barnhard, D. Bienhoff, B. Blair, V. Clark, J. Cyrus, B. DeWitt, C. Dreyer, Commercial lunar propellant architecture: A collaborative study of lunar propellant production, *Reach* 13 (2019) 100026.
- [6] A.M. Hein, M. Saidani, H. Tollu, Exploring potential environmental benefits of asteroid mining, arXiv preprint arXiv:1810.04749 (2018).
- [7] N.J. Lindsey, Lunar station protection: Lunar regolith shielding, *Science and Technology Series*, 108 (2004) 143-148.
- [8] S.D. Rosenberg, R.L. Beegle, G.A. Guter, F.E. Miller, M. Rothenberg, The onsite manufacture of propellant oxygen from lunar resources, NASA Johnson Space Center, *Space Resources Volume 3: Materials* (1992).
- [9] F. Chandler, D. Bienhoff, J. Cronick, G. Grayson, Propellant Depots for Earth Orbit and Lunar Exploration, AIAA SPACE 2007 Conference & Exposition (2007).
- [10] R.L. Ash, W.L. Dowler, G. Varsi, Feasibility of rocket propellant production on Mars, *Acta Astronautica*, 5 (1978) 705-724.
- [11] A. Muscatello, R. Devor, J. Captain, Atmospheric processing module for Mars propellant productions, *Earth and Space* (2018) 444-454.
- [12] J.E. Kleinhenz, A. Paz, An ISRU propellant production system for a fully fueled Mars Ascent Vehicle, 10th Symposium on Space Resource Utilization (2017) 423.
- [13] D. Vaniman, D. Pettit, G. Heiken, Uses of lunar sulfur, *Lunar Bases and Space Activities of the 21st Century* (1998).

- [14] R.N. Grugel, H. Toutanji, Sulfur “concrete” for lunar applications–Sublimation concerns, *Advances in Space Research* 41 (2008) 103-112.
- [15] H.A. Toutanji, S. Evans, R.N. Grugel, Performance of lunar sulfur concrete in lunar environments, *Construction and Building Materials*, 29 (2012) 444-448.
- [16] L.A. Taylor, W.D. Carrier III, Production of oxygen on the Moon: Which processes are best and why, *AIAA Journal* 30(12) (1992) 2858-2863.
- [17] C. Schwandt, J.A. Hamilton, D.J. Fray, I.A. Crawford, The production of oxygen and metal from lunar regolith, *Planetary and Space Science* 74(1) (2012) 49-56.
- [18] M. J. Gaffey, Rotational spectral variations of asteroid (8) Flora: Implications for the nature of the S-type asteroids and for the parent bodies of the ordinary chondrites, *Icarus* 60 (1984) 83-114.
- [19] T. Nakamura, T. Noguchi, M. Tanaka, M.E. Zolensky, M. Kimura, A. Tsuchiyama, A. Nakato, T. Ogami, H. Ishida, M. Uesugi, T. Yada, Itokawa dust particles: a direct link between S-type asteroids and ordinary chondrites, *Science* 333 (6046) (2011) 1113-1116.
- [20] T. Hiroi, M.E. Zolensky, C.M. Pieters, M.E. Lipschutz, Thermal metamorphism of the C, G, B, and F asteroids seen from the 0.7 μm , 3 μm , and UV absorption strengths in comparison with carbonaceous chondrites, *Meteoritics & Planetary Science* 31 (3) (1996) 321-327.
- [21] O.N. Menzies, P.A. Bland, F.J. Berry, G. Cressey, A Mössbauer spectroscopy and X-ray diffraction study of ordinary chondrites: Quantification of modal mineralogy and implications for redox conditions during metamorphism, *Meteoritics & Planetary Science* 40 (7) (2005) 1023-1042.
- [22] P.A. Bland, G. Cressey, O.N. Menzies, Modal mineralogy of carbonaceous chondrites by X-ray diffraction and Mössbauer spectroscopy, *Meteoritics & Planetary Science* 39 (1) (2004) 3-16.
- [23] S.R. Taylor, S. McLennan, *Planetary crusts: their composition, origin and evolution* (Vol. 10), Cambridge University Press (2009).
- [24] D.L. Blaney, T.B. McCord, Indications of sulfate minerals in the Martian soil from Earth-based spectroscopy, *Journal of Geophysical Research: Planets* 100 (E7) (1995) 14433-14441.
- [25] M.D. Lane, M.D. Dyar, J.L. Bishop, Spectroscopic evidence for hydrous iron sulfate in the Martian soil, *Geophysical Research Letters* 31 (19) (2004).
- [26] E.K. Gibson, G.W. Moore, Sulfur abundances and distributions in the valley of Taurus-Littrow, *Lunar and Planetary Science Conference Proceedings Vol. 5* (1974) 1823-1837.

- [27] A.A. Berezhnoy, N. Hasebe, T. Hiramoto, B.A. Klumov, Possibility of the presence of S, SO₂, and CO₂ at the poles of the Moon, *Publications of the Astronomical Society of Japan* 55 (4) (2003) 859-870.
- [28] L. Haskin, P. Warren, 8. Lunar Chemistry, in: G.H. Heiken, D.T. Vaniman, B.M. French, *Lunar Sourcebook, a user's guide to the Moon*, Cambridge University Press, New York, 1991, pp. 357-475.
- [29] H.Y. McSween, I.O. McGlynn, A.D. Rogers, Determining the modal mineralogy of Martian soils, *Journal of Geophysical Research: Planets* 115 (E7) (2010).
- [30] T.L. Dunn, T.J. McCoy, J.M. Sunshine, H.Y. McSween, A coordinated spectral, mineralogical, and compositional study of ordinary chondrites, *Icarus* 208 (2) (2010) 789-797.
- [31] J.A. Salem, Transparent armor ceramics as spacecraft windows, *Journal of the American Ceramic Society* 96 (1) (2013) 281-289.
- [dataset] [32] Thermodynamics Research Center, 'Thermodynamics Source database' in NIST Chemistry WebBook, NIST Standard Reference Database Number 69, Eds P.J. Linstrom, W.G. Mallard, National Institute of Standards and Technology, Gaithersburg MD, 20899 <https://doi.org/10.18434/T4D303>.
- [33] A.H. Cutler, P. Krag, A carbothermal scheme for lunar oxygen production, *Lunar bases and space activities of the 21st century* (1985) 559.
- [34] M.A. Gibson, C.W. Knudsen, Lunar oxygen production from ilmenite, *Lunar bases and space activities of the 21st century* (1985) 543.
- [35] J. Pesl, R.H. Eric, High temperature carbothermic reduction of Fe₂O₃-TiO₂-M_xO_y oxide mixtures, *Minerals Engineering* 15 (11) (2002) 971-984.
- [36] C.I. Honniball, P.G. Lucey, S. Li, S. Shenoy, T.M. Orlando, C.A. Hibbitts, D.M. Hurley, and W.M. Farrell, Molecular water detected on the sunlit Moon by SOFIA, *Nature Astronomy* (2020): 1-7.
- [37] U. Hegde, R. Balasubramaniam, S. Gokoglu, Analysis of water extraction from lunar regolith, 50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition (2012) 634.
- [38] A.B. Verchovsky, M. Anand, S.J. Barber, S. Sheridan, G.H. Morgan, A quantitative evolved gas analysis for extra-terrestrial samples, *Planetary and Space Science* 181 (2020) 104830.
- [39] N.H. Brett, K.J.D. MacKenzie, J.H. Sharp, The thermal decomposition of hydrous layer silicates and their related hydroxides, *Quarterly Reviews, Chemical Society*, 24 (2) (1970) 185-207.

- [40] J.K. Laursen, The process principles, *Hydrocarbon Engineering*, 12 (8) 47-50.
- [41] M. Thompson, J.J. Fortney, M. Telus, D. Lederman, T. Joshi, Meteorite Outgassing Experiments to Inform Chemical Abundances of Super-Earth Atmospheres, AGUFM, (2019) P23B-3487.
- [42] J. Van Herk, H.S. Pietersen, R.D. Schuiling, Neutralization of industrial waste acids with olivine—The dissolution of forsteritic olivine at 40–70 C, *Chemical Geology* 76 (3-4) (1989) 341-352.
- [43] R.C.L. Jonckbloedt, Olivine dissolution in sulphuric acid at elevated temperatures—implications for the olivine process, an alternative waste acid neutralizing process, *Journal of Geochemical Exploration*, 62 (1-3) (1998) 337-346.
- [44] H.E. King, O. Plümper, T. Geisler, A. Putnis, Experimental investigations into the silicification of olivine: Implications for the reaction mechanism and acid neutralization, *American Mineralogist*, 96 (10) (2011) 1503-1511.
- [45] A. Lazaro, H.J.H. Brouwers, G. Quercia, J.W. Geus, The properties of amorphous nano-silica synthesized by the dissolution of olivine, *Chemical Engineering Journal* 211 (2012) 112-121.
- [46] E.C. Marcucci, B.M. Hynek, Laboratory simulations of acid-sulfate weathering under volcanic hydrothermal conditions: Implications for early Mars, *Journal of Geophysical Research: Planets*, 119 (3) (2014) 679-703.
- [47] H. Chen, Z. Zheng, Z. Chen, X.T. Bi, Reduction of hematite (Fe₂O₃) to metallic iron (Fe) by CO in a micro fluidized bed reaction analyzer: A multistep kinetics study, *Powder Technology* 316 (2017) 410-420.
- [48] T.L. Skafte, Z. Guan, M.L. Machala, C.B. Gopal, M. Monti, L. Martinez, E. Stamate, S. Sanna, J.A.G. Torres, E.J. Crumlin, M. García-Melchor, Selective high-temperature CO₂ electrolysis enabled by oxidized carbon intermediates, *Nature Energy* 4 (10) (2019) 846-855.
- [49] A.E. Rubin, Kamacite and olivine in ordinary chondrites: Intergroup and intragroup relationships, *Geochimica et Cosmochimica Acta* 54 (5) (1990) 1217-1232.
- [50] R.T. Dodd, W.R. Van Schmus, D.M. Koffman, A survey of the unequilibrated ordinary chondrites, *Geochimica et Cosmochimica Acta* 31 (6) (1967) 921-951.
- [51] R. Kessel, J.R. Beckett, E.M. Stolper, The thermal history of equilibrated ordinary chondrites and the relationship between textural maturity and temperature, *Geochimica et Cosmochimica Acta* 71 (7), (2007) 1855-1881.
- [52] D.J. Fray, G.Z. Chen, Reduction of titanium and other metal oxides using electroreduction,

Materials science and technology 20 (3) (2004) 295-300.

[53] G.Z. Chen, D.J. Fray, T.W. Farthing, Direct electrochemical reduction of titanium dioxide to titanium in molten calcium chloride, *Nature* 407 (6802) (2000) 361-364.

[54] B.A. Lomax, M. Conti, N. Khan, N.S. Bennett, A.Y. Ganin, M.D. Symes, Proving the viability of an electrochemical process for the simultaneous extraction of oxygen and production of metal alloys from lunar regolith, *Planetary and Space Science* 180 (2020) p.104748.

[55] W. Steurer, Vapor phase pyrolysis, *Space Resources* 3 (1992) 210-213

[56] C. Senior, Lunar oxygen production by pyrolysis, *Space Programs and Technologies Conference* (1993) pp. 1663

[57] J.P. Matchett, B.R. Pomeroy, E.H. Cardiff, An oxygen production plant in the lunar environment: A vacuum pyrolysis approach, *Space Resources Roundtable VII: LEAG Conference on Lunar Exploration*, Vol 1287 (2005).

[58] H.M. Sargeant, F.A.J Abernathy, S.J. Barber, I.P. Wright, M. Anand, S. Sheridan, A. Morse, Hydrogen reduction of ilmenite: Towards an in situ resource utilization demonstration on the surface of the Moon, *Planetary and Space Science* 180 (2020) p. 104751.

[59] V. Swamy, L.S. Dubrovinsky, Thermodynamic data for the phases in the CaSiO₃ system, *Geochimica et Cosmochimica Acta* 61:6 (1997) 1181-1191.

[60] R.M. Garrels, C.L. Christ, *Solutions, Minerals and Equilibria*. Ny, Ny: Harper & Row p403 (1965)

[61] H. Zhu, R.C. Newton, O.J. Kleppa, Enthalpy of formation of Wollastonite (CaSiO₃) and anorthite (CaAl₂Si₂O₈) by experimental phase equilibrium measurements and high-temperature solution colorimetry, *American Mineralogist* 79 no. 1-2 (1994) 134-144.

[62] D.R. Lide, *CRC handbook of chemistry and physics: a ready-reference book of chemical and physical data*, 84th edition, CRC Press, Boca Raton (2004)

[63] L.M. Anovitz, A.H. Treiman, E.J. Essene, B.S. Hemingway E.F. Westrum, V.J. Wall, R. Burriel, S.R. Bohlen, The heat-capacity of ilmenite and phase equilibria in the system Fe-TO, *Geochimica et Cosmochimica Acta* 49 (10) (1985) 2027-2040.

[64] R.A. Robie, B.C. Finch, B.S. Hemingway, Heat capacity and entropy of fayalite (Fe₂SiO₄)

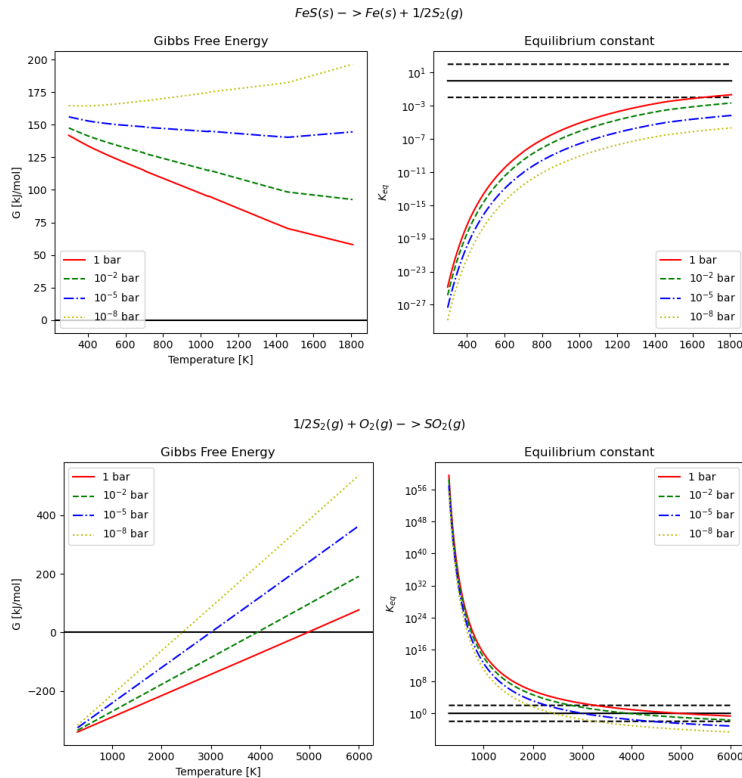
between 5.1 and 383 K: comparison of calorimetric and equilibrium values for the QFM buffer reaction, American Mineralogist 67 (5-6) (1982) 463-469.

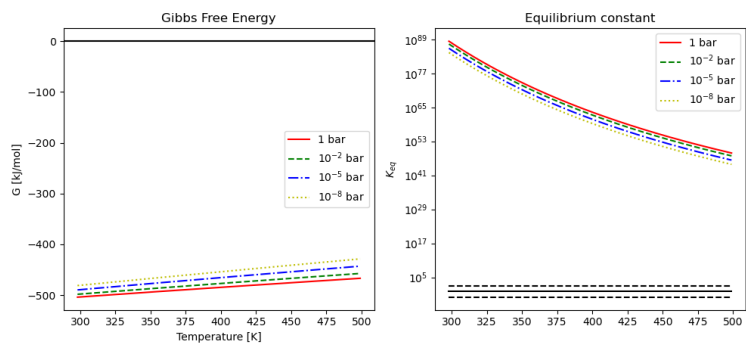
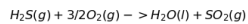
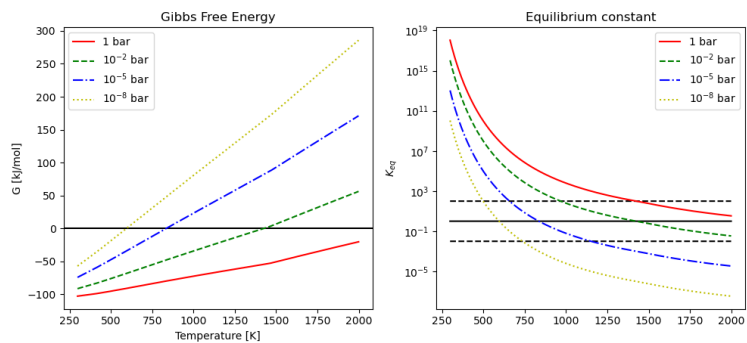
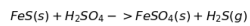
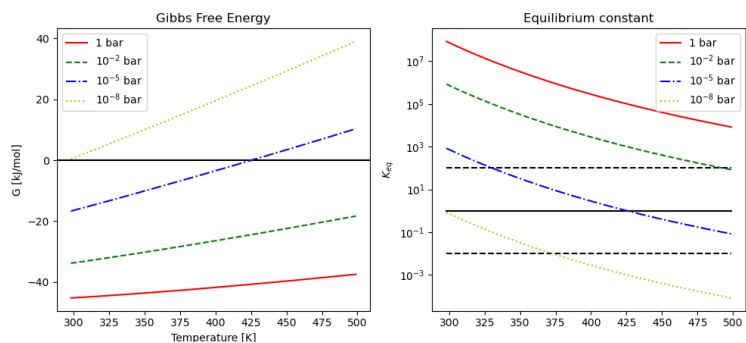
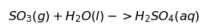
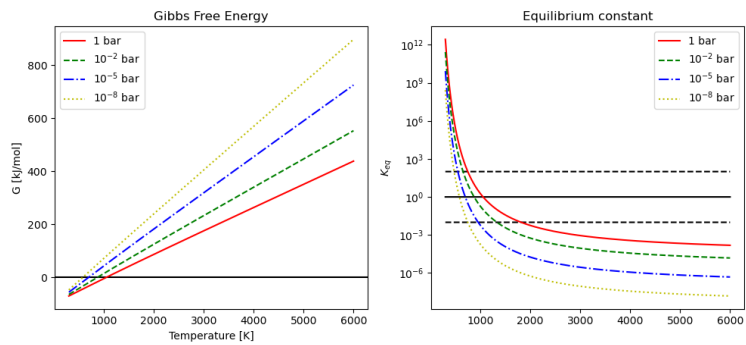
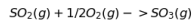
[65] L. Cemic, E. Dachs, Heat capacity of ferrosilite, Fe₂Si₂O₆, Physics and chemistry of minerals 33(7) (2006) 457-464.

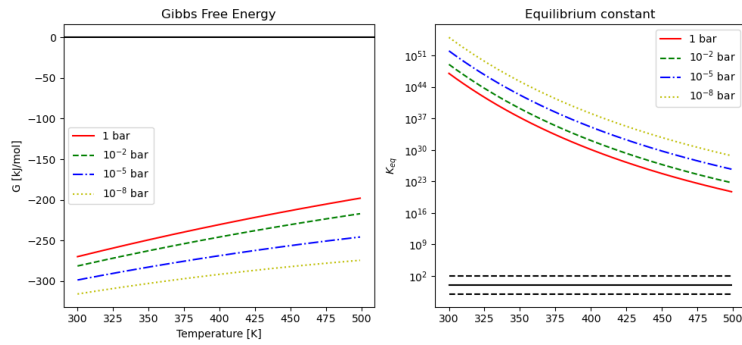
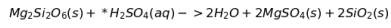
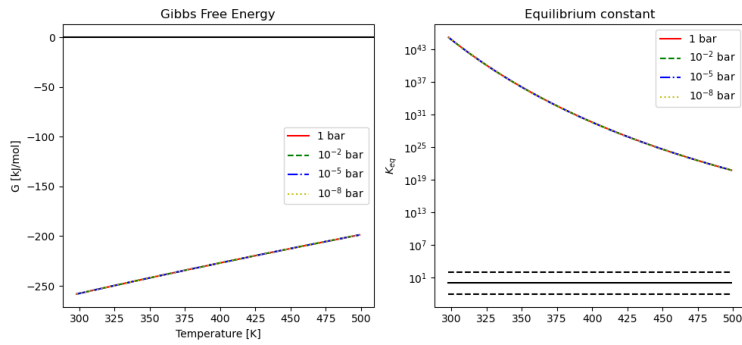
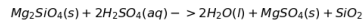
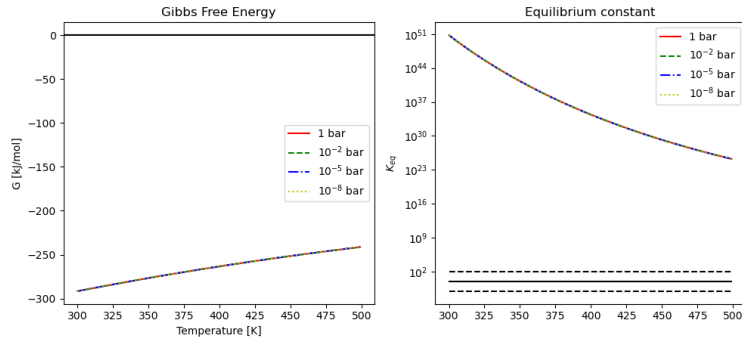
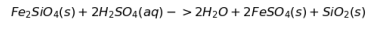
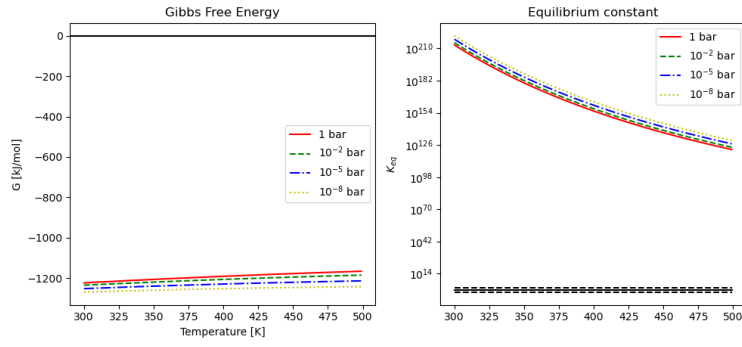
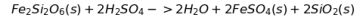
Supplementary Materials

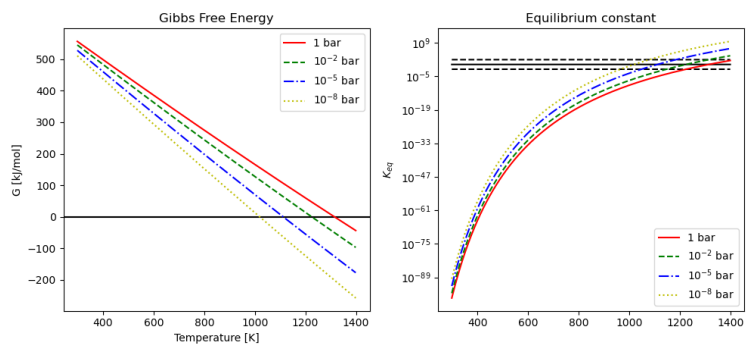
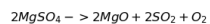
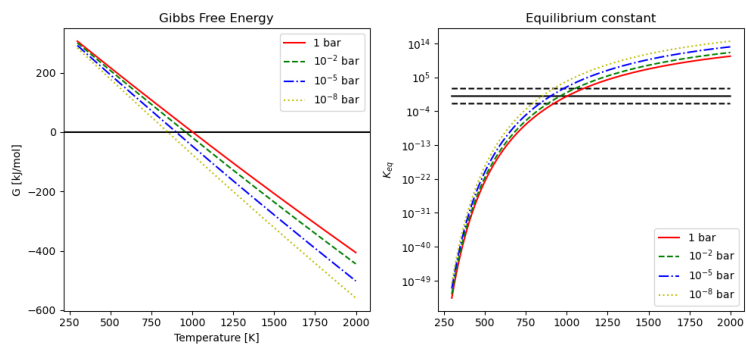
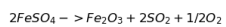
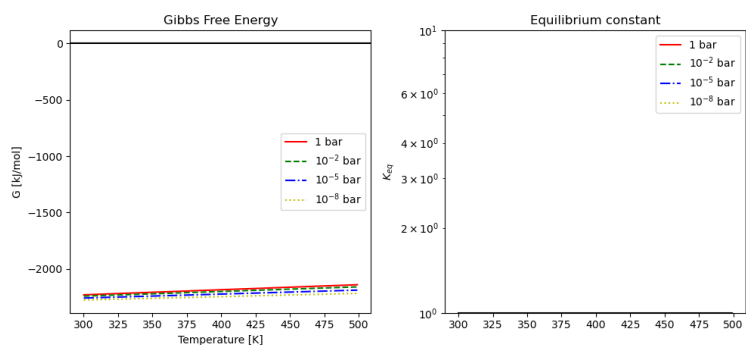
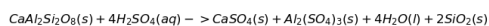
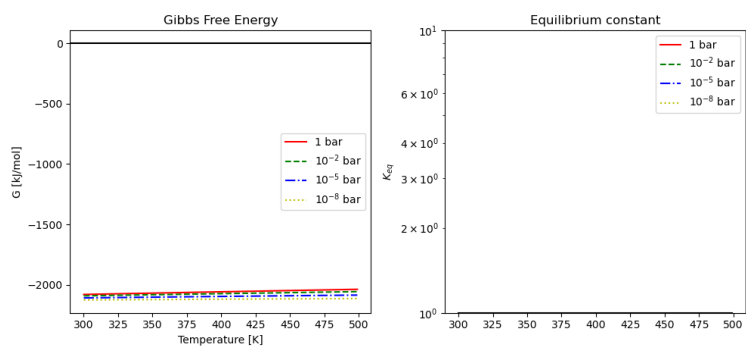
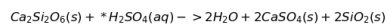
Anderson et al. (2021) “The Silicate-Sulfuric Acid Process: Mineral Processing for In Situ Resource Utilization (ISRU)”

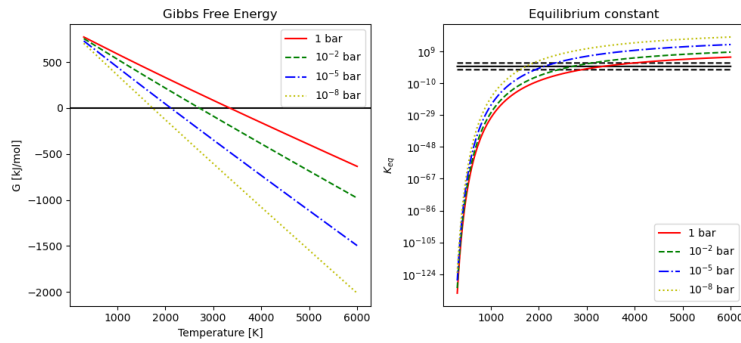
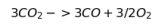
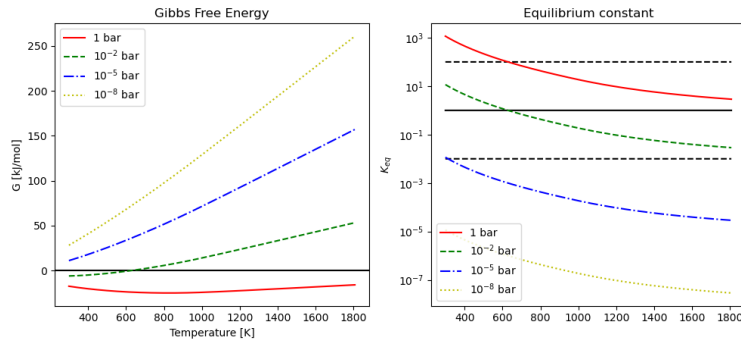
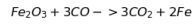
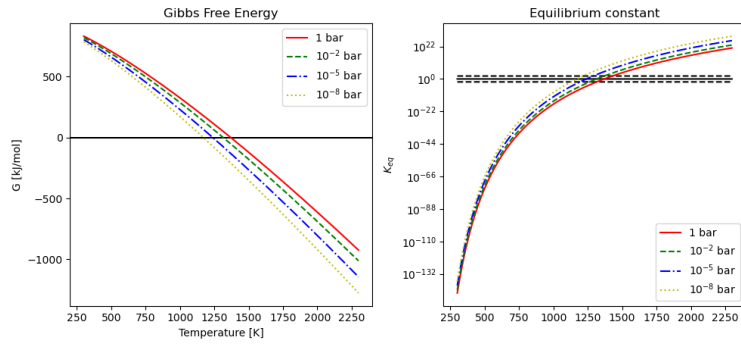
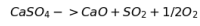
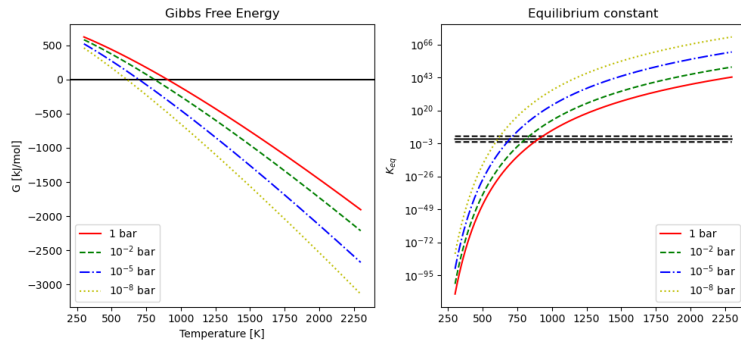
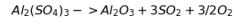
SM Figs. 1-18. Gibbs Free energy and Equilibrium Constant plots for each reaction. In each equilibrium constant graph we plot 3 vertical lines: 1 solid black at $K_{eq} = 1$, and 2 dotted black lines at $K_{eq} = 0.01$; 100. These arbitrarily show where the reaction will contain significant amount of both reactants and products.











SM Table 1. List of sources for thermodynamic data. All compounds not listed here were sourced from NIST Chemistry Webook [32].

Compound	Reference
CaSiO ₃	[59]
Al ₂ (SO ₄) ₃	[60]
CaAl ₂ Si ₂ O ₈	[61]
CaSO ₄	[62]
FeTiO ₃	[63]
Fe ₂ SiO ₄	[64]
Fe ₂ Si ₂ O ₆	[65]

[59] V. Swamy, L.S. Dubrovinsky, Thermodynamic data for the phases in the CaSiO₃ system, *Geochimica et Cosmochimica Acta* 61:6 (1997) 1181-1191.

[60] R.M. Garrels, C.L. Christ, *Solutions, Minerals and Equilibria*. Ny, Ny: Harper & Row p403 (1965)

[61] H. Zhu, R.C. Newton, O.J. Kleppa, Enthalpy of formation of Wollastonite (CaSiO₃) and anorthite (CaAl₂Si₂O₈) by experimental phase equilibrium measurements and high-temperature solution colorimetry, *American Mineralogist* 79 no. 1-2 (1994) 134-144.

[62] D.R. Lide, *CRC handbook of chemistry and physics: a ready-reference book of chemical and physical data*, 84th edition, CRC Press, Boca Raton (2004)

[63] L.M. Anovitz, A.H. Treiman, E.J. Essene, B.S. Hemingway E.F. Westrum, V.J. Wall, R. Burriel, S.R. Bohlen, The heat-capacity of ilmenite and phase equilibria in the system Fe-TO, *Geochimica et Cosmochimica Acta* 49 (10) (1985) 2027-2040.

[64] R.A. Robie, B.C. Finch, B.S. Hemingway, Heat capacity and entropy of fayalite (Fe₂SiO₄) between 5.1 and 383 K: comparison of calorimetric and equilibrium values for the QFM buffer reaction, *American Mineralogist* 67 (5-6) (1982) 463-469.

[65] L. Cemic, E. Dachs, Heat capacity of ferrosilite, Fe₂Si₂O₆, *Physics and chemistry of minerals* 33(7) (2006) 457-464.

Chapter VII: Conclusions

Work Done

The work outlined in this thesis has completed three main accomplishments: 1) The recovery of an observed meteorite fall using drones and machine learning; 2) Characterizing an orbital meteorite recovered by the Desert Fireball Network; 3) Devise a new methodology for harvesting useable resources on the Moon, Mars and Asteroids. The first accomplishment has alleviated the final bottleneck for meteor(oid/ite) science in the Desert Fireball Network, allowing us to now recover meteorites at much faster rate, and at a far lower cost of both time and effort. Where before, an orbital meteorite would be recovered with 300 labor-days of effort, we reduced this to 12 labor-days (though this may be an optimistic estimate based on only 1 attempt, which happened to be a success). Another advantage with this approach is its portability to other meteorite fall locations, since it relies on easily obtained, local image data to train the neural network to peak performance. The second accomplishment outlined in this work, has allowed meteoriticists another perspective viewing the dynamical and geochemical nature of small bodies and asteroidal debris in the solar system. Although Murrili inhabits a well-studied place in meteoritics (equilibrated H chondrites), it constitutes another brick laid into the foundation of understanding how small debris is created, then transferred all across the solar system. Its anomalous impact-shock features also indicate a lack of current understanding regarding how impact shock pressure is propagated through the fabric of chondritic meteorites. The third accomplishment that I have presented here, outlines and explores the possible uses of extra-terrestrial sulfur, specifically when it is repurposed as sulfuric acid. In that chapter I explore often overlooked components of extra-terrestrial ores that would prove to be invaluable in the context of a space-based economy.

Work To Do

Although this thesis contains a plethora of completed works and new techniques, like any good science project, the work has really just begun. For the new meteorite searching approach, a few bottlenecks still remain, the most prohibitive of which involves verifying 3rd stage candidates. The current system requires a drone pilot to manually type a list of GPS coordinates (the locations of meteorite candidates) into the GS Pro app before launching the automated flight mode, in which the smaller Mavic Pro visits each waypoint for a follow up inspection. This pre-flight setup often takes nearly as long as the flight itself and does not currently feature a shortest-path or 'traveling salesperson' algorithm to minimize flight time. Furthermore, our searching framework would greatly benefit from increased person-mobility in the context of 4th stage meteorite candidate elimination. Considering that current team members must walk, sometimes more than 1 km, to make an in-person visit to finally confirm or eliminate a candidate's meteorite status, an electric mountain bike with reinforced tires would easily expedite this final step. Adjustments should also be made to reduce the number of false positives users must sort, while also optimizing the machine learning code such that data processing would ideally keep in pace with data collection. Other small improvements can also be made to the python code which makes the entire system possible. Improvements such as a user interface that allows users to run particular pieces of code and process data would massively expand the accessibility of this system to other meteorite hunters around the globe.

The characterization of collected meteorites will continue and continue to advance our understanding of small debris in near earth space. Although ordinary chondrites are unfairly considered by some meteoriticists to be 'boring', constructing a dataset with in-depth details from a wide array of analytical techniques, across dozens or hundreds of these extraordinary chondrites will undoubtedly illuminate the

connection between meteorites and their asteroidal origins, and could unravel secrets of low-gravity impact processing.

The Silicate Sulfuric Acid Process awaits the next obvious step: experimental characterization. An experiment campaign targeting ordinary chondrites, carbonaceous chondrites, and Martian and Lunar regolith simulants would first characterize which minerals could be magnetically separated from the rest of the mostly silicate ore. The next step is the most crucial, where we would characterize the efficiency at which sulfuric acid dissolves the remaining non-magnetic ore. From there, we would explore the outgassing behaviour of the sulfate products and specifically try to decompose iron(II) sulfate into iron oxide, for further reduction into iron metal.

Appendix A (Meteorite Searching Code)

```
# -*- coding: utf-8 -*-  
''''
```

Created on Thu Sep 6 22:26:21 2018

@author: seamus (seamus.anderson@postgrad.curtin.edu.au)

This is the main function library for the machine learning side of the drone searching process.

Function overviews:

img_splitter(IMG, shape):

input This function takes an image and splits it into more images, specified by the 'shape' and returns a list containing the smaller patches of the larger image.

predict_and_highlight(model_path, folder, input_shape, pred_thres=0.9, silent=True):

input This function takes an image and splits it into more images, specified by the 'shape' and returns a list containing the smaller patches of the larger image.

list_files(folder, f_type='.jpg'):

Takes a folder and file extension, and returns all the filenames (fullpaths) in that directory with that extension.

add_mets_to_tiles(met_folder, tile_folder, dest_folder, resize_lim=5):

it This takes a folder of meteorite images and a folder of tile images. For each tile image, randomly selects a meteorite and overlays it onto the tile with a random orientation, random size at a random location. Also saves the images into a given destination folder.

**make_synt_set(img_folder,
met_folder,**

```
save_folder,  
img_shape = (100,100),  
met_size_lims = (20,50) ):
```

Makes a synthetic training set, using a folder with background images, folder of meteorite images, and name of where to save the new images.

```
''''''
```

```
import pandas as pd  
import matplotlib.pyplot as plt
```

```
import os  
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'  
os.environ['TF_ENABLE_AUTO_MIXED_PRECISION'] = '1'  
import sys  
import csv  
import PIL  
import time  
import keras  
import shutil
```

```
import datetime  
import numpy as np  
#import matplotlib.pyplot as plt  
from datetime import datetime as dtm  
from PIL import ImageDraw, ImageFont, ImageFilter, ImageEnhance  
from os.path import split as osplit  
from os.path import join as ojoin
```

```
from keras import regularizers  
from keras.models import Sequential  
from keras.layers import Conv2D, MaxPooling2D, Activation, Dropout, Flatten, Dense,  
normalization, AveragePooling2D
```

```
from keras.preprocessing.image import ImageDataGenerator  
from keras.callbacks import ModelCheckpoint, EarlyStopping  
import multiprocessing as mp  
from multiprocessing import Pool
```

```
import random
```

```
# This allows for other process to run on the GPU
```



```

    # adds them to the return list 'files', if file extension matches
    for i in range(len(contents)):
        if(f_type.upper() in contents[i].upper()):
            files.append(os.path.join(folder, contents[i]))
        if((exclude != None) and (exclude.upper() in contents[i].upper())):
            files.remove(os.path.join(folder, contents[i]))

    return sorted(files)

```

```

#####
#####
#####
#####
#####
#####
#####

```

def copy_files(src_dir, dest_dir, num, prefix='', contains=None, random=True, move=False):
 "Simple copy function (great for revolver training), it randomly grab files from a dir and
 copies
 them to a new one. Option to move instead of copy.

Inputs:

src_dir	[str]	directory with images to copy/move
dest_dir	[str]	directory images are copied to
num	[int]	number of files to move
prefix	[str]	Optional add string to front of copied filename
contains	[str]	Optional check for string in image names
random	[bool]	True if you want to randomly select
move	[bool]	True if you want to move instead of copy

Outputs:

[void]

Dependencies:

list_files()

```

'''

if(contains != None):
    fnames = list_files(src_dir, contains)
else:
    fnames = list_files(src_dir)

    # Make dest dir
if(not os.path.exists(dest_dir)):
    os.mkdir(dest_dir)

    # Optional shuffle
if(random == True):
    np.random.shuffle(fnames)

if(num > len(fnames)):
    num = len(fnames)

    # Actually move
for i in range(num):
    dest_fname = os.path.join(dest_dir, (prefix + os.path.split(fnames[i])[1]))

    if(move == True):
        os.rename( fnames[i], dest_fname)

    else:
        shutil.copyfile(fnames[i], dest_fname)

return

```

```

#####
#####
#####
#####
#####
#####
#####

```

```
def img_splitter(img_array, tile_shape=(125,125), stride=70, img_num=None):
    '''This function takes an array and returns tiles of size 'tile_shape' with overlap characterized
    by 'stride'.
```

This is usually used for `make_synth_set()` and `predict_and_highlight()`.

Inputs:

`img_array` [array] the image to be split, in array format
`tile_shape` [tuple] size (y, x) of the tiles to be created
`stride` [int] number of pixels to stride over in each dimension, before

making a new tile

Outputs:

`tiles` [list] each element contains a 2d array, which is the cropped

tile

`xys` [list] each element contains a 1d array, which is the corners

[y1,y2,x1,x2] of the tile in the original image

```
'''
```

```
    # Don't mess with the original image
```

```
    img = np.copy(img_array)
```

```
    # Preparing the returns
```

```
    tiles = []
```

```
    xys = []
```

```
    # Window width for x and y
```

```
    wy = tile_shape[0]
```

```
    wx = tile_shape[1]
```

```
    # Number of times to stride through the image
```

```
    y_num = (img.shape[0] // stride) - (wy // stride)
```

```
    x_num = (img.shape[1] // stride) - (wx // stride)
```

```
    # Keep track of image number (deprecated)
```

```
    if(img_num == None):
```

```
        img_num = 0
```

```
    # Start splitting from the top right corner of the image
```

```
    # splitting most of the image
```

```
    for i in range(y_num):
```

```
        y1 = i * stride
```

```
        y2 = i * stride + wy
```

```

    for j in range(x_num):
        x1 = j * stride
        x2 = j * stride + wx

        tiles.append( img[y1:y2, x1:x2])
        xys.append(np.array([y1,y2, x1,x2, img_num]))

    # Right edge of the image
    for i in range(y_num):
        x1 = img.shape[1] - wx
        x2 = img.shape[1]
        y1 = (i * stride) + (img.shape[0] - ((y_num - 1) * stride + wy))
        y2 = y1 + wy

        tiles.append( img[y1:y2, x1:x2])
        xys.append(np.array([y1,y2, x1,x2, img_num]))

    # Bottom edge of the image
    for i in range(x_num):
        x1 = (i * stride) + (img.shape[1] - ((x_num - 1) * stride + wx) )
        x2 = x1 + wx
        y1 = img.shape[0] - wy
        y2 = img.shape[0]

        tiles.append( img[y1:y2, x1:x2])
        xys.append(np.array([y1,y2, x1,x2, img_num]))

return tiles, xys

```

```

#####
#####
#####
#####
#####
#####
#####

```

```
def make_false_set(set_dir, dest_dir, stride=125, tile_shape=(125,125), img_lim=60):
```

```
'''
```

This function makes false tiles out of whole images. Within 'set_dir' there should be a subdir 'false_imgs', which contains images with no meteorites.

There can also be a file within 'set_dir' called 'false_regions.txt' with callouts for false regions in the images in 'set_dir'. Format should reflect ImageJ output

Inputs:

```
set_dir      [str]  fullpath to dir where source images are stored
dest_dir     [str]  fullpath to dir where the tiles will be saved
stride      [int]  stride in which the kernel moves over the image
```

(current presets allow for no overlap)

```
tile_shape   [tuple] shape of output tiles (2D)
img_lim     [int]  limit to the number of fully false images to use for tiles (some
```

sets have >400 false_imgs)

Outputs:

```
void
```

```
'''
```

```
# Create output dir if it doesn't exist
```

```
if(not os.path.exists(dest_dir)):
```

```
    os.mkdir(dest_dir)
```

```
# Allocate sub-function splitting and saving tiles
```

```
def split_n_save(fname, dest_dir, stride=50, tile_shape=(125,125)):
```

```
    img_arr = np.asarray(PIL.Image.open(fname))
```

```
    # Split images
```

```
    tiles, xys = img_splitter(img_arr, stride=stride, tile_shape=tile_shape)
```

```
    # save each tile
```

```
    for j in range(len(tiles)):
```

```
        # set name,
```

```
        # img name
```

```
# num
```

```
        sname = os.path.split(set_dir)[1] + '__' + os.path.split(fname)[1].split('.')[0] +
```

```
        '__' + str(j) + '.jpg'
```

```
        sname = os.path.join(dest_dir, sname)
```

```
        tile = PIL.Image.fromarray(tiles[j])
```

```
        # Randomly adjust brightness
```

```
        factor = np.round((np.random.random() + 0.5), 2)
```

```

        enhan = ImageEnhance.Brightness(tile)
        ntile = enhan.enhance(factor)
        ntile.save(sname)

    return len(tiles)

''' Create tiles from /false_imgs subdir '''
'''-----'''

false_dir = os.path.join(set_dir, 'false_images')
print('Creating False tiles, from subdir')
if(os.path.exists(false_dir)):
    fnames = list_files(false_dir)
        # Trim number of images to use for tile-making
    if(len(fnames) > img_lim):
        random.shuffle(fnames)
        fnames = fnames[:img_lim]
        # Make Tiles
    for i in range(len(fnames)):
        split_n_save(fnames[i], dest_dir, stride=stride, tile_shape=tile_shape)

''' Create tiles from /set_dir/false_regions.txt '''
'''----- '''

false_regions = os.path.join(set_dir, 'false_regions.txt')
print('Creating False tiles, from user-labelled regions')
if(os.path.exists(false_regions)):
    df = pd.read_csv(false_regions, sep='\t')

        # Go through each line in false_regions.txt callout
    for i in range(len(df)):
        line = df.iloc[i]
            # Filename construction/declaration
        try:
            fname = os.path.join(set_dir, (line['Label'].split(':')[1] + '.jpg'))
            IMG = PIL.Image.open(fname)

```

```

    except:
        fname = os.path.join(set_dir, (line['Label'].split(':')[1] + '.JPG'))
        IMG = PIL.Image.open(fname)

        # we don't use split_n_save() here because we only want part of the
img not the whole thing
        img = np.asarray(IMG)
        tiles, xys = img_splitter(img[line['BY']:(line['BY'] + line['Height']),
line['BX']:(line['BX'] + line['Width']), stride=stride, tile_shape=tile_shape)

        # Save each tile
        for j in range(len(tiles)):

            # img name          # num          # set          name,
            sname = os.path.split(set_dir)[1] + '__' +
os.path.split(fname)[1].split('.')[0] + '__' + str(j) + '.jpg'
            sname = os.path.join(dest_dir, sname)
            tile = PIL.Image.fromarray(tiles[j])
            # randomly adjust brightness
            factor = np.round((np.random.random() + 0.5), 2)
            enhan = ImageEnhance.Brightness(tile)
            ntile = enhan.enhance(factor)
            # Save
            ntile.save(sname)

        return

#####
#####
#####
#####
#####
#####

def crop_met_imgs(folder, train_dir, eval_dir, stride=10, tile_l=125, eval_percent=0.2, silent=False):
    """

```


This function makes meteorite tiles from their original images, using callouts from 'meteorite_locations.txt' within 'folder'.

The text file should be in ImageJ output format.

Inputs:

folder	[str]	fullpath to the directory containing the images and the '.txt' file containing meteorite locations in each image
train_dir	[str]	fullpath to dir where training tiles are saved
eval_dir	[str]	fullpath to dir where evaluation tiles are saved
stride	[int]	number of pixels to stride over the feature in each step
tile_l	[int]	height-width of the tiles
eval_percent	[float]	percentage of lines in text file to set aside for validation
silent	[bool]	False means print everything, pretty sure this feat. is outdated

Outputs:

[void]

'''

```
# Create output dirs
if(not os.path.exists(train_dir)):
    os.mkdir(train_dir)
if(not os.path.exists(eval_dir)):
    os.mkdir(eval_dir)

# Get image names and txt filenames
contents = list_files(folder)
txt_fname = list_files(folder, 'met')[0]

df = pd.read_csv(txt_fname, sep='\t')

# Prep log file
sname = os.path.join(eval_dir, 'original_image_details.txt')
f = open(sname, 'w')
f.write('\N\tprob\tfilename\tx1\tx2\ty1\ty2\n')

# Loop through each position listed in txt file
```

```

for i in range(len(df)):
    string = '\r\tCurrent meteorite location: ' + str(i+1) + '/' + str(len(df) + 1)

    if(silent != True):
        sys.stdout.write(string)
        sys.stdout.flush()

    line = df.iloc[i]

    # Filename construction/declaration
    if(': ' in line['Label']):
        img_fname = line['Label'].split(':')[1]
        img_fname = list_files(folder, img_fname)[0]
    else:
        img_fname = line['Label']
        img_fname = os.path.join(folder, img_fname)

    IMG = PIL.Image.open(img_fname)

    img = np.asarray(IMG)

    cx = float(line['BX'])
    cy = float(line['BY'])
    w = float(line['Width'])
    h = float(line['Height'])

    # Some mathy declaration stuff that I can't remember why it works but it does
    mx1 = cx
    mx2 = cx + w
    my1 = cy
    my2 = cy + h

    x_num = int( (tile_1 - w) // stride)
    y_num = int( (tile_1 - h) // stride)

    x1 = int(mx2 - tile_1)
    y1 = int(my2 - tile_1)
    x2 = int(x1 + tile_1)
    y2 = int(y1 + tile_1)

```

```

        # Making training tiles
    if(i <= len(df) * (1 - eval_percent)):
        save_dir = train_dir
    if(i > len(df) * (1 - eval_percent)):
        save_dir = eval_dir
    string = str(i) + '\t' + '1.0' + '\t' + img_fname + '\t' + str(x1) + '\t' + str(x2) + '\t' +
str(y1) + '\t' + str(y2) + '\n'
    f.write(string)

    # Iterate through the strides
    for j in range(x_num):
        while( x1 < 0 ):
            x1 += stride
            x2 += stride
            j += 1
        if(x2 > img.shape[1]):
            break

    for k in range(y_num):
        while( y1 < 0 ):
            y1 += stride
            y2 += stride
            k += 1
        if(y2 > img.shape[0]):
            break

    tile = np.copy(img[y1:y2, x1:x2])

    # Rotate each possible stride
    for l in range(4):
        # Save
        TILE = PIL.Image.fromarray(tile).rotate((l*90))
        sname = os.path.join(save_dir,
(str(i)+'_'+str(j)+'_'+str(k)+'_'+str(l)+'-' +
'__' + os.path.split(folder)[1] +
'__' +
os.path.split(img_fname)[1]))

        # Random Brightness adjustment
        factor = np.round((np.random.random() + 0.5), 2)
        enhan = ImageEnhance.Brightness(TILE)
        ntile = enhan.enhance(factor)
        #SAVE
        ntile.save(sname)

```

```
y1 += stride
y2 += stride
```

```
x1 += stride
x2 += stride
y1 = int(my2 - tile_l)
y2 = int(y1 + tile_l)
```

```
f.close()
string = '\r\tCurrent meteorite location: ' + str(i+1) + '/' + str(len(df) + 1)
```

```
if(silent != True):
    sys.stdout.write(string)
    sys.stdout.flush()
    print('\n\n')
```

```
return
```

```
#####
#####
#####
#####
#####
#####
```

```
def make_training_set(imgs_dir, set_dir, tstride=12, fstride=70, tile_l=125, eval_percent=0.2,
keep_extra=False):
```

```
    ''' This func takes raw images from training data collection, and makes True and False,
    Training and
    Validation training sets (tiles).
```

```
    Inputs:
```

```
        imgs_dir    [str]    fullpath to directory of images, which contains
    meteorite images, annotation file, and subdir with false images
        set_dir     [str]    fullpath to saving directory, sub dirs for train and
    validation will be created within
        tstride     [int]    stride length for true tiles
        fstride     [int]    stride length for false tiles
```

`tile_l` [int] length of one side of the tiles (tiles will be square)

Outputs:

`void`

4 June 2021

'''

`# Delcare directories`

`ftemp_dir = os.path.join(set_dir, 'extra_false')`

`train_dir = os.path.join(set_dir, 'train_dir')`

`valid_dir = os.path.join(set_dir, 'valid_dir')`

`train_f_dir = os.path.join(train_dir, 'False')`

`train_t_dir = os.path.join(train_dir, 'True')`

`valid_f_dir = os.path.join(valid_dir, 'False')`

`valid_t_dir = os.path.join(valid_dir, 'True')`

`if(not os.path.exists(train_dir)):`

`os.mkdir(train_dir)`

`if(not os.path.exists(valid_dir)):`

`os.mkdir(valid_dir)`

`if(not os.path.exists(train_f_dir)):`

`os.mkdir(train_f_dir)`

`if(not os.path.exists(train_t_dir)):`

`os.mkdir(train_t_dir)`

`if(not os.path.exists(valid_f_dir)):`

`os.mkdir(valid_f_dir)`

`if(not os.path.exists(valid_t_dir)):`

`os.mkdir(valid_t_dir)`

'''

`# Make true files`

`p = mp.Process(target = crop_met_imgs,`

`args = [imgs_dir, train_t_dir, valid_t_dir, tstride, tile_l,`

`eval_percent, False])`

`p.start()`

'''

`crop_met_imgs(imgs_dir,`

`# imgs_dir`

```

        train_t_dir,          # train_dir
        valid_t_dir,        # eval_dir
        stride      = tstride,
        tile_l      = tile_l,
        eval_percent = eval_percent,
        silent      = False)

    # Make False tiles
    make_false_set(imgs_dir,          # set_dir
                  ftemp_dir,        # dest_dir
                  stride            = fstride,
                  tile_shape       = (tile_l,tile_l),
                  img_lim          = 100)

    # Get count of True tiles
    n_train_t = len(list_files(train_t_dir, '.jpg'))
    n_valid_t = len(list_files(valid_t_dir, '.jpg'))

    # Copy False tiles
    # Valid
    copy_files(ftemp_dir, valid_f_dir, n_valid_t, move=True)
    # Train
    copy_files(ftemp_dir, train_f_dir, n_train_t)

    # Remove temp tiles
    if(keep_extra == False):
        shutil.rmtree(ftemp_dir)

    # Write to log file
    sname = os.path.join(set_dir, 'training_set_log')
    strr = 'N train tiles: ' + str(2*n_train_t) + '\n' + 'N valid tiles: ' + str(2*n_valid_t)
    f = open(sname, 'w')
    f.write(strr)
    f.close()

    return

```

```
#####
#####
#####
#####
#####
#####
```

```
def highlight_annotes(folder, dest_dir, h_w=4):
```

```
    '''This function highlights, in full images, the meteorites a user has marked to use as training tiles. This is good for
```

```
        making examples for the human test dir, used by the gui. *Note* the bounding boxes are intentionally off-center of the meteorite.
```

```
        The bounding boxes ARE NOT representative of the prior human annotation, they are all 125x125
```

```
    Inputs:
```

```
        folder      [str]  full path to the folder containing full images, with mets
in the frame, dir should also contain
                                'meteorite_locations.txt' to indicate where the
mets are located.
```

```
        dest_dir    [str]  fullpath to destination dir where images will be saved
```

```
        h_w         [int]  highlight width for the bounding box
```

```
    '''
```

```
        # Make destination directory
```

```
    if(not os.path.exists(dest_dir)):
```

```
        os.mkdir(dest_dir)
```

```
        # Intake meteorite locations
```

```
    met_loc_fname = os.path.join(folder, 'meteorite_locations.txt')
```

```
    f = open(met_loc_fname, 'r')
```

```
    lines = f.readlines()[1:] # ignore header line
```

```
    f.close()
```

```

        # Make log file
log_fname = os.path.join(dest_dir, 'candidate_locations.txt')
if(os.path.exists(log_fname)):
    log = open(log_fname, 'a')
else:
    log = open(log_fname, 'w')
    log.write(('Y1\tY2\tX1\tX2\tImage'))

    # Go through each meteorite instance
for i in range(len(lines)):
    content = lines[i].split('\t')
        # Open image
img_fname = list_files(folder, content[1].replace('Stack:', ''))[0]
overlay_sname = os.path.join(dest_dir, (os.path.split(folder)[1] + '+candidate_' +
os.path.split(img_fname)[1]))
if(os.path.exists(overlay_sname)):
    img_fname = overlay_sname

img = PIL.Image.open(img_fname)

        # Figure out where to draw bounding boxes
        # Add random variance, so they are not all centered exactly
yvar = 63 - float(content[5]) / 2.
xvar = 63 - float(content[4]) / 2.

cy = int(float(content[3]) + float(content[5]) / 2. + np.random.randint((-1*yvar),
yvar))
cx = int(float(content[2]) + float(content[4]) / 2. + np.random.randint((-1*xvar),
xvar))

y1 = cy - 63
y2 = y1 + 125
x1 = cx - 63
x2 = x1 + 125

xys = np.array([y1,y2,x1,x2])

        # write the suspect's tile locations in log
string = str('\n' + str(xys[0]) +
            '\t' + str(xys[1])      +
            '\t' + str(xys[2]) +
            '\t' + str(xys[3]) +

```



```

\t' + os.path.split(overlay_sname)[1] )

log.write(string)

# Annotate the overlay image at the tile's location
draw = PIL.ImageDraw.Draw(img)
draw.line([xys[2], xys[0], xys[2], xys[1]], fill='yellow', width=h_w) # Left
draw.line([xys[3], xys[0], xys[3], xys[1]], fill='yellow', width=h_w) # Right

draw.line([xys[2], xys[1], xys[3], xys[1]], fill='yellow', width=h_w) # Bottom
draw.line([xys[2], xys[0], xys[3], xys[0]], fill='yellow', width=h_w) # Top
draw.text([xys[2], xys[0]], str(i) )

# Save the overlain image
img.save(overlay_sname)
log.close()

return

```

```

#####
#####
#####
#####
#####
#####
#####

```

```

def
summarize_train_sets(imgs_dir='/data0/Seamus_data/MET_searching/keras_train/full_images_all'
, st=12, side=125):
'''

```

This function summarizes all the training data we have on hand in graph format. The training sets should be subdirs located within 'imgs_dir'

Inputs:

imgs_dir	[str]	fullpath to dir containing full images with meteorites and text files with location callouts
st	[int]	stride? idk
side	[int]	length of one tile on one side

Outputs:

```

void

'''

dirs = list_files(imgs_dir, ', '.')

dfs = []
diams = []
labels = []

    # Go through each 'training_set' subdir
for i in range(len(dirs)):
    print(osplit(dirs[i])[1])
    met_log = ojoin(dirs[i], 'meteorite_locations.txt')
    df = pd.read_csv(met_log, sep='\t')
    df['set'] = [osplit(dirs[i])[1] for j in range(len(df))]
    df['n_tiles'] = [(int((side - df['Width'].iloc[j])/st) * int((side - df['Height'].iloc[j])/st)) *
4 for j in range(len(df))]
    df['diam'] = [np.mean((df['Width'].iloc[j], df['Height'].iloc[j])) for j in
range(len(df))]

    diam = np.array(df['diam'])
    label = '(' + str(len(df)) + ')' + osplit(dirs[i])[1]

    diams.append(diam)
    labels.append(label)
    dfs.append(df)

full_df = pd.concat(dfs)
tot_n_tiles = np.sum(full_df['n_tiles'])

print('Number of annotations: ', len(full_df))
print('Total number of tiles: ', tot_n_tiles )

    # Make log file
full_df.to_csv(ojoin(imgs_dir, 'training_sets_log.csv'), index=False)

    # Make plot
plt.title('Annotations Histogram')
plt.xlabel('Pixel Diameter')

```


Fullpath to the validation directory, if you wan it different from the training data

```
save_as [str] = False
```

Optional save name for the model (not fullpath), otherwise, save name will be generated based on time of creation

```
saved_model [str] = False
```

Optional filepath to a saved model to continue training

Outputs:

```
model_sname [str] full path to .h5 model file
```

model-YYYY-MM-DD.h5 [file] weighted model trained on the given data found in train_dir

```
model-YYYY-MM-DD.png [file] training history-plot of the model
```

```
'''
```

```
''' Preparing Names, directories, and files'''
```

```
'''-----'''
```

```
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '1'
```

```
# Setting save name for the sub_dir, and by extention model and plot name
```

```
# The name is generated using the time it was made
```

```
time = str(datetime.datetime.now())
```

```
time = time.replace(' ', '_')
```

```
time = time.replace(':', '-')
```

```
time = time.replace('.', '-')
```

```
temp_name = ('model_' + time)
```

```
# Save model as given name (optional)
```

```
if(save_as!= False):
```

```
temp_name = save_as
```

```
models_dir = os.path.join(os.path.split(train_dir)[0], ('models-' + os.path.split(train_dir)[1]))
```

```
save_dir = os.path.join(models_dir, temp_name)
```

```
# Check for and make generic models dir and specific dir for this model
```

```
if(os.path.exists(models_dir) == False):
```

```

        os.mkdir(models_dir)
if(os.path.exists(save_dir) == False):
    os.mkdir(save_dir)

    # Preparing model log and plot names
model_sname = temp_name + '.h5'
model_sname = os.path.join(save_dir, model_sname)

plot_sname = temp_name + '.png'
plot_sname = os.path.join(save_dir, plot_sname)

txt_sname = temp_name + '_log.txt'
txt_sname = os.path.join(save_dir, txt_sname)

'''Constructing model architecture'''
'''-----'''

    # Constructing a model
model = Sequential()

# good model starts with 24 filters, then doubles at every layer

    # Liam's Seamus-modified arc.
model.add(Conv2D(30, 3, strides=1,
                 input_shape=(tile_size[0], tile_size[1], 3) ))
model.add(Activation('relu'))
model.add(
normalization.BatchNormalization() )
model.add(MaxPooling2D((2,2), 2) )

model.add(Conv2D(60, 3, strides=1) )
model.add(Activation('relu'))
model.add(
normalization.BatchNormalization() )
model.add(MaxPooling2D((2,2), 2))

model.add(Conv2D(120, 3, strides=1))
model.add(Activation('relu'))
model.add(
normalization.BatchNormalization() )

```

```

model.add(MaxPooling2D((2, 2), 2))

model.add(Conv2D(240, 3, strides=1))
model.add(Activation('relu'))
model.add(
normalization.BatchNormalization() )
model.add(MaxPooling2D((2, 2), 2))

model.add(Flatten() )
model.add(Dense(1000) )
model.add(Activation('relu') )

model.add(Dropout(0.5) )

model.add(Dense(150))
model.add(Activation('relu'))

model.add(Dropout(0.5) )

model.add(Dense(1) )
model.add(Activation('sigmoid') )

model.summary()

# Compile the model
model.compile(loss = 'binary_crossentropy',
optimizer = 'rmsprop',
metrics = ['accuracy'] )

"""Declaring Hyperparameters, and defining train/val data from directories"""

# Load optional saved model and its log file
if(saved_model != False):
model = keras.models.load_model(saved_model)
old_log = "
old_log_name = saved_model[:-3] + '.txt'
if(os.path.exists(old_log_name)):

```

```

f = open(old_log_name, 'r')
old_log = f.read()
f.close()

# Preparing training and validation data
train_datagen = ImageDataGenerator(rescale = 1./255. ,
                                   validation_split = 0 )
valid_datagen = ImageDataGenerator(rescale = 1./255. ,
                                   validation_split = .99 )

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size = tile_size,
    batch_size = n_batch_size,
    class_mode = 'binary',
    subset = 'training')

# Preparing validation data
valid_generator = None
valid_steps = None # These are set 0 incase you don't want to validate (not
recommended)

# Validate on same population of images as training
if(valid_dir != False):

    valid_steps = n_valid_steps
    val_batch_size = n_batch_size

    if(auto_calc_steps == True):
        true_dir = os.path.join(valid_dir, 'True')
        false_dir = os.path.join(valid_dir, 'False')
        n_total_tiles = len(list_files(false_dir)) + len(list_files(true_dir))
        n_valid_steps = int(n_total_tiles / n_batch_size)

    valid_generator = valid_datagen.flow_from_directory(
        valid_dir,
        target_size = tile_size,
        batch_size = val_batch_size,

```

```

class_mode = 'binary',
subset      = 'validation',
shuffle=True)

# Setup checkpoint
# *NOTE* this only works when validation is present
backup_name = os.path.join(os.path.split(model_sname)[0], ('backup_' +
os.path.split(model_sname)[1]))
checkpoint = ModelCheckpoint(backup_name,
                             monitor = 'val_acc',
                             mode    = 'max',
                             save_best_only = True)
#es = EarlyStopping(monitor='val_acc', mode='max', patience=50)

# Auto calculate steps
if(auto_calc_steps == True):
    true_dir = os.path.join(train_dir, 'True')
    false_dir = os.path.join(train_dir, 'False')
    n_total_tiles = len(list_files(false_dir)) + len(list_files(true_dir))
    n_epoch_steps = int((n_total_tiles / n_batch_size) / 3 ) #
Manually set to reduce steps by factor of 5

#####

''' Train the model '''
'''-----'''

hist = model.fit_generator(
    train_generator,
    steps_per_epoch = n_epoch_steps,
    epochs          = n_epochs,
    validation_data = valid_generator,
    validation_steps = n_valid_steps,
    callbacks       = [checkpoint],
    shuffle        = True
)

# Getting history and summary of model training
model.summary()

```



```
history = hist.history
```

```
'''Saving model and accessory files '''  
'''-----'''
```

```
model.save(model_sname)
```

```
    # Writing text log file  
with open(txt_sname, 'a+') as file:
```

```
    file.write(str(datetime.datetime.now()))  
        # Giving credit to optional saved model  
    if(saved_model != False):  
        string = ('\n\nThis model was trained using pre-existing model: '  
                + saved_model + '\n')  
        file.write(string)  
        string = ('\n\nPrevious Log: ' + '\n\n' + old_log + '\n\n')  
        file.write(string)
```

```
        # Writing training history and summary to text file  
    string = ('\n\nTrained from directory: ' + train_dir +  
            '\nTraining Params: ' +  
            '\nBatch Size = ' + str(n_batch_size) +  
            '\nEpochs = ' + str(n_epochs) +  
            '\nEpoch Steps = ' + str(n_epoch_steps) +  
            '\nTile Size = ' + str(tile_size) +  
            '\nHistory: ' +  
            '\n\tAccuracy:\n ' + str(history['acc'][-1]) )
```

```
    string = string + '\n\n'  
    file.write(string)  
    file.write('Summary:\n')  
    model.summary(print_fn=lambda x: file.write(x + '\n'))  
    file.write('\n\n\n')  
file.close()
```

```
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
```

```
plt.title('Training History')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim((0,1))
plt.plot(np.arange(n_epochs), history['acc'], 'r-', label='Training')
plt.plot(np.arange(n_epochs), history['val_acc'], 'b--', label='Validation')
plt.legend(loc='lower right')
plt.savefig(plot_sname)
plt.clf()
```

```
return model_sname
```

```
#####
#####
#####
#####
#####
#####
```

```
def revolver_train(train_data_dir, model_sname, saved_model=False, n_epochs=8):
```

```
    ''' This function revolves the false training data every 'n_epochs' to give the model a chance
to train on everything
```

```
        in the instances (most) where there's waaaayy to much false data to keep pace with
true tiles
```

```
- 'train_data_dir' should be 'train_sets' in accordance with training_pipeline()
```

```
-This assumes True tiles already in train_dir
```

```
'''
```

```

print('Started Revolver training')
    # Pointing out directory paths
false_pool = os.path.join(train_data_dir, 'false_pool')
valid_dir = os.path.join(train_data_dir, 'valid_dir')
train_dir = os.path.join(train_data_dir, 'train_dir')
if(True):
    true_dir = os.path.join(train_dir, 'True')
    false_dir = os.path.join(train_dir, 'False')
        # Clear false_dir if it exists, then make a fresh one
    if(os.path.exists(false_dir)):
        shutil.rmtree(false_dir)
        os.mkdir(false_dir)

        # Count number of true and false tiles
n_true_tiles = len(LIB.list_files(true_dir))
n_false_tiles = len(LIB.list_files(false_pool))

n_revolutions = int(n_false_tiles / n_true_tiles) * 2

print('Number of True tiles: ', n_true_tiles)
print('Number of False tiles: ', n_false_tiles)
print('Number of Revolutions: ', n_revolutions)
time.sleep(1)

    # Start training revolutions (move false tiles in; train; move false tiles out)
for i in range(n_revolutions):
    # Copy False tiles over to train_dir
    src = false_pool
    dest = false_dir
    num = n_true_tiles

    LIB.copy_files(src, dest, num)

        # train
    saved_model = LIB.train_meteorite_network(
        train_dir,
        tile_size
        = (125, 125),
        n_epochs
    = n_epochs,

```

```

        = valid_dir,
        = model_sname,
saved_model,
        = True    )

        # Remove False dir and therefore tiles from train_dir (make it ready for the
next one)
        shutil.rmtree(false_dir)
        # Make new empty false_dir
        os.mkdir(false_dir)

    return

#####
#####
#####
#####
#####
#####

# Experimental version, may contain bugs

def predict_on_dir_2(model_path, folder, input_shape=(125,125), pred_thresh=0.9, stride=70, h_w=
4, gpu_batch_size=190, silent=False):
    ''' This function takes a trained model and a folder of images, to make predictions
        overlain on the original images.

    Inputs:
        model_path    [str] Fullpath to the trained model
        folder        [str] Fullpath to the folder of images to predict on
        input_shape   [tuple] Shape the model requires as an input (2d)
        pred_thres    [float] Threshold for highlighting a patch

```

stride [int] Number of pixels to stride over when predicting
on an image

h_w [int] Highlighter width
n_cores [int] Number of cores to use in process
gpu_batch_size [int] Batch size for the gpu
silent [bool] If != True, print info during runtime

Outputs:

total_predictions [int]
Total number of predictions across all the images
predictions.txt [file]
Lists information about each image predicted on, along with function parameters used.

overlayed image(s) [file(s)]
The original images are overlaid with white boxes around patches with a high probability of containing a meteorite (specified by

'pred_thres').

These overlays are stored in sub-folder: 'folder\\predictions'.

Dependencies:

list_files(): lists images to be predicted on in 'folder'

```
'''
```

```
times = []  
t0 = dtm.now()
```

```
''' Check for previous prediction attempt, or start a new interim log '''  
'''-----'''
```

```
imgs_dir = ojoin(folder, 'images')  
interim_fname = ojoin(folder, 'interim_log.txt')
```

```
    # Previous attempt detected #  
    #-----#
```

```
if(os.path.exists(interim_fname) == True):  
    resumed = True  
    print('Previous prediction attempt detected')
```

```

        # Reference directories
f = open(interim_fname, 'r')
save_dir = f.read().replace('\n', '')
f.close()
hist_dir = ojoin(save_dir, 'histograms')
overlay_dir = ojoin(save_dir, 'suspect_images')
predcsv_dir = ojoin(save_dir, 'prediction_csvs')
all_results_fname = ojoin(save_dir, (osplit(folder)[1] + '_all_results.txt'))
        # get fnames to predict on
finished_images = list_files(predcsv_dir, '.csv')
all_images = list_files(imgs_dir, '.jpg')
fnames = [] # Fnames to predict on

        # Get list of completed images, from pred csv dir
for i in range(len(finished_images)):
        finished_images[i] = osplit(finished_images[i])[1]

for i in range(len(all_images)):
        image_name = osplit(all_images[i])[1].replace('.JPG', '.csv')
        if(image_name not in finished_images):
                fnames.append(all_images[i])

# First attempt (no existing interim or final log) #
#-----#

else:
    resumed = False
    if(silent != True):
        print('This is the first logged prediction attempt')
        # Prep predictions (output) folders
    time_str = str(dtm.now()).replace(' ', '_').replace(':', '-').split('.')[0]
    pred_name = 'predictions_' + time_str
    pred_name = osplit(model_path)[1].replace('.h5', '_') + pred_name
    save_dir = ojoin(folder, pred_name)
    hist_dir = ojoin(save_dir, 'histograms')
    overlay_dir = ojoin(save_dir, 'suspect_images')
    predcsv_dir = ojoin(save_dir, 'prediction_csvs')
    all_results_fname = ojoin(save_dir, (osplit(folder)[1] + '_all_results.txt'))
    os.mkdir(save_dir)
    os.mkdir(hist_dir)
    os.mkdir(overlay_dir)

```

```

os.mkdir(predcsv_dir)
    # get image fnames to predict on
fnames = list_files(imgs_dir, '.jpg')
    # start new interim log
interim_log = open(interim_fname, 'w')
interim_log.write(save_dir)
interim_log.close()

# Load model #
#-----#
model = keras.models.load_model(model_path)
if(silent != True):
    print('Loading Model:\n\t' + model_path)
    print('Predicting on folder:\n\t' + folder)

# Define Multi-process function for saving predictions #
#-----#
def save_predictions(preds, xys, fname, predcsv_dir, hist_dir):
    short_fname = os.path.split(fname)[1]
        # Convert preds and xys to arrays
    xys = np.asarray(xys)
    pred_arr = np.zeros(len(preds))
    for j in range(len(preds)):
        pred_arr[j] = float(preds[j][0])
        # Construct dataframe
    df = pd.DataFrame({ 'pred_val':    pred_arr,
                       'Y1':         xys[:,0],
                       'Y2':         xys[:,1],
                       'X1':         xys[:,2],
                       'X2':         xys[:,3],
                       'fname':      [short_fname for i in
range(len(preds))]    })
        # Save csv
    sname = os.path.join(predcsv_dir, short_fname).replace('.JPG','.csv')
    df.to_csv(sname, index=False)

    # Save histogram
    histname = ojoin(hist_dir, short_fname.replace('.JPG','.png'))

```

```

plt.title(osplit(fname)[1])
plt.ylabel('Number')
plt.xlabel('Prediction value')
plt.xlim((0, 1.3))
plt.hist(pred_arr, bins=20, log=True)
plt.savefig(histsname)
plt.clf()

return

```

```

''' Predict on each image '''
'''-----'''

```

```

jobs = []
    # Loop through each image
for i in range(len(fnames)):
    print(osplit(fnames[i])[1])
        # Open image
    img = PIL.Image.open(fnames[i])
    img_array = np.asarray(img)
    img.close()
    del img
        # Split image
    tiles, xys = img_splitter(img_array, tile_shape=input_shape, stride=stride)
        # Predict on image
    preds = model.predict(np.asarray(tiles)/255., batch_size=gpu_batch_size)
        # Save predictions to csv
    p = mp.Process(        target = save_predictions,
                        args = [preds, xys, fnames[i], predcsv_dir, hist_dir] )
    jobs.append(p)
    p.start()

```

```

if(len(fnames) > 0):
    p.join()

```

```

predtime = dtm.now() - t0

```



```

''' Round up all the results '''
'''-----'''

t1 = dtm.now()

    # Compile the csv's into one file
csv_fnames = list_files(predcsv_dir, '.csv')
csv_dfs = [pd.read_csv(csv_fnames[i]) for i in range(len(csv_fnames))]

final_df_sname = os.path.join(save_dir, 'all_predictions.csv')
final_df = pd.concat(csv_dfs)
final_df.to_csv(final_df_sname)

    # Make histogram of all the prediction values from the entire directory
hist_sname = os.path.join(save_dir, 'all_predictions_hist.png')
all_preds = final_df['pred_val']
string = ''
for i in np.arange(9, 0, -1):
    n_fp = len(np.where(all_preds > (i*0.1))[0])
    string += '>0.' + str(i) + ': ' + str(n_fp) + '\n'

string += '>0.01: ' + str(len(np.where(all_preds > 0.01)[0])) + '\n'

plt.title(('All results: ' + osplit(folder)[1]))
plt.ylabel('Number')
plt.xlabel('Prediction value')
plt.xlim((0, 1.3))
plt.text(1.025, 1000, string)
plt.hist(all_preds, bins=20, log=True)
plt.savefig(hist_sname)
plt.clf()

    # Write final log file
final_pred_log_fname = ojoin(folder, ('final_prediction_log_' + osplit(save_dir)[1] + '.txt'))
if(not os.path.exists(final_pred_log_fname)):
    n_imgs = len(list_files(imgs_dir))
    dt = t1-t0
    avg_t = dt.total_seconds() / len(fnames)
    notes = ''
    n_suspects = len(np.where(all_preds > (0.9))[0])

```

```

n_sus_per_img = n_suspects / float(n_imgs)
if(resumed == True):
    notes = 'This prediction was stopped, then resumed. Times may not be
accurate.'

```

```

final_log_string = ('Final prediction log' + '\n' +
                    'Start Time: ' + str(t0) + '\n' +
                    'End Time: ' + str(t1) + '\n' +
                    'Delta Time: ' + str(dt) + '\n' +
                    'Directory: ' + folder + '\n' +
                    'Model: ' + model_path + '\n' +
                    'N Suspects over 0.9: ' + str(n_suspects) +
'\n' +
                    'Suspects per image: ' + str(n_sus_per_img)
+ '\n' +
                    'Total Number of images: ' + str(n_imgs) +
'\n' +
                    'Avg time per img [sec]: ' + str(avg_t) +
'\n' +
                    'Notes:\n\n\t' + notes
)
f = open(final_pred_log_fname, 'w')
f.write(final_log_string)
f.close()

```

```

os.remove(interim_fname)

```

```

# If pred threshold is given, make the overlays and select tiles of interest
if(pred_thresh != None):
    pull_preds(save_dir, pred_thresh, h_w=h_w)

```

```

return save_dir

```

```

#####
#####
#####

```

```

def save_overlay(args):
    # Open csv
    csv_fname = args[0]
    imgs_dir = args[1]
    thresh = args[2]
    overlay_dir = args[3]
    problem_dir = args[4]
    problem_thresh = args[5]

    h_w = 4

    df = pd.read_csv(csv_fname)
    suspects = df[df['pred_val'] > thresh]

    # If nothing is over-threshold, skip this image/csv
    if(len(suspects) == 0):
        return

    this_fname = df['fname'].iloc[0]
    img_fname = ojoin(imgs_dir, this_fname)

    # If there are too many over-threshold things, save it in a separate folder
    if(len(suspects) > problem_thresh):
        new_name = os.path.join(problem_dir, this_fname)
        shutil.copyfile(img_fname, new_name)

        return

    overlay_sname = ojoin(overlay_dir, ('predicted_' + osplit(img_fname)[1]))

    # Open Image
    img = PIL.Image.open(img_fname)
    img_array = np.asarray(img)
    mask = np.zeros(img_array.shape, bool)

    # Save csv of overthreshold predictions, in pred dir
    thresh_sname = ojoin(osplit(overlay_dir)[0], osplit(overlay_sname)[1].replace('.JPG',
'.csv'))
    suspects['fname'] = [osplit(overlay_sname)[1] for i in range(len(suspects))]

```

```

suspects.to_csv(thresh_sname, index=False)

    # Sort the suspects
sorted_df = suspects.sort_values(['pred_val'], ascending=False)

tpreds = np.asarray(sorted_df['pred_val'])
txys = np.asarray(sorted_df[['Y1','Y2','X1','X2']])

draw = PIL.ImageDraw.Draw(img)

    # Loop through each suspect in the image and overlay with a bounding box
for j in range(len(tpreds)):
    xys = txys[j]

        # Check if this tile is overlapping with a previously saved one
if(np.sum(mask[xys[0]:xys[1], xys[2]:xys[3]]) == 0):
    # Mask this tile, (don't save again)
    mask[xys[0]:xys[1], xys[2]:xys[3]] = 1

        # Annotate the overlay image at the tile's location
draw.line([xys[2], xys[0], xys[2], xys[1]], fill='yellow', width=h_w) # Left
draw.line([xys[3], xys[0], xys[3], xys[1]], fill='yellow', width=h_w) # Right

draw.line([xys[2], xys[1], xys[3], xys[1]], fill='yellow', width=h_w) # Bottom
draw.line([xys[2], xys[0], xys[3], xys[0]], fill='yellow', width=h_w) # Top
#draw.text([xys[2], xys[0]], str(tpreds[j]) )

img.save(overlay_sname)

##### THis may cause problems #####3
#####
img.close()
del img

return

# 208 /257

def pull_preds(pred_dir, thresh, problem_thresh=50, h_w=4):

```

'''This function gathers the prediction values and locations for all the images that have already been inferred on by the model.

Any tile that is over threshold is compiled into 'pred_dir/predictions.txt' to use later for sorting. This func also overlays the images with bounding boxes.

Inputs:

pred_dir [str] the prediction directory within a flight or image dir, if image/flight dir is given, it defaults to the first pred_dir it can find
thresh [float] user-defined threshold between 0 and 1, typically 0.9
h_w [int] highlighter width for overlaying bounding boxes

Ouputs:

pred_dir/predictions.txt [file] csv with over-thresh pred values, locations and image names.

'''

''' Figure out where we are '''

'''-----'''

```
imgs_dir = os.path.join(os.path.split(pred_dir)[0], 'images')
overlay_dir = os.path.join(pred_dir, 'suspect_images')
problem_dir = os.path.join(pred_dir, 'problem_images')
csv_dir = os.path.join(pred_dir, 'prediction_csvs')
```

```
if(not os.path.exists(overlay_dir)):
    os.mkdir(overlay_dir)
if(not os.path.exists(problem_dir)):
    os.mkdir(problem_dir)
```

''' MP function for saving overlay '''

'''-----'''

''' Go through each image '''

'''-----'''

```
csv_fnames = list_files(csv_dir, '.csv')
```

```
args = [[csv_fnames[i], imgs_dir, thresh, overlay_dir, problem_dir, problem_thresh] for i in
range(len(csv_fnames))]
```

```
with Pool(processes=8) as pool:
    pool.map(save_overlay, args)
```

```
# Combine overthresh csv into one 'predictions.txt' for the gui to read
csv_fnames = list_files(pred_dir, '.csv', 'all_predictions.csv') # chill out, its ignoring
all_predictions.csv
```

```
csv_dfs = [pd.read_csv(csv_fnames[i]) for i in range(len(csv_fnames))]
```

```
sname = str(thresh) + '+_detections.txt'
final_df_sname = os.path.join(pred_dir, sname)
final_df = pd.concat(csv_dfs)
final_df = final_df[['Y1','Y2','X1','X2','fname']]
```

```
final_df.to_csv(final_df_sname, sep='\t', index=False)
```

```
# Get rid of redundant csvs
for i in range(len(csv_fnames)):
    os.remove(csv_fnames[i])
```

```
return
```

```
#####
#####
#####
```

```
# Current Working Version
```

```
#####  
#####  
#####  
#####
```

```
def all_preds_to_files(pred_dir):
```

```
'''
```

This function makes tiles from all over-threshold predictions, this is good for an initial retrain, where you've just predicted on a set of images you know to be false.

Inputs:

pred_dir [str] fullpath to predictions directory within img-containing dir

Outputs:

tile(s) [files] the tiles that the model thought were meteorites

```
'''
```

```
    # Directory declaration
```

```
    log_fname = os.path.join(pred_dir, 'detections.txt')
```

```
    sus_dir = os.path.join(pred_dir, 'suspect_images')
```

```
    tile_dir = os.path.join(pred_dir, 'all_tiles')
```

```
    img_dir = os.path.split(pred_dir)[0]
```

```
    img_dir = os.path.join(img_dir, 'images')
```

```
    if(not os.path.exists(tile_dir)):
```

```
        os.mkdir(tile_dir)
```

```
    f = open(log_fname, 'r')
```

```
    lines = f.readlines()[1:]
```

```
    f.close()
```

```
    prev_fname = 'init'
```



```

'''

    # Init names and paths
img_dir = os.path.join(flight_dir, 'images')
pred_dir = list_files(flight_dir, 'predictions', '.txt')[0]
sort_logs = list_files(pred_dir, 'sorting_log_complete')
follow_up_dir = os.path.join(pred_dir, 'follow_up_candidates')
retrain_dir = os.path.join(pred_dir, 'tiles_for_retrain')

''' Check for:
        -final_prediction_log
        -sorting_log_complete
        -out_folder '''
    # Check for existing output dirs, make them if they don't exist
if(not os.path.exists(follow_up_dir)):
    os.mkdir(follow_up_dir)
if(not os.path.exists(retrain_dir)):
    os.mkdir(retrain_dir)

    # Stop program if insufficient sorting or no final log
if(len(list_files(flight_dir, 'final_prediction_log')) == 0):
    print('No final log detected...\n\tAborting...')
    return
if(len(sort_logs) < n_sorters):
    print('Insufficient completed sorting logs...\n\tAborting...')
    return

''' Make txt file for candidates'''
candid_fname = os.path.join(follow_up_dir, 'candidate_locations.txt')
candid_f = open(candid_fname, 'w')
candid_f.write('Y1\tY2\tX1\tX2\tImage')
candid_out_list = []

sort_lines = []

''' Read sorting logs '''
for i in range(len(sort_logs)):
    f = open(sort_logs[i], 'r')
    sort_lines.extend(f.readlines()[6:])

```

```

        f.close()

prev_fname = ""

''' Sort into 'follow ups' vs 'retrain' '''
    # Loop through each 'prediction'
for i in range(len(sort_lines)):
    print(sort_lines[i])
    # Yes case
    if('Y\t' in sort_lines[i]):
        sl = sort_lines[i].split('\t')

        orig_img_fname = sl[-1].replace('\n', '')
        orig_img_fname = orig_img_fname.replace('predicted_', '')
        orig_img_fname = os.path.join(img_dir, orig_img_fname)

        sname = 'candidate_' + os.path.split(orig_img_fname)[1]
        sname = os.path.join(follow_up_dir, sname)
        # Check to see if this image has previous candidates in it
        if(os.path.exists(sname)):
            IMG = PIL.Image.open(sname)
        else:
            IMG = PIL.Image.open(orig_img_fname)

            # Draw bounding box
            draw = PIL.ImageDraw.Draw(IMG)
            draw.line([int(sl[4]), int(sl[2]), int(sl[4]), int(sl[3])], fill='yellow', width=h_w)
# Left
            draw.line([int(sl[5]), int(sl[2]), int(sl[5]), int(sl[3])], fill='yellow', width=h_w)
# Right
            draw.line([int(sl[4]), int(sl[3]), int(sl[5]), int(sl[3])], fill='yellow', width=h_w)
# Bottom
            draw.line([int(sl[4]), int(sl[2]), int(sl[5]), int(sl[2])], fill='yellow', width=h_w)
# Top

            # Save image
            IMG.save(sname)

            # Save entry into candidate log
            string = '\n' + sl[2] + '\t' + sl[3] + '\t' + sl[4] + '\t' + sl[5] + '\t' +
os.path.split(sname)[1]
            candid_out_list.append(string)

```



```
def eval_model(model_path, met_folder, real_met_folder='../real_mets/tiles', full_img_folder=None,
pred_thresh=0.9, input_shape=(125,125), n_cores=5, silent=False):
```

```
    """This function takes a model, unique meteorite tiles and full images, in order to evaluate
        the percentage of meteorites it will recognize and the average number of false
positives per image.
```

Inputs:

```

        model_path          [str] full path to the .h5 model to be evaluated
        met_folder          [str] folder containing 'fake' meteorite tiles
        real_met_folder [str] fullpath to dir with real meteorite tiles, NOTE: there
should be 1 subdir which contains the tiles (blame keras)
        full_img_folder [str] folder containing full images (no meteorites)
        pred_thresh       [float] prediction threshold, preds below this are not
counted
        input_shape       [tuple] pixel shape of tiles used in the model
        silent            [bool] if False, print updates/results
```

Outputs:

```
<model_fname>_eval.txt [file] log file of the evaluation
```

```
'''
```

```
    # Load model
```

```
model = keras.models.load_model(model_path)
```

```
''' Predict on 'fake' meteorite tiles '''
```

```
'''-----'''
```

```
tile_fnames = list_files(met_folder)
```

```
tile_preds = []
```

```
pred_count = 0.0
```

```
met_log_fname = model_path.replace('.h5', '_fakemet_preds.csv')
```

```
    # Loop through each meteorite tile and predict
```

```
for i in range(len(tile_fnames)):
```

```

tile = np.asarray(PIL.Image.open(tile_fnames[i])).reshape(1, 125, 125, 3)
pred = float(model.predict(tile / 255.)[0])
tile_preds.append(pred)

if(pred >= pred_thresh):
    pred_count += 1.0

tile_fnames[i] = osplit(tile_fnames[i])[1]

# Percent of meteorite tiles labelled 'True'
tile_percent = pred_count / float(len(tile_fnames))

# Write tile predictions to .csv
df = pd.DataFrame({'pred_val': tile_preds,
                  'fname':    tile_fnames })
df.to_csv(met_log_fname, index=False)

# Make Histogram
tile_preds = np.asarray(tile_preds)

string = ""
for i in np.arange(9, 0, -1):
    n    = np.round((len(np.where(tile_preds > (i*0.1))[0]) / len(tile_preds) * 100), 2)
    string += '>0.' + str(i) + ': ' + str(n) + '\n'

plt.title((osplit(model_path)[1] + ' Met detection'))
plt.ylabel('Num Predictions')
plt.xlabel('Prediction Value')
plt.xlim((0,1.3))
plt.text(1.025, 100, string)
plt.hist(tile_preds, bins=20, log=True)
plt.savefig(met_log_fname.replace('preds.csv', 'hist.png'))

''' Predict on REAL meteorite tiles '''
'''-----'''

```

```

# Init REAL met folder
test_datagen = ImageDataGenerator(rescale=1./255)
test_generator = test_datagen.flow_from_directory(
    real_met_folder,
    target_size = (125,125),
    color_mode = 'rgb',
    shuffle      = False,
    class_mode = 'binary',
    batch_size = 400    )

# Get Fnames of real met tiles
fnames = test_generator filenames
n_tiles = len(fnames)

# Make dictionary for meteorites (group together permutations)
new_fnames = []

for i in range(len(fnames)):
    new_fnames.append('__'.join(fnames[i].split('__')[1:]))

new_fnames = list(dict.fromkeys(new_fnames))
new_preds  = [[] for i in range(len(new_fnames))]
new_stds   = [[] for i in range(len(new_fnames))]

# Predict
preds = model.predict_generator(test_generator, steps = 61)

# Group together preds based on original image fnames
for i in range(len(preds)):
    for j in range(len(new_fnames)):
        if(new_fnames[j] in fnames[i]):
            new_preds[j].append(preds[i])

# Condense preds into mean and std for each original img
for i in range(len(new_preds)):
    arr = np.asarray(new_preds[i])
    #np.where(arr > 0.9, 1, 0)
    new_stds[i] = np.std(arr)
    new_preds[i] = np.mean(arr)

```

```

# Save REAL met preds to csv
real_met_sname = model_path.replace('.h5', '_real_met_preds.csv')

df = pd.DataFrame()
df['fname'] = new_fnames
df['pred_mean'] = new_preds
df['pred_std'] = new_stds
df.sort_values(['pred_mean'], inplace=True)
df.to_csv(real_met_sname)

# Make Histogram
plt.clf()
plt.hist(preds, bins=20, log=True)
plt.savefig(real_met_sname.replace('preds.csv', 'hist.png'))

# Get size of meteorites(from 'meteorite_locations.txt' files, should be already
compiled in /real_mets)
real_met_sizes_fname = os.path.join(os.path.split(real_met_folder)[0], 'real_met_sizes.csv')
real_met_sizes_df = pd.read_csv(real_met_sizes_fname)

df['diam'] = [0 for i in range(len(df))]

# Match diams to fnames in preds df/csv
for i in range(len(df)):
    for j in range(len(real_met_sizes_df)):
        if(df['fname'].iloc[i] == real_met_sizes_df['fname'].iloc[j]):
            df['diam'].iloc[i] = real_met_sizes_df['diam'].iloc[j]

# Make Scatterplot
plt.clf()
plt.ylabel('Prediction Value')
plt.xlabel('Pixel Diameter of Meteorite')
plt.figure(figsize=(10,10))
for i in range(len(df)):
    if('Mad' in df['fname'].iloc[i]):
        plt.errorbar(df['diam'].iloc[i],
                    df['pred_mean'].iloc[i],
                    yerr=df['pred_std'].iloc[i], fmt='bo')
    if('Mun' in df['fname'].iloc[i]):
        plt.errorbar(df['diam'].iloc[i],
                    df['pred_mean'].iloc[i],
                    yerr=df['pred_std'].iloc[i], fmt='ro')

```

```

plt.savefig(real_met_sname.replace('preds.csv', 'scatter.png'))

"""
''' Predict on full images, to find false positive rate '''

    overlay_dir, tile_count, proc_rate = predict_and_highlight(model_path, full_img_folder,
input_shape=input_shape, n_cores = n_cores )
    n_false_pos = float(tile_count)
    n_full_imgs = float(len(list_files(full_img_folder)))

        # Rate of false positives per image
false_pos_avg = n_false_pos / n_full_imgs

time_str      = str(dtm.now())
time_str      = time_str.replace(' ', '_')
time_str      = time_str.replace(':', '-')
time_str      = time_str.split('.')[0]

output_str = ( time_str +
                '\nEvaluated Model: ' + os.path.split(model_path)[1] +
                '\nMeteorites Found:  ' + str(int(pred_count)) + '/' +
str(len(tile_fnames)) + ' (' + str(np.round(tile_percent*100, 1)) + '%)' +
                '\nAvg. False pos.:  ' + str(np.round(false_pos_avg, 1)) + ' '
(n_imgs=' + str(int(n_full_imgs)) + ')' + '\n' +
                '\nAvg ime per img:  ' + str(proc_rate) + ' [sec]' + '\n' )

        # Print results and write to file
if(silent != True):
    print(output_str)

sname = model_path.split('.')[0] + '_eval.txt'
sname = os.path.join(os.path.split(model_path)[0], ('eval_' + os.path.split(model_path)[1][:-
3] + '.txt'))
f = open(sname, 'a')
f.write(output_str)
f.close()

```


''''''

return #(tile_percent, false_pos_avg)

```
#####  
#####  
#####  
#####  
#####  
#####
```

def make_overview(survey_dir):

'''

This function creates a .csv overview which summarizes which directories (or flights) have been predicted on, their results, as well as progress towards sorting the false positives.

Inputs:

survey_dir [str] fullpath to the survey dir which contains subdirs containing images from a flight

Outputs:

survey_dir/overview.csv [file] Summary file

'''

pd.option_context('display.max_rows', None, 'display.max_columns', None)

flight_dirs = list_files(survey_dir, 'flight')

df = pd.DataFrame(columns=['Flights Num.', 'N Images', 'ML Prediction', 'Detections', 'Sorter A', 'Sorter B', 'N Candidates'], index=None)

```

# Loop through each flight dir
for i in range(len(flight_dirs)):
    # get n_images
    imgs_dir = ojoin(flight_dirs[i], 'images')
    imgs = len(list_files(imgs_dir))

    # check for prediction
    pred_dirs = list_files(flight_dirs[i], 'predictions')
    final_logs = list_files(flight_dirs[i], 'final')
    inter_logs = list_files(flight_dirs[i], 'interim')

    csvs_dir = list_files(pred_dirs[0], 'csvs')[0]
    if(len(final_logs) != 0):
        pred = 'Completed'
    if(len(pred_dirs) == 0):
        pred = 'Not Started'
    if(len(inter_logs) != 0):

        n_done = len(list_files(csvs_dir, '.csv'))
        pred = str(int( float(n_done) / float(imgs) * 100 )) + '%'

    # Detections
    det_fname = ojoin(pred_dirs[0], 'detections.txt')
    if(not os.path.exists(det_fname)):
        detections = 0
    if( os.path.exists(det_fname)):
        f = open(det_fname, 'r')
        detections = len(f.readlines()) - 1

    # check for sorting
    sorts = ['', '']
    sort_logs = list_files(pred_dirs[0], 'sorting_log')

    for j in range(len(sort_logs)):
        if(j == 2):
            break
        if('complete' in sort_logs[j]):
            sorts[j] = 'Completed'
        if('complete' not in sort_logs[j]):
            sorter_name = osplit(sort_logs[j])[1].replace('_sorting_log.txt', '')
            sorts[j] = sorter_name

```

```

        # check for candidates
    if('Completed' in sorts):
        com_sort_log = list_files(pred_dirs[0], 'complete')[0]
        f = open(com_sort_log, 'r')
        lines = f.readlines()
        f.close()
        cand = lines - 1

    if('Completed' not in sorts):
        cand = 0

    df = df.append({
        'Flight Num.':      osplit(flight_dirs[i])[1],
        'N Images':         imgs,
        'ML Prediction':    pred,
        'Detections':       suspects,
        'Sorter A':         sort,
        'Sorter B':         sortB,
        'N Candidates':     cand },

        ignore_index=True)

df = df.append({
    'Flight Num.': 'total',
    'N Images':    df['images'].sum(),
    'ML Prediction': '',
    'Detections': df['suspects'].sum(),
    'Sorter A':    '',
    'Sorter B':    '',
    'candidates': df['candidates'].sum() }, ignore_index=True)

sname = ojoin(survey_dir, 'overview.csv')
df.to_csv(sname, index=False)
try:
    os.system(('xdg-open ' + sname))
return

```

```
except:
    return
```

```
#####
#####
#####
#####
#####
#####
```

```
def equalize_train_set(train_dir):
```

```
    '''
```

This function moves excess tiles from 'False' /subdir to an 'Extra' one, this is good for making balanced retraining sets

```
    '''
```

```
        # Identify True/False subdirs
```

```
        true_dir = os.path.join(train_dir, 'True')
```

```
        false_dir = os.path.join(train_dir, 'False')
```

```
        # Make subdir for extra false files (to be moved into)Untitled
```

```
        extra_f_dir = os.path.join(os.path.split(train_dir)[0], 'Extra_false_equalized')
```

```
        extra_t_dir = os.path.join(os.path.split(train_dir)[0], 'Extra_true_equalized')
```

```
        if(not os.path.exists(extra_t_dir)):
```

```
            os.mkdir(extra_t_dir)
```

```
        if(not os.path.exists(extra_f_dir)):
```

```
            os.mkdir(extra_f_dir)
```

```
        # Count number of tiles to move
```

```
        n_true = len(list_files(true_dir))
```

```
        n_false = len(list_files(false_dir))
```

```
        n_extra = n_false - n_true
```

```
        # Too many true tiles
```

```
        if(n_extra < 0):
```

```
            # List false tiles and shuffle
```

```
            true_tiles = list_files(true_dir)
```

```
            random.shuffle(true_tiles)
```

```
        # Move
```

```

    for i in range(abs(n_extra)):
        tile_name = true_tiles[i]
        new_name = os.path.join(extra_t_dir, (str(i) + '.jpg'))
        os.rename(tile_name, new_name)

    # Too many false tiles
    if(n_extra > 0):
        # List false tiles and shuffle
        false_tiles = list_files(false_dir)
        random.shuffle(false_tiles)

    # Move
    for i in range(n_extra):
        tile_name = false_tiles[i]
        new_name = os.path.join(extra_f_dir, (str(i) + '.jpg'))
        os.rename(tile_name, new_name)

    return

```

```

# -*- coding: utf-8 -*-
"""

```

Created on Mon Mar 4 21:07:34 2019

```

@author: 19418948
"""

```

```

import os

```

```

import tkinter
from datetime import datetime as dtm
import time
import numpy as np
import multiprocessing as mp

```

```

from tkinter import Tk, Label, Button, filedialog, Entry, Canvas, IntVar
from PIL import ImageTk, Image, ImageOps, ImageDraw

```

```

#####
#####
#####
#####
#####
#####

```

```

# This is the directory to the images thrown up every 5-10 frames in a row

```

```

global human_test_dir

```

```

human_test_dir = r'/data0/Seamus_data/MET_searching/keras_train/human_test_dir'

```

```

#####
#####
#####
#####
#####
#####

```

```

def list_files(folder, f_type='.jpg', exclude=None):

```

```

    """Takes a folder and file extension, and returns all the filenames (fullpaths) in that
        directory with that extension.

```

```

    ** NOTE this can work for any sub-dir or file if you replace the extension with what **
    * you want to look for

```

```

        *

```

```

Inputs:

```

```

    folder [str] full path to the folder to be searched
    f_type [str] file extension including the '.'
    exclude [str] key word or phrase to exclude when looking for files

```

```

Outputs

```

```

    files [list] full path to files containing 'f_type' extension

```

```

'''

```

```

    # List contents of the given 'folder'
    contents = os.listdir(folder)
    # Prep list for files to be extracted

```

```
files = []
    # Loops through all the items found in 'folder'
    # adds them to the return list 'files', if file extension matches
for i in range(len(contents)):
    if(f_type.upper() in contents[i].upper()):
        files.append(os.path.join(folder, contents[i]))
    if((exclude != None) and (exclude.upper() in contents[i].upper())):
        files.remove(os.path.join(folder, contents[i]))

return files
```

```
#####
#####
#####
#####
#####

#####
#####
#####
#####
#####
```

class Boulder:

```
def __init__(self, master):

    ''' Startup window, username input, flight select '''
    '''-----'''

    # Create window
```

```

self.master = master
master.title("Boulder")

        # Print welcome message
global user
global welcome
init_str = '''This interface is designed to help meteorite searchers sort through
tiles deemed to have a high probability of containing a meteorite.

Please enter your name to get started.'''

welcome = Label(master, text=init_str)
welcome.pack()

        # Text entry box for username
self.user = Entry(master)
self.user.pack()

        # Set up Button for selecting folder, which will later start
self.proceed_b = Label(master, text="Press <Return> to select meteorite tiles
directory")
self.proceed_b.pack()

master.bind("<Return>", self.proceed)

#####
#####
#####
#####
#####

        # This function is where the meat of the sorting happens
def proceed(self, master):
    global you_fucked_up_fname
    global extra

        # This displays the real meteorites on the periphery for reference
#=====

```



```

#you_fucked_up_fname =
'/data0/Seamus_data/MET_searching/keras_train/drone_searching/gui/graphics/misc/fucked_up1.g
if'

you_fucked_up_fname = 'graphics/misc/you-fucked-up.jpg'
bun1_fname = 'graphics/display_examples/bunburra_1_alpha.png'
bun2_fname = 'graphics/display_examples/bunburra_2_alpha.png'
murr_fname = 'graphics/display_examples/Murilli_alpha.png'

# How many extra pixels to pad the suspect tile with
extra = 70 # tile dimension (125, 125, 3)

# Make real meteorite size scale bars
img_scale = 5 # [pix/cm]
img_scale = 1.5 # [mm/pix]

size_lims = np.array([75, 57.5, 40]) # Upp->Med->Low [mm] (diameter)
size_lims /= img_scale # size lims is now in

[pix]

bun1 = Image.open(bun1_fname)
bun2 = Image.open(bun2_fname)
murr = Image.open(murr_fname)

bun1_r = bun1.size[0] / bun1.size[1]
bun2_r = bun2.size[0] / bun2.size[1]
murr_r = murr.size[0] / murr.size[1]

newsize1 = ( int(size_lims[2]), int(size_lims[2]/bun1_r) )
newsize2 = ( int(size_lims[1]), int(size_lims[1]/bun2_r) )
newsize3 = ( int(size_lims[0]), int(size_lims[0]/murr_r) )

bun1 = bun1.resize(newsize1)
bun2 = bun2.resize(newsize2)
murr = murr.resize(newsize3)

```

```
self.bun1 = ImageTk.PhotoImage(bun1)
self.bun2 = ImageTk.PhotoImage(bun2)
self.murr = ImageTk.PhotoImage(murr)
```

```
#=====
```

```
global sorting_log
global time_log_fname
global pred_lines
global img_dir
global not_folder
global met_folder
global t0
global test_fname
```

```
global last_yes
last_yes = IntVar()
last_yes.set(0)
```

```
    # This is the user's position in the predictions
```

```
global j
j = IntVar()
j.set(0)
```

```
    #human_test_dir =
r'E:\ML\time_of_day_training\synth_sets\Feb15_sets\all_times_comp\True'
    global human_test_fnames
    human_test_fnames = list_files(human_test_dir)
    #input_folder =
'/hdd/Forrest_survey_dir_SSD1/Flight_01/no_doubles_predictions_2020-06-03_13-27-04'
```

```
    # Record input folder selection from user
    # and get file filenames
os.chdir('/data0')
```

```

input_folder = filedialog.askdirectory()
img_dir     = os.path.join(input_folder, 'suspect_images')
pred_fname  = os.path.join(input_folder, 'detections.txt')
pred_file   = open(pred_fname, 'r')
pred_lines  = pred_file.readlines()[1:]
pred_file.close()

```

```

# Get entered username
username = self.user.get()

```

```

# Delete welcome message, username text box, and folder select button
self.user.pack_forget()
self.proceed_b.pack_forget()
welcome.pack_forget()

```

```

#####
''' Make or resume logs '''
'''-----'''
#####

```

```

# Make sorting log for this user
sorting_log = os.path.join(input_folder, (username + '_sorting_log.txt'))

```

```

# Check for previous sorting log
if(os.path.exists(sorting_log) == True):
    # Read and split by entries
    f = open(sorting_log, 'r')
    log_content = f.readlines()
    f.close()

```

**# Loop through each sorting entry to find the most recently sorted jpg,
note this is a decrement for loop**

```

for i in range(len(log_content)-1, 0, -1):
    # Check for a real file log line
    if(('jpg' in log_content[i]) or
       ('JPG' in log_content[i])):
        last_line = log_content[i]
        # find the index value for pred_lines

```



```

# Determine number of test tiles this round, and in what grid position they
will be
global num_test_tiles
global test_positions
global user_responses
global first_set
global n_fuck_ups

n_fuck_ups = 0

first_set = True

num_test_tiles = IntVar()
num_test_tiles.set(np.random.randint(0,3))

user_responses = []
if(num_test_tiles.get() > 0):
    test_positions = np.random.randint(0, 9, num_test_tiles.get())
else:
    test_positions = []

# Designate current real samples to be shown, accounting for how many test
there will be
global current_lines
current_lines = pred_lines[j.get() : (j.get() + 9 - num_test_tiles.get())]
x = 0 # x is an iterator through this set of real tiles

t0 = dtm.now()
    ## Display each tile; real sample or otherwise
    # Start by making a blank PIL Image instance
DISP_IMG = Image.new('RGB', (835,835), color='white')

    # Allocate lists to store this batch's tiles and xy corrdts
tiles = mp.Manager().list(range(9))

    # Write function for loading each tile, later used in multi-
processing
def get_tile(img_fname, xys, i):
    # load full image
    full_img = ImageOps.expand(Image.open(img_fname), border= (3*extra),

```

fill='black')

```
tile = Image.fromarray(np.asarray(full_img)[ xys[0]:xys[1], xys[2]:xys[3] ])
draw = ImageDraw.Draw(tile)
draw.text([10,10], (str(i+1) + ' - ' + os.path.split(img_fname)[1] ))
tiles[i] = tile
return
```

jobs = []

Get each sus tile or test tile

for i in range(9):

CASE: Test tile

if(i in test_positions):

```
test_txt_fname = list_files(human_test_dir,
'candidate_locations.txt')[0]
```

```
test_f = open(test_txt_fname, 'r')
```

```
test_lines = test_f.readlines()[1:]
```

```
test_f.close()
```

```
test_line = test_lines[ np.random.randint(0, len(test_lines)-1) ]
```

```
sl = test_line.split('\t')
```

```
img_fname = os.path.join(human_test_dir, sl[-1].replace('\n', ''))
```

```
y1 = int(sl[0]) - extra + (3 * extra)
```

```
y2 = int(sl[1]) + extra + (3 * extra)
```

```
x1 = int(sl[2]) - extra + (3 * extra)
```

```
x2 = int(sl[3]) + extra + (3 * extra)
```

```
xys = (y1,y2,x1,x2)
```

CASE: Real sample

else:

```
sl = (current_lines[x].replace('\n', '')).split('\t')
```

```
x += 1
```

```
img_fname = os.path.join(img_dir, sl[-1])
```

```
y1 = int(sl[0]) - extra + (3 * extra)
```

```
y2 = int(sl[1]) + extra + (3 * extra)
```

```
x1 = int(sl[2]) - extra + (3 * extra)
```

```
x2 = int(sl[3]) + extra + (3 * extra)
```

```
xys = (y1,y2,x1,x2)
```

```
p = mp.Process(target = get_tile,
```

```

                                args = [img_fname, xys, i]
    jobs.append(p)
    p.start()

for job in jobs:
    job.join()

    # paste tiles onto display image
for i in range(9):
    if(i==0):
        paste_pos = (10, 560)
    if(i==1):
        paste_pos = (285, 560)
    if(i==2):
        paste_pos = (560, 560)
    if(i==3):
        paste_pos = (10, 285)
    if(i==4):
        paste_pos = (285, 285)
    if(i==5):
        paste_pos = (560, 285)
    if(i==6):
        paste_pos = (10, 10)
    if(i==7):
        paste_pos = (285, 10)
    if(i==8):
        paste_pos = (560, 10)

    DISP_IMG.paste(tiles[i], paste_pos)

self.DISP_IMG = ImageTk.PhotoImage(DISP_IMG)

global win_w
global win_h
win_w = 1000
win_h = 1000

disp_text = 'Number of Fuck Ups: ' + str(n_fuck_ups)

# Make Window

```

```

self.canvas = Canvas(self.master, width=win_w, height=win_h)
self.canvas.create_text( win_w/2, int(0.06 * win_h), anchor='center', text=disp_text)
self.canvas.create_image(win_w/2, win_h/2, anchor='center',
image=self.DISP_IMG)
self.canvas.create_image(int(win_w /4), int(win_h*0.95), anchor='center',
image=self.bun1)
self.canvas.create_image(int(win_w /2), int(win_h*0.95), anchor='center',
image=self.bun2)
self.canvas.create_image(int(win_w*3/4), int(win_h*0.95), anchor='center',
image=self.murr)
self.canvas.pack()
t0 = dtm.now()

```

Binding functions to keystrokes

```

self.master.bind('<KP_1>', self.s1)
self.master.bind('<KP_2>', self.s2)
self.master.bind('<KP_3>', self.s3)
self.master.bind('<KP_4>', self.s4)
self.master.bind('<KP_5>', self.s5)
self.master.bind('<KP_6>', self.s6)
self.master.bind('<KP_7>', self.s7)
self.master.bind('<KP_8>', self.s8)
self.master.bind('<KP_9>', self.s9)

self.master.bind('<KP_Enter>', self.advance)
self.master.bind('<KP_Subtract>', self.retreat)
self.master.bind('<BackSpace>', self.strike)

```

```

global n_cands
n_cands = 0

```

```

global n_tests
n_tests = int(num_test_tiles.get())

```

```

global start_time
start_time = dtm.now()

```

```

#####
#####
#####

```



```

def s1(self, master):
    print('1')
    user_responses.append('1')
def s2(self, master):
    print('2')
    user_responses.append('2')
def s3(self, master):
    print('3')
    user_responses.append('3')
def s4(self, master):
    print('4')
    user_responses.append('4')
def s5(self, master):
    print('5')
    user_responses.append('5')
def s6(self, master):
    print('6')
    user_responses.append('6')
def s7(self, master):
    print('7')
    user_responses.append('7')
def s8(self, master):
    print('8')
    user_responses.append('8')
def s9(self, master):
    print('9')
    user_responses.append('9')

def strike(self, master):
    if(len(user_responses) == 0):
        return
    else:
        print('Removing "', user_responses[-1], '" from user_responses ')
        user_responses.pop()

```

```

#####
#####

```

```
#####
```

```
def advance(self, master):
    print('Advance')
    global n_fuck_ups
    global user_responses
    global test_positions
    global first_set
    global n_cands
    global n_tests
    global num_test_tiles
    global current_lines

    first_set = False

    # Eliminate duplicates
    yes_tiles = list(set(user_responses))

    # Check for test(s)
    for i in range(len(test_positions)):
        if(str(test_positions[i]+1) not in yes_tiles):
            n_fuck_ups += 1

    # Record Yes-No status of every real tile
    x = 0 # x is kind of like an iterator within each set of displayed samples, with: global
    j acting as the first in the set
    for i in range( num_test_tiles.get() + len(current_lines) ):
        if(i in test_positions):
            continue
        if(str(i+1) in yes_tiles):
            f = open(sorting_log, 'a+')
            new_line = ( 'Y\t\t' + pred_lines[j.get()+x] )
            f.write(new_line)
            f.close()
            x += 1
            n_cands += 1
        if(str(i+1) not in yes_tiles):
            f = open(sorting_log, 'a+')
            new_line = ( '\tN\t\t' + pred_lines[j.get()+x] )
```

```
f.write(new_line)
f.close()
x += 1
```

```
# Set start iterator for new set
j.set(j.get() + x )
```

```
# Check if you're done
if(j.get() >= (len(pred_lines) - 1) ):
    os.rename(sorting_log, sorting_log.replace('.', '_complete.'))
    print('Done sorting this flight!')
    exit()
```

```
# Clear old display
self.canvas.pack_forget()
```

```
# Make New Display
```

```
# Determine number of test tiles this round, and in what grid position
```

they will be

```
num_test_tiles = IntVar()
num_test_tiles.set(np.random.randint(0,3))
```

```
user_responses = []
if(num_test_tiles.get() > 0):
    test_positions = np.random.randint(0, 9, num_test_tiles.get())
    if(len(test_positions) == 2):
        while(test_positions[0] == test_positions[1]):
            test_positions[1] = np.random.randint(0, 9)
else:
    test_positions = []
```

there will be

```
# Designate current real samples to be shown, accounting for how many test
```

```
try:
```

```

        current_lines = pred_lines[j.get() : (j.get() + 9 - num_test_tiles.get())]
        print('current_lines ',len(current_lines))
except:
    current_lines = pred_lines[j.get():]
    print('Last Line, current_lines ', len(current_lines))

x = 0 # x is an iterator through this set of real tiles

t0 = dtm.now()
    ## Display each tile, real sample or otherwise
    # Start by making a blank PIL Image instance
DISP_IMG = Image.new('RGB', (835,835), color='white')

    # Allocate lists to store this batch's tiles and xy corrdts
tiles = mp.Manager().list(range(len(current_lines) + num_test_tiles.get()))

    # Write function for loading each tile, later used in multi-
processing
def get_tile(img_fname, xys, i):
    # load full image
    full_img = ImageOps.expand(Image.open(img_fname), border= (3*extra),
fill='black')
    tiles[i] = Image.fromarray(np.asarray(full_img)[ xys[0]:xys[1],
xys[2]:xys[3] ])

jobs = []
for i in range(len(current_lines) + num_test_tiles.get()):
    # Test tile case
    if(i in test_positions):
        test_txt_fname = list_files(human_test_dir,
'candidate_locations.txt')[0]
        test_f = open(test_txt_fname, 'r')
        test_lines = test_f.readlines()[1:]
        test_f.close()
        test_line = test_lines[ np.random.randint(0, len(test_lines)-1) ]
        sl = test_line.split('\t')
        img_fname = os.path.join(human_test_dir, sl[-1].replace('\n', ''))
        y1 = int(sl[0]) - extra + (3 * extra)
        y2 = int(sl[1]) + extra + (3 * extra)

```

```

        x1 = int(sl[2]) - extra + (3 * extra)
        x2 = int(sl[3]) + extra + (3 * extra)
        xys = (y1,y2,x1,x2)

        # Real sample case
    else:
        try:
            sl = (current_lines[x].replace('\n', '')).split('\t')
        except:
            print('Read line error, i: ', str(i) , ' x: ', str(x), '\n',
'\n'.join(current_lines))

            print(test_positions)
            exit()

        x += 1
        img_fname = os.path.join(img_dir, sl[-1])
        y1 = int(sl[0]) - extra + (3 * extra)
        y2 = int(sl[1]) + extra + (3 * extra)
        x1 = int(sl[2]) - extra + (3 * extra)
        x2 = int(sl[3]) + extra + (3 * extra)
        xys = (y1,y2,x1,x2)

    p = mp.Process(target = get_tile,
                    args = [img_fname, xys, i])

    jobs.append(p)
    p.start()

for job in jobs:
    job.join()

    # paste tiles onto display image
for i in range(9):
    if(i==0):
        paste_pos = (10, 560)
    if(i==1):
        paste_pos = (285, 560)
    if(i==2):
        paste_pos = (560, 560)
    if(i==3):
        paste_pos = (10, 285)

```

```

if(i==4):
    paste_pos = (285, 285)
if(i==5):
    paste_pos = (560, 285)
if(i==6):
    paste_pos = (10, 10)
if(i==7):
    paste_pos = (285, 10)
if(i==8):
    paste_pos = (560, 10)

```

```

DISP_IMG.paste(tiles[i], paste_pos)

```

```

self.DISP_IMG = ImageTk.PhotoImage(DISP_IMG)

```

```

global win_w
global win_h

```

```

win_w = 1000

```

```

win_h = 1000

```

```

dt = dtm.now() - start_time
dt = str(dt)
sp = dt.split('.')[0]
sp = sp.split(':')
sp = sp[1] + ':' + sp[2] + ' [mm:ss]'

```

```

disp_text = ('Elapsed Time:      \t' + sp + '\n' +
             'Passed Tests:      \t' + str(n_tests - n_fuck_ups ) + ' \ ' +
str(n_tests) + ' (' + str(np.round((n_tests + 1 - n_fuck_ups)/(n_tests+1)*100,2)) + '%)\n' +
             'Candidates Identified:\t' + str(n_cands) + '\n' +
             'Progress:          \t' + str(j.get()) + ' / ' + str(len(pred_lines))
+ ' ' + str(np.round(j.get() / len(pred_lines)* 100, 4) ) + '%')

```

```

# Make Window

```

```

self.canvas = Canvas(self.master, width=win_w, height=win_h)
self.canvas.create_text( win_w/2, int(0.04 * win_h), anchor='center', text=disp_text)

```

```

        self.canvas.create_image(win_w/2, win_h/2, anchor='center',
image=self.DISP_IMG)
        self.canvas.create_image(int(win_w /4), int(win_h*0.95), anchor='center',
image=self.bun1)
        self.canvas.create_image(int(win_w /2), int(win_h*0.95), anchor='center',
image=self.bun2)
        self.canvas.create_image(int(win_w*3/4), int(win_h*0.95), anchor='center',
image=self.murr)
        self.canvas.pack()

```

```

t0 = dtm.now()

```

Binding functions to keystrokes

```

self.master.bind('<KP_1>', self.s1)
self.master.bind('<KP_2>', self.s2)
self.master.bind('<KP_3>', self.s3)
self.master.bind('<KP_4>', self.s4)
self.master.bind('<KP_5>', self.s5)
self.master.bind('<KP_6>', self.s6)
self.master.bind('<KP_7>', self.s7)
self.master.bind('<KP_8>', self.s8)
self.master.bind('<KP_9>', self.s9)

```

```

self.master.bind('<KP_Enter>', self.advance)
self.master.bind('<KP_Subtract>', self.retreat)

```

```

n_tests += int(num_test_tiles.get())

```

```

#####
#####
#####

```

```

def retreat(self, master):
    global first_set
    global n_cands

```

```

global n_tests
global test_positions

if(first_set == True):
    print('First set, cannot go back!')
    return
else:
    print('Retreat')
first_set = True

    # Set start iterator for previous 9 samples
j.set(j.get() - 9)

    # Clear last 9 entries in the log file
f = open(sorting_log, 'r')
lines = f.readlines()[:-9]
f.close()

f = open(sorting_log, 'w')
for line in lines:
    f.write(line)
f.close()

    # Clear old display
self.canvas.pack_forget()

    # Make New Display

    # Skip test tile allocations
global num_test_tiles

num_test_tiles = IntVar()
num_test_tiles.set(0)

user_responses = []

test_positions = []

```



```

        # Designate current real samples to be shown
global current_lines
current_lines = pred_lines[j.get() : (j.get() + 9) ]
x = 0 # x is an iterator through this set of real tiles

t0 = dtm.now()
    ## Display each tile, real sample or otherwise
    # Start by making a blank PIL Image instance
DISP_IMG = Image.new('RGB', (835,835), color='white')

        # Allocate lists to store this batch's tiles and xy corrdrs
tiles = mp.Manager().list(range(9))

        # Write function for loading each tile, later used in multi-
processing
def get_tile(img_fname, xys, i):
    # load full image
    full_img = ImageOps.expand(Image.open(img_fname), border= (3*extra),
fill='black')
    tiles[i] = Image.fromarray(np.asarray(full_img)[ xys[0]:xys[1],
xys[2]:xys[3] ])

jobs = []
for i in range(9):
    # Test tile case
    if(i in test_positions):
        test_txt_fname = list_files(human_test_dir,
'candidate_locations.txt')[0]
        test_f = open(test_txt_fname, 'r')
        test_lines = test_f.readlines()[1:]
        test_f.close()
        test_line = test_lines[ np.random.randint(0, len(test_lines)-1) ]
        sl = test_line.split('\t')
        img_fname = os.path.join(human_test_dir, sl[-1].replace('\n', ''))
        y1 = int(sl[0]) - extra + (3 * extra)
        y2 = int(sl[1]) + extra + (3 * extra)
        x1 = int(sl[2]) - extra + (3 * extra)
        x2 = int(sl[3]) + extra + (3 * extra)
        xys = (y1,y2,x1,x2)

```

```

        # Real sample case
    else:
        sl = (current_lines[x].replace('\n', '')).split('\t')
        x += 1
        img_fname = os.path.join(img_dir, sl[-1])
        y1 = int(sl[0]) - extra + (3 * extra)
        y2 = int(sl[1]) + extra + (3 * extra)
        x1 = int(sl[2]) - extra + (3 * extra)
        x2 = int(sl[3]) + extra + (3 * extra)
        xys = (y1,y2,x1,x2)

    p = mp.Process(target = get_tile,
                  args = [img_fname, xys, i])
    jobs.append(p)
    p.start()

for job in jobs:
    job.join()

```

```

# paste tiles onto display image
for i in range(9):
    if(i==0):
        paste_pos = (10, 560)
    if(i==1):
        paste_pos = (285, 560)
    if(i==2):
        paste_pos = (560, 560)
    if(i==3):
        paste_pos = (10, 285)
    if(i==4):
        paste_pos = (285, 285)
    if(i==5):
        paste_pos = (560, 285)
    if(i==6):
        paste_pos = (10, 10)
    if(i==7):
        paste_pos = (285, 10)
    if(i==8):
        paste_pos = (560, 10)

```

```

        DISP_IMG.paste(tiles[i], paste_pos)

self.DISP_IMG = ImageTk.PhotoImage(DISP_IMG)

global win_w
global win_h
win_w = 1000
win_h = 1000

disp_text = ('Number of Passed Tests: ' + str(n_tests - n_fuck_ups) + '\ ' + str(n_tests)
+ ' (' + str(np.round((n_tests - n_fuck_ups)/n_tests,2)) + '%)\n' +
            str(j.get()) + ' / ' + str(len(pred_lines)) + ' ' +
str(np.round(j.get() / len(pred_lines) * 100, 4) ) + '%')
        # Make Window
self.canvas = Canvas(self.master, width=win_w, height=win_h)
self.canvas.create_text( win_w/2, int(0.06 * win_h), anchor='center', text=disp_text)
self.canvas.create_image(win_w/2, win_h/2, anchor='center',
image=self.DISP_IMG)
self.canvas.create_image(int(win_w /4), int(win_h*0.95), anchor='center',
image=self.bun1)
self.canvas.create_image(int(win_w /2), int(win_h*0.95), anchor='center',
image=self.bun2)
self.canvas.create_image(int(win_w*3/4), int(win_h*0.95), anchor='center',
image=self.murr)
self.canvas.pack()
t0 = dtm.now()

        # Binding functions to keystrokes
self.master.bind('<KP_1>', self.s1)
self.master.bind('<KP_2>', self.s2)
self.master.bind('<KP_3>', self.s3)
self.master.bind('<KP_4>', self.s4)
self.master.bind('<KP_5>', self.s5)
self.master.bind('<KP_6>', self.s6)
self.master.bind('<KP_7>', self.s7)
self.master.bind('<KP_8>', self.s8)
self.master.bind('<KP_9>', self.s9)

self.master.bind('<KP_Enter>', self.advance)
self.master.bind('<KP_Subtract>', self.retreat)

```

```
#####  
#####  
#####
```

```
root = Tk()  
my_gui = Boulder(root)  
root.mainloop()
```

```
# -*- coding: utf-8 -*-  
# Advanced zoom example. Like in Google Maps.  
# It zooms only a tile, but not the whole image. So the zoomed tile occupies  
# constant memory and not crams it with a huge resized image for the large zooms.  
import random  
import tkinter as tk  
from tkinter import ttk  
from PIL import Image, ImageTk  
import os  
  
class AutoScrollbar(ttk.Scrollbar):  
    ''' A scrollbar that hides itself if it's not needed.  
        Works only if you use the grid geometry manager '''  
    def set(self, lo, hi):  
        if float(lo) <= 0.0 and float(hi) >= 1.0:  
            self.grid_remove()  
        else:  
            self.grid()  
            ttk.Scrollbar.set(self, lo, hi)  
  
    def pack(self, **kw):  
        raise tk.TclError('Cannot use pack with this widget')
```

```

def place(self, **kw):
    raise tk.TclError('Cannot use place with this widget')

class Zoom_Advanced(tk.Frame):
    """ Advanced zoom of the image """
    def __init__(self, mainframe, path):
        """ Initialize the main Frame """
        tk.Frame.__init__(self, master=mainframe)

        title = os.path.split(path)[1]

        self.master.title(title)
        # Vertical and horizontal scrollbars for canvas
        vbar = AutoScrollbar(self.master, orient='vertical')
        hbar = AutoScrollbar(self.master, orient='horizontal')
        vbar.grid(row=0, column=1, sticky='ns')
        hbar.grid(row=1, column=0, sticky='we')
        # Create canvas and put image on it
        self.canvas = tk.Canvas(self.master, highlightthickness=0,
                                xscrollcommand=hbar.set,
                                yscrollcommand=vbar.set)
        self.canvas.grid(row=0, column=0, sticky='nswe')
        self.canvas.update() # wait till canvas is created
        vbar.configure(command=self.scroll_y) # bind scrollbars to the canvas
        hbar.configure(command=self.scroll_x)
        # Make the canvas expandable
        self.master.rowconfigure(0, weight=1)
        self.master.columnconfigure(0, weight=1)
        # Bind events to the Canvas
        self.canvas.bind('<Configure>', self.show_image) # canvas is resized
        self.canvas.bind('<ButtonPress-1>', self.move_from)
        self.canvas.bind('<B1-Motion>', self.move_to)
        self.canvas.bind('<MouseWheel>', self.wheel) # with Windows and MacOS, but not
Linux
        self.canvas.bind('<Button-5>', self.wheel) # only with Linux, wheel scroll down
        self.canvas.bind('<Button-4>', self.wheel) # only with Linux, wheel scroll up
        self.canvas.bind('y', self.YES)
        self.canvas.bind('n', self.NO)
        self.image = Image.open(path) # open image
        self.width, self.height = self.image.size
        self.imscale = 1.0 # scale for the canvaas image

```

```

self.delta = 1.3 # zoom magnitude
# Put image into container rectangle and use it to set proper coordinates to the image
self.container = self.canvas.create_rectangle(0, 0, self.width, self.height, width=0)
# Plot some optional random rectangles for the test purposes

'''
minsize, maxsize, number = 5, 20, 10
for n in range(number):
    x0 = random.randint(0, self.width - maxsize)
    y0 = random.randint(0, self.height - maxsize)
    x1 = x0 + random.randint(minsize, maxsize)
    y1 = y0 + random.randint(minsize, maxsize)
    color = ('red', 'orange', 'yellow', 'green', 'blue')[random.randint(0, 4)]
    self.canvas.create_rectangle(x0, y0, x1, y1, fill=color, activefill='black')
self.show_image()

'''

def scroll_y(self, *args, **kwargs):
    ''' Scroll canvas vertically and redraw the image '''
    self.canvas.yview(*args, **kwargs) # scroll vertically
    self.show_image() # redraw the image

def scroll_x(self, *args, **kwargs):
    ''' Scroll canvas horizontally and redraw the image '''
    self.canvas.xview(*args, **kwargs) # scroll horizontally
    self.show_image() # redraw the image

def move_from(self, event):
    ''' Remember previous coordinates for scrolling with the mouse '''
    self.canvas.scan_mark(event.x, event.y)

def move_to(self, event):
    ''' Drag (move) canvas to the new position '''
    self.canvas.scan_dragto(event.x, event.y, gain=1)
    self.show_image() # redraw the image

def YES():
    print('Yes')

def NO():

```

```
print('No')
```

```
def wheel(self, event):  
    ''' Zoom with mouse wheel '''  
    x = self.canvas.canvasx(event.x)  
    y = self.canvas.canvasy(event.y)  
    bbox = self.canvas.bbox(self.container) # get image area  
    if bbox[0] < x < bbox[2] and bbox[1] < y < bbox[3]: pass # Ok! Inside the image  
    else: return # zoom only inside image area  
    scale = 1.0  
    # Respond to Linux (event.num) or Windows (event.delta) wheel event  
    if event.num == 5 or event.delta == -120: # scroll down  
        i = min(self.width, self.height)  
        if int(i * self.imscale) < 30: return # image is less than 30 pixels  
        self.imscale /= self.delta  
        scale /= self.delta  
    if event.num == 4 or event.delta == 120: # scroll up  
        i = min(self.canvas.winfo_width(), self.canvas.winfo_height())  
        if i < self.imscale: return # 1 pixel is bigger than the visible area  
        self.imscale *= self.delta  
        scale *= self.delta  
    self.canvas.scale('all', x, y, scale, scale) # rescale all canvas objects  
    self.show_image()
```

```
def show_image(self, event=None):  
    ''' Show image on the Canvas '''  
    bbox1 = self.canvas.bbox(self.container) # get image area  
    # Remove 1 pixel shift at the sides of the bbox1  
    bbox1 = (bbox1[0] + 1, bbox1[1] + 1, bbox1[2] - 1, bbox1[3] - 1)  
    bbox2 = (self.canvas.canvasx(0), # get visible area of the canvas  
            self.canvas.canvasy(0),  
            self.canvas.canvasx(self.canvas.winfo_width()),  
            self.canvas.canvasy(self.canvas.winfo_height()))  
    bbox = [min(bbox1[0], bbox2[0]), min(bbox1[1], bbox2[1]), # get scroll region box  
           max(bbox1[2], bbox2[2]), max(bbox1[3], bbox2[3])]  
    if bbox[0] == bbox2[0] and bbox[2] == bbox2[2]: # whole image in the visible area  
        bbox[0] = bbox1[0]  
        bbox[2] = bbox1[2]  
    if bbox[1] == bbox2[1] and bbox[3] == bbox2[3]: # whole image in the visible area  
        bbox[1] = bbox1[1]  
        bbox[3] = bbox1[3]
```

```

self.canvas.configure(scrollregion=bbox) # set scroll region
x1 = max(bbox2[0] - bbox1[0], 0) # get coordinates (x1,y1,x2,y2) of the image tile
y1 = max(bbox2[1] - bbox1[1], 0)
x2 = min(bbox2[2], bbox1[2]) - bbox1[0]
y2 = min(bbox2[3], bbox1[3]) - bbox1[1]
if int(x2 - x1) > 0 and int(y2 - y1) > 0: # show image if it in the visible area
    x = min(int(x2 / self.imscale), self.width) # sometimes it is larger on 1 pixel...
    y = min(int(y2 / self.imscale), self.height) # ...and sometimes not
    image = self.image.crop((int(x1 / self.imscale), int(y1 / self.imscale), x, y))
    imagetk = ImageTk.PhotoImage(image.resize((int(x2 - x1), int(y2 - y1))))
    imageid = self.canvas.create_image(max(bbox2[0], bbox1[0]), max(bbox2[1],
bbox1[1]),
                                                                    anchor='nw',
image=imagetk)
    self.canvas.lower(imageid) # set image into background
    self.canvas.imagetk = imagetk # keep an extra reference to prevent garbage-
collection
    print('reached end of show_img()')

'''

path =
'/data0/Seamus_data/MET_searching/keras_train/full_images_all/Lake_Grace_1/DJI_0046.JPG' #
place path to your image here
root = tk.Tk()
app = Zoom_Advanced(root, path=path)
root.mainloop()

'''

```


Appendix B (Meteorite Classification Code)

'''

Seamus Anderson
29 Jan 2020

This library is used to process data from meteorite analyses.

'''

```
import gc
import numpy as np
import PIL.Image
import os
import sys
import pandas as pd
import keras
import matplotlib.pyplot as plt

from keras.models import Sequential
from keras.layers import Dense
from keras.utils import to_categorical

import matplotlib.pyplot as plt

import skimage

PIL.Image.MAX_IMAGE_PIXELS = None
```

```
#####
#####
#####
#####
#####
#####
```

```

def list_files(folder, f_type='.jpg', exclude=None):
    """Takes a folder and file extension, and returns all the filenames (fullpaths) in that
        directory with that extension.

        ** NOTE this can work for any sub-dir or file if you replace the extension with what **
        * you want to look for
            *

Inputs:
    folder [str] full path to the folder to be searched
    f_type [str] file extension including the '.'
    exclude [str] key word or phrase to exclude when looking for files

Outputs
    files [list] full path to files containing 'f_type' extension
    ...

    # List contents of the given 'folder'
    contents = os.listdir(folder)
    # Prep list for files to be extracted
    files = []
    # Loops through all the items found in 'folder'
    # adds them to the return list 'files', if file extension matches
    for i in range(len(contents)):

        if(f_type.upper() in contents[i].upper()):
            files.append(os.path.join(folder, contents[i]))
        if((exclude != None) and
            (exclude.upper() in contents[i].upper()) and
            (contents[i] in files)):

            files.remove(os.path.join(folder, contents[i]))

    return sorted(files)

```

```
#####
#####
#####
#####
#####
#####
```

```
def crop_imgs(folder, out_folder, xys, f_type='.jpg'):
```

```
    ''' This crops images all images contained in 'folder' according the bounds 'xys'.
```

```
    Inputs:
```

```
        folder [str]  fullpath to the folder of images
        out_folder [str]  fullpath to destination of the images
        xys [array]  bounds of the desired crop [x1,x2,y1,y2]
        f_type [str]  extension of the file type (with '.' included)
```

```
    '''
```

```
        # Make output folder
```

```
        if(not os.path.exists(out_folder)):
```

```
            os.mkdir(out_folder)
```

```
        # Get filenames fo images to crop
```

```
        fnames = list_files(folder, f_type=f_type)
```

```
        # Loop through each file and crop
```

```
        for i in range(len(fnames)):
```

```
            sys.stdout.write( ( '\r' + str(i+1) + ' / ' + str(len(fnames)) ) )
```

```
            sys.stdout.flush()
```

```
            sname = os.path.join(out_folder, ('cropped_' + os.path.split(fnames[i])[1]) )
```

```
            img = PIL.Image.open(fnames[i])
```

```
            simg = img.crop((xys[0], xys[2], xys[1], xys[3]))
```

```
            simg.save(sname)
```

```
        print('\n\n')
```

return

```
#####  
#####  
#####  
#####  
#####  
#####
```

def prep_TIMA_imgs(dirr, f_type='.png', elem_delim='-'):

''' Prepares TIMA images into mineral maps to be predicted on. Images can be any size so long as they are the same size.

This func assumes each element map is reported on all 3 color channels eg: [122,122,122], [241,241,241], from 0-255.

Element maps must be listed like 'Ca-' for Calcium or 'S-' for Sulfur', name your files accordingly

It makes use of the following element maps: Ca, Si, Mg, S, Fe, Cr

This func is optimized for ### Ordinary Chondrites ###

'''

print('### Making Mineral Map ###')

Fe_color = (100, 100, 100) # Grey

Cr_color = (230, 30, 145) # Pink

Tr_color = (250, 160, 3) # Orange

Threshold Fe metal

Fe_thresh = 200

#Make save folder

save_dir = os.path.join(dirr, 'generated_files')

if(not os.path.exists(save_dir)):

os.mkdir(save_dir)

```

    # Get filenames
map_fnames = list_files(dirr, f_type=f_type, exclude='.hdr')

    # Open images
for i in range(len(map_fnames)):
    if(('Ca' + elem_delim) in map_fnames[i]):
        Ca_map = np.asarray(PIL.Image.open(map_fnames[i])).copy().astype('int')
    if(('Si' + elem_delim) in map_fnames[i]):
        Si_map = np.asarray(PIL.Image.open(map_fnames[i])).copy().astype('int')
    if(('Mg' + elem_delim) in map_fnames[i]):
        Mg_map = np.asarray(PIL.Image.open(map_fnames[i])).copy().astype('int')
    if(('S' + elem_delim) in map_fnames[i]):
        S_map = np.asarray(PIL.Image.open(map_fnames[i])).copy().astype('int')
    if(('Fe' + elem_delim) in map_fnames[i]):
        Fe_map = np.asarray(PIL.Image.open(map_fnames[i])).copy().astype('int')
    if(('Cr' + elem_delim) in map_fnames[i]):
        Cr_map = np.asarray(PIL.Image.open(map_fnames[i])).copy().astype('int')
    if(('Ni' + elem_delim) in map_fnames[i]):
        Ni_map = np.asarray(PIL.Image.open(map_fnames[i])).copy().astype('int')
    if(('Ti' + elem_delim) in map_fnames[i]):
        Ti_map = np.asarray(PIL.Image.open(map_fnames[i])).copy().astype('int')
    print(map_fnames[i])

print('Img Shape: ',Ca_map.shape)

ydim = Ca_map.shape[0]
xdim = Ca_map.shape[1]

sil_map = np.zeros(Ca_map.shape)

sil_map[:, :,0] = Ca_map[:, :,0]      #Red
sil_map[:, :,1] = Si_map[:, :,0]     #Green
sil_map[:, :,2] = Mg_map[:, :,0]     #Blue

del Ca_map
del Si_map
del Mg_map

gc.collect()

opq_map = np.zeros(S_map.shape)

```

```

opq_map[:,:,0] = S_map[:,:,0] #Red
opq_map[:,:,1] = Fe_map[:,:,0] + Ni_map[:,:,0] #Green
opq_map[:,:,2] = Cr_map[:,:,0] #Blue

#del S_map
del Fe_map
del Cr_map

gc.collect()

print('converted colors... ')

# Mask and paint the Cr chromite pixels to pink
opq_map[np.where(opq_map[:,:,2] > 80)] = Cr_color

print('Cr converted')

# Get rid of silicate Fe signature, keep for troilite
#opq_map[np.where(opq_map[:,:1] < 130 ), 1] = 0

opq_map = np.where(opq_map[:,:1] < 100, 0, opq_map[:,:1])

print('Fe trimmed')

# Convert metal to grey (over-threshold)
opq_map[np.where(opq_map[:,:1] >= 100)] = Fe_color

print('Metal converted')

opq_map[np.where(S_map[:,:,0] < 200)] = Tr_color

'''

for i in range(opq_map.shape[0]):
    print(i, ' ',opq_map.shape[0])

    for j in range(opq_map.shape[1]):

        # This is iron rich-ish silicates, get rid of iron color
        if(opq_map[i, j, 1] < Fe_thresh and S_map[i, j, 0] < 20 ):

```

```

        opq_map[i,j,1] = 0
        # This is metal, convert to grey
        if(opq_map[i, j, 1] >= Fe_thresh ):           # and np.max(opq_map[i,j])
== opq_map[i,j,1]
            opq_map[i, j] = Fe_color
            # Do nothing for Troilite

'''

print('saving silicate and opaque maps...')

    # Save the Opaque and Silicate maps
    SIL_MAP = PIL.Image.fromarray(sil_map.astype('uint8'))
    OPQ_MAP = PIL.Image.fromarray(opq_map.astype('uint8'))

    SIL_MAP.save(os.path.join(save_dir, 'Silicate-Ca_Si_Mg_map_RGB.png'))
    OPQ_MAP.save(os.path.join(save_dir, 'Opaque-S_Fe_Cr_map_RGB.png'))

    # Sum the two maps
    sum_map = sil_map + opq_map

    # Sanitize any out of bounds pixel values, before transform into int
    sum_map = np.where(sum_map > 255, 255, sum_map)

    # Save sum_map
    SUM_MAP = PIL.Image.fromarray(sum_map.astype('uint8'))

    SUM_MAP.save(os.path.join(save_dir, 'sum_map.png'))

return

```

```
#####
```

```
#####  
#####  
#####  
#####  
#####
```

```
def predict_on_map(img_path, model_path, results_map=False):
```

```
    '''
```

```
    '''
```

```
    print('### Predicting on Mineral Map ###')
```

```
        # Load the model and img
```

```
    model = keras.models.load_model(model_path)
```

```
    mapp = np.asarray(PIL.Image.open(img_path))
```

```
    categories = np.array([
```

```
        'Chromite',
```

```
        'Diopside',
```

```
        'Enstatite',
```

```
        'Fe metal',
```

```
        'Olivine',
```

```
        'Phosphates',
```

```
        'Plagioclase',
```

```
        'Troilite',
```

```
        'Other-Unk'])
```

```
        # g/mL
```

```
    densities = np.array([ 4.79,
```

```
                          3.4,
```

```
                          3.2,
```

```
                          8.0,
```

```
                          2.7,
```

```
                          3.14,
```

```
                          2.68,
```

```
                          4.61,
```

```
                          3.5 ])
```



```

    # Mask off black pixels (non-minerals)
sum_mapp = np.sum(mapp, axis=2)
coords = np.where(sum_mapp != 0)
mask = np.zeros((mapp.shape[0], mapp.shape[1]), dtype=bool)
mask[coords] = True          # Color pixels are now True in

    # Predict on non black pixels
pixels = mapp[mask]
preds = model.predict(pixels)

    # Clean up predictions
preds = np.where(preds<0.5, 0, 1)

    # Count up the minerals and calculate the volume percentage for each flavor
min_count = np.sum(preds, axis=0)
vol_percent = 100 * (min_count / float(len(preds)))
vol_percent = np.append(vol_percent, (100-np.sum(vol_percent)))

wt_percent = 100 * vol_percent * densities / np.sum(vol_percent * densities)

    # Save Data/results
sname = os.path.join(os.path.split(img_path)[0], 'mineral_abundances.csv')
df = pd.DataFrame({'Phase': categories, 'Vol_percent': vol_percent, 'Wt_percent':
wt_percent})
df.to_csv(sname, index=False)

print('Weight_percent:\n')
    # Print results just for funsies
for i in range(len(wt_percent)):
    print(wt_percent[i], '\t', categories[i])

    # RGB
    # Make results map overlay
    # this doesn't work yet

if(results_map==True):
    print('Making Results File')
    colors = np.array([ [230, 30,145],          #Cr          0

```

```

[139, 69, 19],      #Cpx      1
[ 0,200,200],      #Opx      2
[100,100,100],     #Metal    3
[ 0, 0,255],       #Olivine  4
[255, 0, 0],       #Phos     5
[ 0,255, 0],       #Plag     6
[255,165, 0] ]) #Sulf      7
#unk
8
#blank
9

```

```

unk_color = (255,255,255) #unk

result_map = np.copy(mapp)
key_map = np.ones(result_map.shape[:2]) * 9

# Loop through each instance of pixels predicted on
for i in range(len(coords[0])):
    # If result is unknown
    if(np.sum(preds[i] == 0 ):
        color = unk_color
    # Otherwise
    else:
        #color = colors[np.where(preds[i]==1)]
        color = np.dot(preds[i], colors)

    result_map[coords[0][i], coords[1][i]] = color

# Save
result_map = result_map.astype('uint8')

sname = os.path.join(os.path.split(img_path)[0], 'results_map.png')
MAPP = PIL.Image.fromarray(result_map)
MAPP.save(sname)

# Down sample the results map

```

```
kernel = (3,3) # y,x
stride = 1
```

```
'''
for i in range(len(colors)):
    this_color_locs = np.where(preds[:,i]==1)

    color_coords = yxs[this_color_locs]

    for j in range(len(color_coords)):
        coord = color_coords[j]
        result_map[coord[0], coord[1,:]] = colors[i]

unk_locs = np.where(np.sum(preds, axis=1)==0)
color_coords = yxs[unk_locs]

for i in range(len(color_coords)):
    coord = color_coords[i]
    result_map[coord[0], coord[1,:]] = unk_color
'''
```

```
return
```

```
#####
#####
#####
#####
#####
```

```
#####
```

```
def prep_train_data(map_fname, csv_dir):
```

```
    '''
```

list_files will sort csvs alphabetically, so they must be named (see below) to keep consistent vector values

```
    Chromite      0
    Diopside      1
    Enstatite     2
    Fe_metal      3
    Olivine       4
    Phosphate     5
    Plag          6
    Troilite7
```

```
    other (no vector)
```

```
    '''
```

```
        # Open map image and csv list of pixel locations
```

```
        mapp = np.asarray(PIL.Image.open(map_fname))
```

```
        csvs = list_files(csv_dir, '.csv', 'all')
```

```
        # Prep output dataframe, which will contain actual pixel values
```

```
        cols = ['Red','Green','Blue','Label','Vector']
```

```
        final_df= pd.DataFrame(columns=cols)
```

```
        # Loop each csv (mineral type)
```

```
        for i in range(len(csvs)):
```

```
            current_df = pd.read_csv(csvs[i])
```

```
            name = os.path.split(csvs[i])[1].split('_pixels')[0]
```

```
                # Loop each pixel(s) listed in the csv
```

```
                for j in range(len(current_df)):
```

```
                    # Set X and Y bounds
```

```

x1 = current_df['BX' ].iloc[j]
x2 = x1 + current_df['Width' ].iloc[j]
y1 = current_df['BY' ].iloc[j]
y2 = y1 + current_df['Height' ].iloc[j]

pixels = mapp[y1:y2, x1:x2][0]

for k in range(len(pixels)):
    pix = pixels[k]
    final_df = final_df.append({'Red': pix[0],
                                'Green':
pix[1],
                                'Blue': pix[2],
                                'Label': name,
                                'Vector': i},

ignore_index=True)

print(final_df)
exit()

# Save compiled training data
sname = os.path.join(csv_dir, 'all_training_data.csv')
final_df.to_csv(sname, index=False)

'''
# Old version

# Loop each csv (mineral type)
for i in range(len(csvs)):
    current_df = pd.read_csv(csvs[i])
    name = os.path.split(csvs[i])[1].split('_pixels')[0]

    # Loop each pixel(s) listed in the csv
    for j in range(len(current_df)):
        pix = mapp[current_df[['BY']].iloc[j], current_df[['BX']].iloc[j]][0]
        final_df = final_df.append({'Red': pix[0],
                                    'Green': pix[1],
                                    'Blue': pix[2],

```

```
'Label': name,  
'Vector': i},  
ignore_index=True)
```

```
'''
```

```
return
```

```
#####  
#####  
#####  
#####  
#####  
#####
```

```
def train_mineral_network(fname):
```

```
'''
```

```
9 Categories
```

Chromite	0
Ca Px	1
Px	2
metal	3
olivine	4
phos	5
plag	6
sulf	7

```
other
```

```
'''
```

```
print('### Training neural network ###')
```

```
categories = np.array([  
    'Chromite',
```

```

        'Ca Px',
        'Px',
        'Fe_metal',
        'Olivine',
        'Phos',
        'Plag',
        'Sulf'])

    # Build a model
model = Sequential()

model.add(Dense(10, input_dim=3, activation='relu'))
model.add(Dense(10, activation='relu'))
model.add(Dense(10, activation='relu'))
model.add(Dense(10, activation='relu'))
model.add(Dense(10, activation='relu'))
model.add(Dense(8, activation='softmax'))

model.compile( loss = 'categorical_crossentropy',
               optimizer = 'adam',
               metrics = ['accuracy'] )

df = pd.read_csv(fname)

    # Get training data
x = df[['Red', 'Green', 'Blue']]

y = to_categorical(df['Vector'])

    # Fit model to data
model.fit(x, y, epochs=100, shuffle=True, steps_per_epoch=8)

    # Save trained model
model.save('OC_mineral_model.h5')

return

```


#####

#####

Appendix C (ISRU Code)

```
# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.
"""
import os
import time
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

def list_files(dirr):
    out = os.listdir(dirr)
    files = []
    for i in range(len(out)):
        files.append(os.path.join(dirr, out[i]))

    return files

'''


$$C_p^\circ = A + B*t + C*t^2 + D*t^3 + E/t^2$$


$$H^\circ - H^\circ_{298.15} = A*t + B*t^2/2 + C*t^3/3 + D*t^4/4 - E/t + F - H$$


$$S^\circ = A*\ln(t) + B*t + C*t^2/2 + D*t^3/3 - E/(2*t^2) + G$$

'''

def C(c, t):
    return c[0] + c[1]*t + c[2]*t**2. + c[3]*t**3 + c[4]/(t**2)

def H(c, t):
    return c[0]*t + c[1]*(t**2)/2. + c[2]*(t**3)/3. + c[3]*(t**4)/4. - c[4]/t + c[5]

def S(c, t):
    return c[0]*np.log(t) + c[1]*t + c[2]*(t**2)/2. + c[3]*(t**3)/3. - c[4]/(2*t**2) + c[6]
```

```

''' Calculate H, S and C at each T avail '''
'''-----'''

# Get filenames
dirr = './Thermo_csvs'
fnames = list_files(dirr)
dictt = []
final_dfs = []

# Loop through each filename
for j in range(len(fnames)):
    # Read this csv
    df = pd.read_csv(fnames[j])
    name = os.path.split(fnames[j])[1].replace('.csv', '')

    # Account for multiple coeff sets
    cols = df.columns
    pieces = []
    for i in range(len(cols)-1):
        # Determine temp range
        strr = cols[i+1].split(' - ')
        t = np.arange(float(strr[0]), float(strr[1]), 1)/1000.
        c = np.asarray(df[cols[i+1]])

        temp_df = pd.DataFrame({'T': t*1000,
                                'H': H(c,t),
                                'S': S(c,t),
                                'C': C(c,t)})
        pieces.append(temp_df)

    # Concat multiple coeff sets (temp ranges)
    if(len(pieces) > 1):
        final_df = pieces[0]
        for i in range(len(pieces) - 1):
            final_df = pd.concat([final_df, pieces[i+1]], ignore_index=True)
    else:
        final_df = pieces[0]

    dictt.append(name)
    final_dfs.append(final_df)

```

l = 2000

```
dictt.append('Fe2Si2O6')
final_dfs.append(pd.DataFrame({'T': np.arange(300, 2300),
                              'H': [-1437 for i in range(l)],
                              'S': [249 for i in range(l)]}))
```

```
dictt.append('Mg2Si2O6')
final_dfs.append(pd.DataFrame({'T': np.arange(300, 2300),
                              'H': [-3036 for i in range(l)],
                              'S': [259 for i in range(l)]}))
```

```
dictt.append('Fe2SiO4')
final_dfs.append(pd.DataFrame({'T': np.arange(300, 2300),
                              'H': [-1478 for i in range(l)],
                              'S': [151 for i in range(l)]}))
```

```
dictt.append('CaAl2Si2O8')
final_dfs.append(pd.DataFrame({'T': np.arange(300, 2300),
                              'H': [99 for i in range(l)],
                              'S': [3.2 for i in range(l)]}))
```

```
dictt.append('FeTiO3')
final_dfs.append(pd.DataFrame({'T': np.arange(300, 2300),
                              'H': [-1200 for i in range(l)],
                              'S': [182 for i in range(l)]}))
```

```
dictt.append('CaSO4') #
final_dfs.append(pd.DataFrame({'T': np.arange(300, 2300),
                              'H': [-1433 for i in range(l)],
                              'S': [106 for i in range(l)]}))
```

```
dictt.append('Al2SO43')
final_dfs.append(pd.DataFrame({'T': np.arange(300, 2300),
                              'H': [-3430 for i in range(l)],
                              'S': [259 for i in range(l)]}))
```

```
dictt.append('Ca2Si2O6')
final_dfs.append(pd.DataFrame({'T': np.arange(300, 2300),
                              'H': [-1630 for i in range(l)],
                              'S': [81 for i in range(l)]}))
```

```
dfs = final_dfs
```

```
for i in range(len(dfs)):
    dfs[i]['Name'] = dictt[i]
```

```
"""-----"""
```

```
""" Calculate del_G for a given set of reacts/prods """
"""-----"""
```

```
def G(react, prod, rmols, pmols, name):
```

```
    """This function takes in lists of dataframes, which contain the H, S and C values
    for each molecule/element.
```

```
    Inputs:
```

```
    react [list] each list-element is a df for a reactant
    prod [list] each list-element is a df for a product
    rmols [list] number of mols in balanced equation, in order with 'react'
    pmols [list] number of mols in balanced equation, in order with 'prod'
```

```
    Ouputs:
```

```
    del_H [array] Change in enthalpy for reaction [kJ/mol*K]
    del_S [array] Change in entropy for reaction [J/mol*K]
```

```
    """
```

```
    plot_dir = 'GK_plots'
```

```
    R = 8.3144626 # [J/molK]
```

```
    condensed = ['Al',
```

```
                'Al2O3',
```

```
                'Ca',
```

```
                'CaO',
```

```
                'Mg',
```

```
                'MgO',
```

```
                'FeO',
```

```
                'Fe',
```

```
                'Fe2O3',
```

```
                'Ti',
```

```
                'TiO2',
```

```
                'Mg2Si2O6',
```

```
                'Mg2SiO4',
```

```
                'Fe2Si2O6',
```

```
                'Fe2SiO4',
```

```
                'FeSO4',
```

```

'MgSO4',
'CaSO4',
'Al2SO43',
'CaAl2Si2O',
                                'Ca2Si2O6'
]

    # Determine the weakest link for temps reported
all_chems = prod + react
Tmaxs, Tmin = [], []
for i in range(len(all_chems)):
    Tmaxs.append(np.max(all_chems[i]['T']))
    Tmin.append(np.min(all_chems[i]['T']))
Tmaxs = np.asarray(Tmaxs)
Tmin = np.asarray(Tmin)
Tmax = np.min(Tmaxs)
Tmin = np.max(Tmin)
T = np.arange(Tmin, Tmax+1, 1) # Common temperature array for all prod/react

    # Prepare arrays for calculation
Hprods = np.zeros(int(Tmax-Tmin+1))
Hreacts = np.zeros(int(Tmax-Tmin+1))
Sprods = np.zeros(int(Tmax-Tmin+1))
Sreacts = np.zeros(int(Tmax-Tmin+1))
Cprods = np.zeros(int(Tmax-Tmin+1))
Creacts = np.zeros(int(Tmax-Tmin+1))
Gprods = np.zeros((3, int(Tmax-Tmin+1))) # These are pressure corrections for delG,
Greacts = np.zeros((3, int(Tmax-Tmin+1))) # to be added later

Gp = np.zeros((3, len(T)))
p = (10**(-2), # relative pressures in [bar]
     10**(-5),
     10**(-8))

''' Plot Gibbs of each react and prod '''
'''-----'''

markers = ['r-', 'g--', 'b-.', 'y:', 'k', 'k', 'k', 'k', 'k' ]

fig, axs = plt.subplots(1,2)

fig.suptitle(('G [kJ/mol]: ' + name))
fig.set_size_inches(10, 5)
axs[0].set_xlabel('Temperature [K]')
axs[0].set_ylabel('G [kJ/mol]')
axs[0].axhline(0, color='black') # x = 0
axs[1].set_xlabel('Temperature [K]')
axs[1].set_ylabel('G [kJ/mol]')

```

```

    axs[1].axhline(0, color='black')

# Sum vals for prods
for i in range(len(prod)):

    # Find index for temp, for this df
    ind1 = prod[i].index[prod[i]['T'] == Tmin][0] # .tolist()
    ind2 = prod[i].index[prod[i]['T'] == Tmax][0] # .tolist()
    H = np.asarray(prod[i]['H'].loc[ind1:ind2])
    S = np.asarray(prod[i]['S'].loc[ind1:ind2])

# Find G correction for pressure
for j in range(len(p)):
    Gp[j] = (pmols[i] * R * 0.001 * T * np.log(p[j])) # [kJ/mol]
    if(prod[i]['Name'].iloc[0] in condensed):
        Gp[j] = 0

    Hprods += H * pmols[i]
    Sprods += S * pmols[i]
    Gprods += Gp

    G_this = (H - T*S) * pmols[i]
    axs[1].plot(T, G_this, markers[i], label=prod[i]['Name'].iloc[0])

Gp = np.zeros((3, len(T)))

# Sum vals for reacts
for i in range(len(react)):
    ind1 = react[i].index[react[i]['T'] == Tmin][0]
    ind2 = react[i].index[react[i]['T'] == Tmax][0]
    H = np.asarray(react[i]['H'].loc[ind1:ind2])
    S = np.asarray(react[i]['S'].loc[ind1:ind2])

    # Find G correction for pressu
    for j in range(len(p)):
        Gp[j] = (rmols[i] * R * 0.001 * T * np.log(p[j]))

    Hreacts += H * rmols[i]
    Sreacts += S * rmols[i]
    Greacts = Greacts + Gp

    G_this = (H - T*S) * rmols[i]
    axs[0].plot(T, G_this, markers[i], label=react[i]['Name'].iloc[0])

```

```

# Finish this plot

axs[0].title.set_text('Reactants')
axs[1].title.set_text('Products')
axs[0].legend()
axs[1].legend()

plt.tight_layout()
#plt.show()
plt.clf()

# Find del_H and des_S (final-initial)
del_H = Hprods - Hreacts
del_S = Sprods - Sreacts

del_G = del_H - (T.T*( 0.001*del_S)) # Divide by 1000 because S is reported in J/molK
# H is reported in kJ/mol
# ^G is now T corrected
# Now we will correct for pressure
P_del_G = np.asarray([del_G for i in range(4)])
del_Gp = Gprods - Greacts # Pressure correction component
P_del_G[1:, :] = P_del_G[1:, :] + del_Gp

del_G_str = ' del_G: \t' + str(np.round(del_G[0]))
print('\n', name.replace('_', '').replace('$', ''),
      '\n\t' + del_G_str)

# Find eq point
if(np.sum(abs(del_G)) != abs(np.sum(del_G))):

    TT_ind = np.argmin(abs(del_G))
    T_Temp = np.round(T[TT_ind])
    T_str = ' Eq T [K]: \t' + str(T_Temp)

    print('\t' + T_str)
    #plt.text(T[TT_ind], del_G[TT_ind], str(T_Temp))

''' Plot Gibbs and K_eq '''

```

```

"-----"

fig, axs = plt.subplots(1,2)

fig.suptitle(name)
fig.set_size_inches(10, 5)
axs[0].title.set_text('Gibbs Free Energy')
axs[0].set_xlabel('Temperature [K]')
axs[0].set_ylabel('G [kJ/mol]')
axs[0].axhline(0,color='black') # x = 0
#plt.text(T[0], del_G[0], np.round(del_G[0]))

P_str = ['1 bar',
 '$10^{-2}$ bar',
 '$10^{-5}$ bar',
 '$10^{-8}$ bar']
for i in range(len(P_del_G)): #len(P_del_G)
    axs[0].plot(T, P_del_G[i], markers[i], label=P_str[i])

axs[0].legend()

K_eq = np.exp((-1* P_del_G ) / (R * 0.001 * T ))
#Q = np.exp((P_del_G - P_del_G[0]) / (R * 0.001 * T ))
K_eq = np.where(K_eq < 10**(-200), 10**(-200), K_eq)
#Q = np.where(Q < 10**(-200), 10**(-200), Q)

print('\t K_eq: \t', np.median(K_eq))
axs[1].title.set_text('Equilibrium constant')
axs[1].plot(T, 100 * np.ones(len(T)), 'k--') # x = 0
axs[1].plot(T, 0.01* np.ones(len(T)), 'k--')
axs[1].plot(T, np.ones(len(T)), 'k-')
axs[1].set_ylabel('$K_{eq}$')
axs[1].set_xlabel("")

for i in range(len(K_eq)):
    axs[1].plot(T, K_eq[i], markers[i], label=P_str[i])

axs[1].set_yscale('log')
axs[1].legend()
axs[1].set_ylim()

plt.tight_layout()

sname = os.path.join(plot_dir, name.replace('$', '').replace('(', '').replace(')', '').replace('>',
").replace(';',").replace('/',") )

```



```

plt.savefig(sname)
plt.clf()

''' Use K_eq to find the concentrations (T) '''

'''[Mg, Ca]SO4 decomp '''

    # KK is K_eq at 1 bar
    KK = K_eq[0]
    A = np.zeros(len(KK))

    # Move through KK as a func of T
    for i in range(len(KK)):
        # Solve for A at every K_eq(T)

        ''' Al Sulf
        kterm = 55 - ((3**9)/8.* KK[i]**2)
        p = np.poly1d([-1, 11, -55, 145, -290, 442, -442, 290, -145, kterm, -11, 1])
        '''

        ''' FeSO4
        kterm = 35 - (16* KK[i]**2)
        p = np.poly1d([-1, 7, -21, kterm, -35, 21, -7, 1])
        '''

        ''' [Ca, Mg]SO4
        kterm = 10 - (2* KK[i]**2)
        p = np.poly1d([-1, 5, -10, kterm, -5, 1])
        '''

        kterm = 5 - KK[i]/4.
        p = np.poly1d([-1, kterm, -10, 10, -5, 1])

    roots = p.r

    for j in range(len(roots)):
        if(np.real(roots[j]) >= 0
           np.imag(roots[j]) == 0
           ):
            and
            # and np.real(roots[j]) <= 1

            A[i] = np.real(roots[j])

            if(A[i] > 0.99270 ):
                A[i] = 0.99270

```

```

B = 3 * A
C = 2 * (1 - A)
D = 3 * (1 - A)

```

```
plt.clf()
```

```
fig, axs = plt.subplots(2,1)
```

```
fig.suptitle('Carbothermal Reduction')
```

```

axs[0].plot(T, A, 'k-', label='$Fe_2O_3$')
axs[0].plot(T, B, 'k--', label='$CO$')
axs[0].plot(T, C, 'g-.', label='$Fe$')
axs[0].plot(T, D, 'b:', label='$CO_2$')

```

```

# KK is K_eq at e-4 bar
KK = K_eq[3]
A = np.zeros(len(KK))

# Move through KK as a func of T
for i in range(len(KK)):
    # Solve for A at every K_eq(T)
    """ Al Sulf
    kterm = 55 - ((3**9)/8.* KK[i]**2)
    p = np.poly1d([-1, 11, -55, 145, -290, 442, -442, 290, -145, kterm, -11, 1])
    """
    """ FeSO4
    kterm = 35 - (16* KK[i]**2)
    p = np.poly1d([-1, 7, -21, kterm, -35, 21, -7, 1])
    """
    """ [Ca, Mg]SO4
    kterm = 10 - (2* KK[i]**2)
    p = np.poly1d([-1, 5, -10, kterm, -5, 1])
    """
    """ Fe2O3 Carbothermal reduc """
    kterm = 5 - KK[i]/4.
    p = np.poly1d([-1, kterm, -10, 10, -5, 1])

    roots = p.r

    for j in range(len(roots)):
        if(np.real(roots[j]) >= 0 and
           np.imag(roots[j]) == 0 # and np.real(roots[j]) <= 1

```

```

):

    if(A[i] != 0):
        print('Double Root!', T[i])

    A[i] = np.real(roots[j])

    if(A[i] > 0.99270 ):
        A[i] = 0.99270

B = 3 * A
C = 2 * (1 - A)
D = 3 * (1 - A)

    axs[1].set_xlabel('Temperature [K]')

    axs[1].plot(T, A, 'k-', label='$Fe_2O_3$')
    axs[1].plot(T, B, 'k--', label='$CO$')
    axs[1].plot(T, C, 'g-.', label='$Fe$')
    axs[1].plot(T, D, 'b:', label='$CO_2$')
    #axs[1].legend()

    fig.add_subplot(111, frameon=False)
    # hide tick and tick label of the big axis
    plt.tick_params(labelcolor='none', which='both', top=False, bottom=False, left=False, right=False)

    plt.ylabel("Relative Abundance")
    axs[1].grid()
    axs[0].grid()

    axs[0].set_xlim((300, 1400))
    axs[1].set_xlim((300, 1400))

    plt.tight_layout()
    #plt.show()

    return del_G

#print(dfs)

```

```
#####
```

```
# Layout Each Reaction
```

```
#####
```

```
plt.clf()
plt.text(1, 1, 'Start New Calc')
#plt.show()
plt.clf()
```

```
"""
```

```
del_G = G([dfs[dictt.index('H2')], dfs[dictt.index('FeSO4')]],
          [dfs[dictt.index('H2O')], dfs[dictt.index('SO2')], dfs[dictt.index('Fe2O3')] ],
          [1, 2],
          [1, 2, 1],
          '$H_2 (g) + 2 FeSO_4 (s) -> H_2O (l) + SO_2 (g) + Fe_2O_3 (s)$')
```

```
del_G = G([dfs[dictt.index('H2')], dfs[dictt.index('CaSO4')]],
          [dfs[dictt.index('H2O')], dfs[dictt.index('SO2')], dfs[dictt.index('CaO')] ],
          [1, 1],
          [1, 1, 1],
          '$H_2 (g) + CaSO_4 (s) -> H_2O (l) + SO_2 (g) + CaO (s)$')
```

```
del_G = G([dfs[dictt.index('H2')], dfs[dictt.index('Fe2O3')]],
          [dfs[dictt.index('H2O')], dfs[dictt.index('Fe')] ],
          [3, 1],
          [2, 3],
          '$H_2 (g) + 2 Fe_2O_3 (s) -> 2 Fe (s) + 3 H_2O (g-l)$')
```

```
"""
```

```
'''
```

```
# FeS + H2SO4 -> FeSO4 + H2S
```

```
del_G = G([dfs[dictt.index('FeS')], dfs[dictt.index('H2SO4')]],
          [dfs[dictt.index('FeSO4')], dfs[dictt.index('H2S')] ],
          [1, 1],
```

```
[1, 1],
'$FeS (s)+ H_2SO_4 -> FeSO_4 (s) + H_2S (g)$')
```

```
# H2S + 3/2*O2 -> H2O + SO2
```

```
del_G = G([dfs[dictt.index('H2S')], dfs[dictt.index('O2')]],
 [dfs[dictt.index('H2O')], dfs[dictt.index('SO2')] ],
 [1, 1.5],
 [1, 1],
 '$H_2S (g) + 3/2 O_2 (g) -> H_2O (l) + SO_2 (g)$')
```

```
#####
```

```
# FeS -> Fe + 0.5* S2
```

```
del_G = G([dfs[dictt.index('FeS')],
 [dfs[dictt.index('Fe' )], dfs[dictt.index('S2')] ]],
 [1],
 [1, 0.5],
 '$FeS (s) -> Fe (s) + 1/2 S_2 (g)$')
```

```
# FeS + O2 -> Fe + SO2
```

```
del_G = G([dfs[dictt.index('FeS')], dfs[dictt.index('O2')]],
 [dfs[dictt.index('Fe' )], dfs[dictt.index('SO2')] ],
 [1, 1],
 [1, 1],
 '$FeS (s)+ O_ (g) -> Fe (s) + SO_2 (g)$')
```

```
#2 FeS + 7/2 O2 -> Fe2O3 + 2 SO2
```

```
del_G = G([dfs[dictt.index('FeS')], dfs[dictt.index('O2')]],
 [dfs[dictt.index('Fe2O3' )], dfs[dictt.index('SO2')] ],
 [2, 3.5],
 [1, 2],
 '$2 FeS (s) + 7/2 O_2-> Fe_2O_3 (s) +2 SO_2 (g)$')
```

```
#####
```

0.5*S2 + O2 -> SO2

```
del_G = G([dfs[dictt.index('S2')], dfs[dictt.index('O2')]],  
          [dfs[dictt.index('SO2' )]] ,  
          [0.5, 1],  
          [1],  
          '$1/2 S_2 (g)+ O_2 (g) -> SO_2 (g)$')
```

SO2 +0.5* O2 -> SO3

```
del_G = G([dfs[dictt.index('SO2')], dfs[dictt.index('O2')]],  
          [dfs[dictt.index('SO3' )]] ,  
          [1, 0.5],  
          [1],  
          '$SO_2 (g)+ 1/2 O_2(g) -> SO_3 (g)$')
```

SO3 +H2O -> H2SO4

```
del_G = G([dfs[dictt.index('SO3')], dfs[dictt.index('H2O')]],  
          [dfs[dictt.index('H2SO4' )]] ,  
          [1, 1],  
          [1],  
          '$SO_3(g) + H_2O (l) -> H_2SO_4 (aq)$')
```

#####

Mg2SiO4 + 2*H2SO4 -> 2*H2O + 2*MgSO4 + SiO2

```
del_G = G([dfs[dictt.index('Mg2SiO4')], dfs[dictt.index('H2SO4')]],  
          [dfs[dictt.index('H2O' )], dfs[dictt.index('MgSO4')], dfs[dictt.index('SiO2')]],  
          [1, 2],  
          [2, 2, 1],  
          '$Mg_2SiO_4 (s)+ 2 H_2SO_4 (aq) -> 2 H_2O(l) + MgSO_4(s) + SiO_2$')
```

Mg2Si2O6 + 2*H2SO4 -> 2*H2O + 2*MgSO4 + 2*SiO2

```
del_G = G([dfs[dictt.index('Mg2Si2O6')], dfs[dictt.index('H2SO4')]],  
          [dfs[dictt.index('H2O' )], dfs[dictt.index('MgSO4')], dfs[dictt.index('SiO2')]],  
          [1, 2],  
          [2, 2, 2],  
          '$Mg_2Si_2O_6(s) + *H_2SO_4 (aq) -> 2 H_2O + 2 MgSO_4(s) + 2 SiO_2 (s)$')
```

Ca2Si2O6 + 2*H2SO4 -> 2*H2O + 2*CaSO4 + 2*SiO2

```
del_G = G([dfs[dictt.index('Ca2Si2O6')], dfs[dictt.index('H2SO4')]],
          [dfs[dictt.index('H2O')], dfs[dictt.index('CaSO4')], dfs[dictt.index('SiO2')]],
          [1, 2],
          [2, 2, 2],
          '$Ca_2Si_2O_6(s) + *H_2SO_4 (aq) -> 2 H_2O + 2 CaSO_4(s) + 2 SiO_2 (s)$')
```

#####

```
del_G = G([dfs[dictt.index('Mg')], dfs[dictt.index('O2')]],
          [dfs[dictt.index('MgO')]],
          [1, 0.5],
          [1],
          '$Mg + 1/2 O_2 -> MgO $')
```

Fe2SiO4 + 2*H2SO4 -> 2*H2O + 2*FeSO4 + SiO2

```
del_G = G([dfs[dictt.index('Fe2SiO4')], dfs[dictt.index('H2SO4')]],
          [dfs[dictt.index('H2O')], dfs[dictt.index('FeSO4')], dfs[dictt.index('SiO2')]],
          [1, 2],
          [2, 2, 1],
          '$Fe_2SiO_4 (s) + 2 H_2SO_4 (aq) -> 2 H_2O + 2 FeSO_4(s) + SiO_2 (s)$')
```

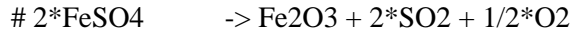
Fe2Si2O6 + 2*H2SO4 -> 2*H2O + 2*FeSO4 + 2*SiO2

```
del_G = G([dfs[dictt.index('Fe2Si2O6')], dfs[dictt.index('H2SO4')]],
          [dfs[dictt.index('H2O')], dfs[dictt.index('FeSO4')], dfs[dictt.index('SiO2')]],
          [1, 2],
          [2, 2, 2],
          '$Fe_2Si_2O_6(s) + 2H_2SO_4 -> 2 H_2O + 2 FeSO_4(s) + 2 SiO_2 (s)$')
```

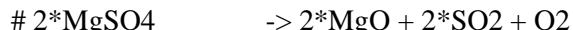
CaAl2Si2O8 + 4*H2SO4 -> CaSO4 + Al2(SO4)3 + 4*H2O + 2*SiO2

```
del_G = G([dfs[dictt.index('CaAl2Si2O8')], dfs[dictt.index('H2SO4')]],
          [dfs[dictt.index('CaSO4')], dfs[dictt.index('Al2SO43')], dfs[dictt.index('H2O')],
          dfs[dictt.index('SiO2')]],
          [1, 4],
          [1, 1, 4, 2],
          '$CaAl_2Si_2O_8(s) + 4 H_2SO_4 (aq) -> CaSO_4(s) + Al_2(SO_4)_3(s) + 4 H_2O (l) + 2 SiO_2 (s)$')
```

#####



```
del_G = G([dfs[dictt.index('FeSO4')]],  
          [dfs[dictt.index('Fe2O3')], dfs[dictt.index('SO2')], dfs[dictt.index('O2')]],  
          [2],  
          [1, 2, 0.5],  
          '$2 FeSO_4 -> Fe_2O_3 + 2 SO_2 + 1/2 O_2$')
```



```
del_G = G([dfs[dictt.index('MgSO4')]],  
          [dfs[dictt.index('MgO')], dfs[dictt.index('SO2')], dfs[dictt.index('O2')]],  
          [2],  
          [2, 2, 1],  
          '$2 MgSO_4 -> 2 MgO + 2 SO_2 + O_2$')
```



```
del_G = G([dfs[dictt.index('CaSO4')]],  
          [dfs[dictt.index('CaO')], dfs[dictt.index('SO2')], dfs[dictt.index('O2')]],  
          [2],  
          [2, 2, 1],  
          '$CaSO_4 -> CaO + SO_2 + 1/2 O_2$')
```



```
del_G = G([dfs[dictt.index('Al2SO43')]],  
          [dfs[dictt.index('Al2O3')], dfs[dictt.index('SO2')], dfs[dictt.index('O2')]],  
          [1],  
          [1, 3, 1.5],  
          '$Al_2(SO_4)_3 -> Al_2O_3 + 3 SO_2 + 3/2 O_2$')
```

#####




```

del_G = G([dfs[dictt.index('Fe2O3')], dfs[dictt.index('CO')]],
          [dfs[dictt.index('CO2' )], dfs[dictt.index('Fe')]],
          [1, 3],
          [3, 2],
          '$Fe_2O_3 + 3 CO -> 3 CO_2 + 2 Fe$')

```

```
# Fe2O3 + 3*CO -> 3*CO2 + 2*Fe
```

```

del_G = G([dfs[dictt.index('Fe2O3')], dfs[dictt.index('H2')]],
          [dfs[dictt.index('H2O' )], dfs[dictt.index('Fe')]],
          [1, 3],
          [3, 2],
          '$Fe_2O_3 + 3 H_2 -> 3 H_2O + 2 Fe$')

```

```
# 3*CO2 -> 3*CO + 3/2*O2
```

```

del_G = G([dfs[dictt.index('CO2')]],
          [dfs[dictt.index('CO' )], dfs[dictt.index('O2')]],
          [3],
          [3, 1.5],
          '$ 3 CO_2 -> 3 CO + 3/2 O_2$')

```

```
#####
```

```
# S -> 1/2 S2
```

```

del_G = G([dfs[dictt.index('S')]],
          [dfs[dictt.index('S2' )]],
          [1],
          [0.5],
          '$S -> +1/2 S_2$')

```

```
# MgO + CO -> Mg + CO2
```

```

del_G = G([dfs[dictt.index('MgO')], dfs[dictt.index('CO' )]],
          [dfs[dictt.index('Mg' )], dfs[dictt.index('CO2')]],
          [1, 1],
          [1, 1],
          'MgO + CO -> Mg + CO2')

```

```
# MgO + H2 -> Mg + H2O
```

```
del_G = G([dfs[dictt.index('MgO')], dfs[dictt.index('H2' )]],  
          [dfs[dictt.index('Mg' )], dfs[dictt.index('H2O')]],  
          [1, 1],  
          [1, 1],  
          'MgO + H2 -> Mg + H2O')
```

```
# CaO + CO -> Ca + CO2
```

```
del_G = G([dfs[dictt.index('MgO')], dfs[dictt.index('CO' )]],  
          [dfs[dictt.index('Mg' )], dfs[dictt.index('CO2')]],  
          [1, 1],  
          [1, 1],  
          'CaO + CO -> Ca + CO2')
```

```
# Al2O3 + 3*CO -> 2*Al + 3*CO2
```

```
del_G = G([dfs[dictt.index('Al2O3')], dfs[dictt.index('CO' )]],  
          [dfs[dictt.index('Al' )], dfs[dictt.index('CO2')]],  
          [1, 3],  
          [2, 3],  
          'Al2O3 + 3*CO -> 2*Al + 3*CO2')
```

```
del_G = G([dfs[dictt.index('TiO2')], dfs[dictt.index('CO' )]],  
          [dfs[dictt.index('Ti' )], dfs[dictt.index('CO2')]],  
          [1, 2],  
          [1, 2],  
          'TiO2 + 2*CO -> Ti + 2*CO2')
```

```
'''
```