An Approach to Lunar Regolith Particle Detection and Classification using Deep Learning

Hira Nadeem, Western University

Supervisor: McIsaac, Kenneth, The University of Western Ontario

A thesis submitted in partial fulfillment of the requirements for the Master of Engineering Science degree in Electrical and Computer Engineering

© Hira Nadeem 2023

Follow this and additional works at: https://ir.lib.uwo.ca/etd

Part of the Artificial Intelligence and Robotics Commons

Recommended Citation


This Dissertation/Thesis is brought to you for free and open access by Scholarship@Western. It has been accepted for inclusion in Electronic Thesis and Dissertation Repository by an authorized administrator of Scholarship@Western. For more information, please contact wlswadmin@uwo.ca.
Abstract

Lunar regolith, unconsolidated rock on the lunar surface, is made up of various particles. Understanding the quantities and locations of these particles on the lunar surface is of particular interest to planetary scientists for mission planning and science objectives. There is a limited supply of lunar regolith samples available on Earth for planetary scientists to characterize. Lunar rover missions over the next decade are expected to provide high-resolution images of the lunar surface. Deep learning can be leveraged to analyze lunar regolith from image data. An object detection model using transfer learning was developed to identify and classify particles of in-situ lunar regolith. A custom dataset using micro-images of the lunar surface from the Apollo missions was labelled and processed. Pre-trained Faster R-CNN and Single Shot Detector with ResNet backbone architectures were tested. The results were promising for the application of deep learning using transfer learning on lunar regolith imagery.

Keywords: deep learning, transfer learning, object detection, image classification, lunar regolith, planetary science, convolutional neural network, Faster R-CNN, Single Shot Detector
Lay Summary

Humanity first set foot on the Moon in 1969 with the Apollo 11 mission. Few missions have explored the surface of the Moon as closely since the Apollo era. With the rise of lunar exploration through the Artemis era, missions have been planned over the next decade and are expected to provide new information about the Moon. Lunar rovers and other robotic systems equipped with specialized instruments will study the lunar surface composition, uncovering a deeper understanding of the formation of the Moon, Earth and the solar system. Missions will also analyze the surface for potential mineral resources that could be used for further exploration, and provide insight for the feasibility of a long-term human presence on the Moon. With this influx of data, there will be many opportunities to use machine learning to reduce the human resource required to process the data. Currently, planetary geologists look at lunar soil under a microscope to understand the composition and characteristics. However, as lunar robotic missions with microscopic imaging equipment begin to return data from the Moon, deep learning models can be used for this task. Using a convolutional neural network (CNN) based model, commonly used for image classification and object detection tasks in machine learning, this work outlines a process developed to demonstrate the capability of applying deep learning to planetary geology applications. Since deep learning requires large datasets of labelled images, a transfer learning approach, which leverages models that are already trained on large datasets for one task, was used to apply to a smaller image dataset of the lunar surface. Different models were tested to understand the best architectures to use, and the models were tuned to improve performance. This work showed that despite small amounts of data from the lunar surface, machine learning models can understand the important features of lunar soil and identify different particles, such as glass, melts, and rock and mineral conglomerates called breccias, with reasonable confidence.
Acknowledgements

I would first like to thank my supervisor, Dr. Kenneth McIsaac, for his unwavering support, encouragement, and guidance throughout my Master’s degree, and for providing the opportunity to perform this meaningful research. I would like to thank my lab mate Alexis Pascual for his guidance, knowledge, and constant pep talks throughout the process of this research. Thank you to my research group, to the Western CubeSat team, and to my colleagues in the Institute for Earth and Space Exploration for providing comfort, advice and friendship throughout my time at Western University. I truly felt at home, supported, and encouraged to learn.

I would like to thank the entire Mission Control Space Services team for welcoming me on the team. Specifically, I would like to thank my supervisor Dr. Melissa Battler for her guidance and expertise that helped shape this research, Cosette Gilmour for providing her time and knowledge to identify rocks on the Moon, and to Andrew MacDonald for his guidance and extensive knowledge in the field of machine learning.

And of course, I would like to thank my parents, Ejaz and Rukhsana Nadeem for always encouraging me to accomplish my goals, my sister Nadia Nadeem for encouragement and support, and especially, my younger sister Sobia Nadeem for editing this thesis, taking care of my mental health throughout writing, and sharing my love of space.
Contents

Abstract i

Lay Summary ii

Acknowledgements iii

List of Figures viii

List of Tables xii

List of Appendices xiii

Acronyms xiv

1 Introduction 1
  1.1 Motivation ........................................ 1
  1.2 Task and Thesis Contribution ......................... 3
  1.3 Thesis Outline ...................................... 4

2 Background and Literature Review 5
  2.1 The Moon ........................................... 5
    2.1.1 Lunar Regolith .................................. 6
           Breccia .......................................... 7
           Melt ............................................ 7
           Glass .......................................... 7
    2.1.2 Lunar Regolith Simulant ....................... 8
2.1.3 Angle of Repose .................................................. 10
2.1.4 Apollo Missions .................................................. 11
  Apollo Lunar Surface Close-up Cam (ALSCC) ..................... 11
2.1.5 Imagery of the Lunar Surface ................................. 13
2.2 Machine Learning .................................................. 15
  2.2.1 Learning Methods .............................................. 15
  2.2.2 Features ....................................................... 16
  2.2.3 Gradient Descent .............................................. 16
  2.2.4 Neural Network (NN) ......................................... 18
  Convolutional Neural Networks (CNN) ............................. 19
  Region-Based Convolutional Neural Networks (R-CNN) ........ 21
2.3 Deep Learning for Object Detection ............................ 22
  2.3.1 Transfer Learning .............................................. 23
  2.3.2 Object Detection Models .................................... 24
    Faster R-CNN .................................................... 24
    Single Shot Detector (SSD) .................................... 25
    ResNet ......................................................... 26
  2.3.3 Model Parameters ............................................ 26
    Epochs, Batch Size, and Steps ................................. 27
    Learning rate .................................................. 28
    Optimizers .................................................... 28
  2.3.4 Training, Testing and Validation ......................... 28
  2.3.5 Data Augmentation ........................................... 29
  2.3.6 Evaluation Metrics ......................................... 30
    Intersection Over Union (IOU) ................................ 31
    Average Precision (AP) and Average Recall (AR) .......... 32
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Captured by the Hasselblad 500EL 70 mm camera from the Apollo 11 command module, <em>Columbia</em>, as the Apollo Lunar Module Eagle ascends from the lunar surface to return to meet <em>Columbia</em> (AS11-44-6642) [38].</td>
<td>5</td>
</tr>
<tr>
<td>2.2</td>
<td>Examples of different rock fragments present in lunar regolith including glass, breccias, impact melts, and agglutinates [37].</td>
<td>8</td>
</tr>
<tr>
<td>2.3</td>
<td>Sample of OB-1 simulant captured by author.</td>
<td>9</td>
</tr>
<tr>
<td>2.4</td>
<td>Sample of JSC-1 simulant captured by author.</td>
<td>10</td>
</tr>
<tr>
<td>2.5</td>
<td>Demonstration of the angle of repose with a block on a slope. The ratio of the resisting force and the force of the block being pushed into the slope [36].</td>
<td>10</td>
</tr>
<tr>
<td>2.6</td>
<td>Approximate locations of the Apollo 11 (Tranquility Base), Apollo 12 (Oceanus Procellarum), and Apollo 14 (Fra Mauro).</td>
<td>13</td>
</tr>
<tr>
<td>2.7</td>
<td>Images of the Moon showing improvements over decades.</td>
<td>14</td>
</tr>
<tr>
<td>2.8</td>
<td>Images of glass sperules on the lunar surface captured by the Yutu-2 rover [53].</td>
<td>15</td>
</tr>
<tr>
<td>2.9</td>
<td>Basic visualization of gradient descent of a cost function. Arbitrarily starting at point A, the gradient descent algorithm’s goal is to step towards the local minima at point B [26].</td>
<td>17</td>
</tr>
<tr>
<td>2.10</td>
<td>Comparison of a big learning rate resulting in the algorithm overshooting the local minimum, and a small learning rate which converges to the local minimum slowly [10].</td>
<td>18</td>
</tr>
<tr>
<td>2.11</td>
<td>Diagram of simple neural network with an input layer, at least two hidden layers, and an output layer [9].</td>
<td>19</td>
</tr>
</tbody>
</table>
2.12 Example of CNN architecture consisting of input image, convolutional layers with ReLu activation, pooling layers, fully connected layer, and softmax activation function for the classification output [45].

2.13 Sigmoid activation vs. ReLu activation [44].

2.14 Overview of R-CNN network for object detection. Regions are proposed over an input image. Then, for each proposal, features are extracted using a CNN before outputting the classification for the region [15].

2.15 One-stage vs. two-stage detectors [4].

2.16 Diagram of how Faster R-CNN includes RPN and ROI pooling into the architecture [40].

2.17 Architecture of an SSD network [34].

2.18 Residual learning block representing the skip connection for identity mapping. The output from the previous layer is applied to the next layer [18].

2.19 Example of data augmentation on an example image from the MNIST handwritten digit dataset [12].

2.20 Confusion matrix representation

2.21 Example of IOU (Green = ground truth box, red = predicted box). [41].

2.22 Descriptions of metrics used in the COCO Evaluation Metrics [6].

3.1 In-focus region of an image of sample of OB-1

3.2 Description of procedure to shift image histogram to remove background, isolate glass particles, and generate a mask overlay.

3.3 Comparison of original, ground truth bounding boxes to predicted bounding boxes. Additional augmented images were tested to see how well the object detector would generalize.

3.4 Plots of loss and mAP of the Faster R-CNN ResNet50 model with the OB-1 dataset.

4.1 Example stereo-pair images from the Apollo 11 images (AS11-45-6709 A/B)
4.2 Sample images of glass, melt, and breccia particles that were labelled in the ALSCC custom dataset. ................................. 48

4.3 Example of object ground truth labels on the full-size images (AS11-45-6708A). Red = melt, Green = glass, Blue = breccia. ........................................ 50

5.1 Evaluation and training loss for all selected models for 25k steps. Blue = training loss, orange = validation loss ......................................................... 62

5.2 Model predicted bounding boxes for 'melt' using test images (cropped portion of AS11-45-6709B image). Green = glass, blue = breccia, and teal = melt. . . . . 64

5.3 Model predicted bounding boxes for 'melt' and 'glass' using test images (cropped portion of AS11-45-6701A image). Green = glass, blue = breccia, and teal = melt. ........................................................................ 65

5.4 Model predicted bounding boxes for 'melt' and 'breccia' using test images (cropped portion of AS11-45-6700A image). Green = glass, blue = breccia, and teal = melt. ........................................................................ 66

5.5 Plots of loss and mAP for Faster R-CNN ResNet101 and SSD ResNet50 for tuned learning rates of $4\times 10^{-3}$ and $4\times 10^{-4}$. ......................................................... 69

5.6 Plots of loss and mAP for Faster R-CNN ResNet101 and SSD ResNet50 after tuning the number of steps. ............................................................... 71

5.7 Object detection on test images after tuning learning rate for Faster R-CNN ResNet101 and SSD ResNet50 models. ......................................................... 74

5.8 Object detection on test images after tuning number of steps for Faster R-CNN ResNet101 and SSD ResNet50 models. ......................................................... 76

A.1 All mAP plots for Faster R-CNN ResNet50. ......................................................... 88

A.2 All mAP plots for Faster R-CNN ResNet101. ......................................................... 89

A.3 All mAP plots for Faster R-CNN ResNet152. ......................................................... 89

A.4 All mAP plots for SSD ResNet50. ................................................................. 90

A.5 All mAP plots for SSD ResNet101. ................................................................. 90
B.1 All AR plots for Faster R-CNN ResNet50. ........................................ 91
B.2 All AR plots for Faster R-CNN ResNet101. ................................. 92
B.3 All AR plots for Faster R-CNN ResNet152. ............................... 92
B.4 All AR plots for SSD ResNet50. ............................................. 93
B.5 All AR plots for SSD ResNet101. ............................................. 93
C.1 All loss plots for Faster R-CNN ResNet50. ................................. 94
C.2 All loss plots for Faster R-CNN ResNet101. ............................... 95
C.3 All loss plots for Faster R-CNN ResNet152. ............................... 95
C.4 All loss plots for SSD ResNet50. ............................................. 96
C.5 All loss plots for SSD ResNet101. ............................................. 96
List of Tables

2.1 Specifications of the ALSCC onboard the Apollo 11, 12, and 14 missions [38] . 12
3.1 Hyperparameters for Faster RCNN with Resnet50 V1 model with OB-1 dataset 42
4.1 Number of images for each mission . . . . . . . . . . . . . . . . . . . . . . . . . . 47
4.2 Number of objects per class for the ALSCC dataset . . . . . . . . . . . . . . . 50
4.3 Number of objects per class before and after augmentation . . . . . . . . . . . 53
4.4 Number of objects per scale in the training, validation and testing sets, and per
object class (after augmentation) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 54
4.5 Pre-trained models available on TF2 Model Zoo that were considered . . . . . 54
4.6 Model Parameters . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 55
5.1 Summary of Model Training Time . . . . . . . . . . . . . . . . . . . . . . . . . . 58
5.2 Summary of Models: mAP after 25k steps . . . . . . . . . . . . . . . . . . . . . 60
5.3 Summary of Model Performance: AR after 25k steps . . . . . . . . . . . . . . 61
5.4 Summary of Models: Loss after 25k steps. All losses are for validation unless
otherwise stated. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 63
5.5 Inference time for selected models . . . . . . . . . . . . . . . . . . . . . . . . . . 64
5.6 Summary of Faster R-CNN ResNet101 performance after tuning learning rate. . 70
5.7 Summary of SSD ResNet50 performance after tuning learning rate. . . . . . . . 70
5.8 Summary of SSD ResNet50 performance after tuning the number of steps. . . . 72
5.9 Summary of Faster R-CNN ResNet101 performance after tuning the number
of steps. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 73
List of Appendices

Appendix A Training and Validation mAP for Models (Section 5.1.1) . . . . . . . . . . 88
Appendix B Training and Validation AR for Models (Section 5.1.2) . . . . . . . . . . 91
Appendix C Training and Validation Loss for Models (Section 5.1.3) . . . . . . . . . . 94
Acronyms

ALSCC  Apollo Lunar Surface Close-Up Camera.

AP   Average Precision.

API  Application Program Interface.

AR   Average Recall.

BGD  Batch Gradient Descent.

CNN  Convolutional Neural Network.

COCO  Common Objects in Context.

FN   False Negative.

FP   False Positive.

HCI  Human-Computer Interaction.

HOG  Histogram of Oriented Gradients.

ILSVRC  ImageNet Large Scale Visual Recognition Challenge.

IOU  Intersection Over Union.

JSC  Johnson Space Center.

LRO  Lunar Reconnaissance Orbiter.

LSCC  Lunar Soil Characterization Consortium.
mAP  Mean Average Precision.

mAR  Mean Average Recall.

MBRSC  Mohammed bin Rashid Space Centre.

NASA  National Aeronautics and Space Administration.

NMS  Non-Maximum Suppression.

NN  Neural Network.


PCAM  Panoramic Camera.

R-CNN  Region-based Convolutional Neural Network.

ROI  Region of Interest.

SGD  Stochastic Gradient Descent.

SSD  Single Shot Detector.

SVM  Support Vector Machines.

TF  TensorFlow.

TN  True Negative.

TP  True Positive.

YOLO  You Only Look Once.
Chapter 1

Introduction

1.1 Motivation

The motivation for this thesis is to demonstrate an application of deep learning to support human-computer interaction (HCI) in lunar exploration missions. Specifically, this thesis outlines a procedure for the development of a dataset of micro-images containing in-situ lunar regolith, and the development of an object detection model using transfer learning to classify and identify distinct particle types present on the lunar surface. Currently, planetary geologists analyze lunar regolith from sample return missions, and observe individual grains and cross-sections under a microscope to begin characterizing particles. Further chemical analysis is also performed to understand the composition. Due to the limited supply of samples returned from missions like the Apollo missions in the 1960s and the Chang’E-5 mission in 2020, characterizing in-situ lunar regolith particles is difficult. Astronauts captured images with great detail during the Apollo missions with the handheld Apollo Lunar Surface Close-up Camera (ALSCC). More recently, the microscopic images of in-situ lunar regolith that were captured on film have been scanned to an online database. This has allowed scientists worldwide to be able to analyze the fine particles in-situ, and draw from the observations made by the astronauts who took them. With the increase in lunar exploration missions planned over the next decade, new images of the Moon will be obtained and greater insight into its composition, history, po-
potential uses of regolith, and feasibility as a base for further human space exploration will be explored. Building on the images captured by astronauts with the ALSCC, intelligent computer systems onboard lunar rovers could analyze large swaths of the lunar surface, collecting images using specialized micro-imager instruments. Using deep learning, these images could be analyzed on Earth or onboard powerful flight computers.

Deep learning has been leveraged in terrestrial applications to support HCI to accomplish goals such as autonomous driving, facial recognition, and natural language processing. In recent years, applications for space have been explored such as for Earth-observation satellite systems to detect cloud cover [46] and to extend human capability of counting craters on the Moon [48]. To properly train deep learning models, large datasets of images are required. Deep learning can be applied to space exploration and development, but is limited by the amount of data available. Using a transfer learning approach, this challenge can be mitigated. Transfer learning is a concept in machine learning where a pre-trained deep learning model applied to a larger, broad dataset can be fine-tuned for a specific application with a smaller dataset such as one containing images of in-situ lunar regolith.

In planetary geology, work has been done to automate the process of analyzing rocks on Earth and on other planetary bodies. Due to the similarity among particles within rock samples such as terrestrial sands, lunar surface regolith, and Martian rock formations, the task of identifying and classifying rocks from image data is challenging. As deep learning models have been successful for datasets with distinct objects, the question is whether they can be successful at extracting features and identifying objects from images where the similarity between objects is high as it is in geologic images. Liu et al. used a machine learning approach to extract features from images of rock minerals on Earth using an Inception-v3 pre-trained model achieving high accuracy results of a comprehensize model between 74-99% for the image recognition task [33]. Kim et al. used a transfer learning approach to classify different types of terrestrial sands and identify the source they were collected from by extracting features such as shape and surface texture resulting in classification accuracy on gray-scale images with an Inception-based network of 98.2% [27]. Kapui et al. analyzed Martian regolith using simple automated im-
1.2. Task and Thesis Contribution

This thesis explores the use of deep learning using a transfer learning approach to limited microscopic image data of the lunar surface. A dataset of in-situ lunar images was generated for the purposes of training a machine learning model to perform object detection of specific age analysis techniques to identify the mode of transport, fluvial or eolian, for the particles on Mars [25]. Li et al. used a VGG-16 transfer model to autonomously classify Martian rocks from images, stating 100% accuracy [30]. Kodikara et al. were interested in using machine learning techniques to determine the mineralogical and physical properties of lunar soils from the Lunar Soil Characterization Consortium (LSCC) [28]. With the success of distinct classification and identification of particles where rock and sand fragments are isolated, the challenge is to understand if deep learning with transfer learning can perform well with a small dataset of images of lunar regolith in-situ.

At the time of writing this thesis, Artemis I has just launched and returned to Earth after many years of anticipation, capturing new views of the Moon and Earth on its way. Humanity is returning to the Moon and with it, the promise of new rovers, scientific instruments, satellites and more that will deliver new data of the lunar surface. Most notably for this work, the Rashid rover developed by the United Arab Emirates’ Mohammed bin Rashid Space Centre (MBRSC) was equipped with a microscopic camera to collect new high-resolution images of the lunar surface. Onboard the Hakuto-R Mission 1 Lander from ispace, it launched in December 2022 to the Moon and planned to land on the surface in March 2023. Additionally, onboard the rover was a Mission Control Space Services AI flight computer which was planned to demonstrate the first application of deep learning beyond Low-Earth Orbit. However, in March 2022, as the ispace lander lost communication with the mission control center and was suspected to be unrecoverable. Despite the failure of this mission, there is great investment in returning to the Moon, and with many missions planned for the next decade, much can be learned to improve for future missions.
particle types within the images. Pre-trained models such as Faster RCNN and Single Shot Detectors (SSD) with ResNet backbone networks were fine-tuned on a small dataset of lunar images to understand if they are able to extract features relating to the geological characteristics of the objects and generalize for unseen data. As this project has been developed through a Mitacs funded project with Mission Control Space Services, the specific interest was to potentially support scientific mission operations and science objectives. It is possible that images collected from the lunar surface could be used to test such a model and improve on the data available once images become available. Ultimately, the work could help planetary geologists with the complex task of identifying particle types of geological data, without having to displace the material from the area of interest.

1.3 Thesis Outline

The thesis is organized as follows: Chapter 2 contains background information about the planetary geology aspect of this project, specifically the Moon and lunar regolith, as well as a literature review of machine learning methods pertaining to object detection and transfer learning. Chapter 3 outlines a proof of concept project which was done in the preliminary phase of this research to understand the problem and implement a procedure before obtaining the lunar image data for the remainder of the work. Chapter 4 details the methodology for preparing the dataset of lunar surface imagery and the considerations for the models selected. Chapter 5 discusses the results of training and testing the selected models with the dataset. Chapter 6 includes the conclusion and discussion about potential future work.
Chapter 2

Background and Literature Review

2.1 The Moon

In 1969, humanity set foot on the Moon for the first time. Since then, a great deal has been learned about the geological history of the Moon, technologies have advanced, and plans to return to the surface have been made. While it is Earth’s nearest neighbor, there is still more to be learned.

Figure 2.1: Captured by the Hasselblad 500EL 70 mm camera from the Apollo 11 command module, Columbia, as the Apollo Lunar Module Eagle ascends from the lunar surface to return to meet Columbia (AS11-44-6642)[38].
The Moon was formed approximately 4.6 billion years ago, shortly after the Earth formed [36]. The current theory of the Moon formation is the giant impact theory, which hypothesizes that the Moon formed after the Earth collided with a small object while it was forming and material was captured in Earth’s gravity, forming the Moon as a satellite. The only way to understand whether this theory stands true and how the rest of the solar system formed, including Earth, is by studying the Moon and other celestial bodies in the solar system closely. With lunar exploration missions, a great deal can be learned about the Moon, the solar system, and how to leverage the resources for potential use to push humanity deeper into space.

2.1.1 Lunar Regolith

The lunar surface is composed of a thick layer of unconsolidated rock called ‘lunar regolith’. This regolith is highly variable, consisting of particles such as breccia fragments, mineral and rock fragments, glass shards and beads, and melted, consolidated combinations of different particles called ‘agglutinates’. These particles vary in size, ranging from microscale to a few centimeters. As the majority of the regolith is fine-grained, the average grain size is between 60–80μm [20]. Depending on the location of the regolith on the lunar surface, the composition can vary. In the lunar mare, or lowlands, the composition is primarily basaltic, containing iron and titanium from possible volcanic activity in the early life of the Moon. In the lunar highlands, the composition is primarily anorthositic, containing aluminum, calcium, and silicon possibly originating from the once molten core of the Moon, and the rise and subsequent cooling to the surface of lighter minerals such as plagioclase and feldspar [39].

Lunar regolith is formed from impact processes such as micrometeorite impacts and large-scale meteor impacts. Over billions of years, the surface has been bombarded, resulting in the surface to be covered in varying sizes of rock fragments and dust. The thickness of the lunar regolith that sits atop the underlying bedrock is approximately 20 meters. As impacts bombard the surface, large impacts result in craters that excavate the regolith and expose the bedrock. The process of impact cratering results in rocks to break apart and heat up, depositing the regolith over large distances. This can also cause the rocks to melt, resulting in fragments
2.1. The Moon

to stick together forming breccias and agglutinates or rapidly become molten and cool forming melts and glass.

It is difficult to study the history of Earth through terrestrial regolith. Weathering and erosion caused by the presence of Earth’s atmosphere, human activity, and the changing geological landscape have removed evidence of the early state of the Earth. Conversely, the Moon does not have geologic activity, has low gravity, and virtually no atmosphere, meaning much of the surface remains in pristine condition.

While there are a variety of different particles that the lunar regolith is composed of, in this work, the fragments in the lunar regolith of most interest are breccia, glass, and melt.

Breccia

Breccias are composed of rock, mineral, agglutinate, and glass fragments [20]. The appearance of breccias is varied, and range greatly in size from small fragments on the millimeter scale to large rocks on the centimeter scale. Breccias are formed from impact cratering processes that result in the surface material to break apart and melt together. The colouration, textures and the types of mineral inclusions vary as well. There are different types of classes of breccia. For example, fragmental breccia contain only the fragmental material, whereas regolith breccia contains glass and agglutinates.

Melt

Melts are formed when molten impact melt splashes onto exposed regolith and generally include inclusions of other materials [20]. Melt appears as shiny material, similar to glass, and irregularly shaped, sometimes appearing bubbly. The scale of these melts can range from millimeter to centimeter range.

Glass

Clear lunar glass beads and fragments without inclusions form from impacts and past volcanic activity. Glass beads are typically small and spherical. There are a high population of glass
particles in lunar regolith, approximately 3–5% of all lunar regolith [20]. Within the regolith, there are many small scale fragments. These are primarily formed by impact melting resulting in the formation of small beads due to the lack of atmosphere and low gravity on the Moon. These particles are generally on the millimeter to submillimeter scale.

Additionally, volcanic glasses for glass spherules which could have been generated from explosive eruptions, called fire-fountain eruptions, that had shot material high into the air and resulted in the formation of spherules when the material landed [36]. Commonly, these glass beads present as orange and green in colour. Impacts can expose these materials that have been covered over a long period of time since the Moon no longer has volcanic activity.

![Figure 2.2: Examples of different rock fragments present in lunar regolith including glass, breccias, impact melts, and agglutinates [37].](image)

2.1.2 Lunar Regolith Simulant

There are limited samples of lunar regolith that were collected and returned to Earth during the Apollo missions. To support research on Earth for understanding the lunar surface com-
2.1. The Moon

position, lunar regolith simulant has been developed by scientists. There are various different types of lunar regolith simulants that have been developed representing different characteristics of the lunar surface. The lunar regolith simulants can vary based on whether they are more anorthositic, representing the lunar highlands, or more basaltic, representing the lunar mare.

OB-1

The OB-1 is a lunar regolith simulant developed by NORCAT to represent the anorthosite-rich lunar highlands (Fig. 2.3) [2]. The simulant is composed of Shawmere anorthosite and olivine slag glass originating from Kapuskasing Structural Zone of Ontario, Canada. It gets its name from its olivine-bywonite composition.

![Sample of OB-1 simulant captured by author.](image)

Dr. Melissa Battler from Mission Control Space Services, who has heavily supported this research, was involved in the development of this lunar regolith simulant as a part of her Master of Science thesis during her time at the University of New Brunswick.

JSC-1A

JSC-1 was developed by NASA Johnson Space Center to represent the lunar mare (“seas” or lowlands) and the basaltic composition of this region (Fig. 2.4). The simulant is composed of basalt ash from the San Francisco Volcanic Field in Arizona.
2.1.3 Angle of Repose

Another important characteristic of in-situ lunar regolith is the angle of repose. It is a distinguishing factor between lunar regolith on the surface of the Moon compared to lunar regolith simulant on Earth as different forces interact with the material. It is the steepest angle before loose rock becomes unstable. Furthermore, it is the balance between resistance and the force of gravity on a planetary body. The angle of repose for rock is approximately 30°. The angle of repose of lunar regolith is approximately 40°, greater than loose rock on Earth [36]. Due to the small, sharp particles and electrostatic forces, the regolith particles cling together.

In the low gravity environment on the Moon, this angle of repose is close to that of wet sand,
and requires special considerations, such as simulation in a vacuum environment, to replicate on Earth to properly understand regolith on the lunar surface.

### 2.1.4 Apollo Missions

With the success of the Apollo missions, a new era of lunar research began as more information was returned to the Earth in the form of sample return and surface imagery. From 1961 to 1972, 11 total Apollo missions took place. The missions conducted tests for equipment, with six missions landing humans on the surface to take images, collect samples, and provide observations of the surface. Different experiments were conducted to further understand the composition and formation of the Moon. Six missions returned samples from the lunar surface to the Earth, totaling approximately 370–380 kg. These samples were tested extensively on Earth in vacuum environments to understand the surface composition. Three Soviet rover sample return missions successfully returned approximately 300 grams of samples between 1970–1976. In 2020, the Chang’E 5 mission returned 1.73 kg of samples from the Moon [50].

**Apollo Lunar Surface Close-up Cam (ALSCC)**

During the Apollo 11, 12, and 14 missions, the ALSCC instrument was onboard to take high resolution images of the lunar surface. The images were captured by astronauts on the surface of the Moon using the handheld instrument. It was developed by Eastman Kodak and contracted by NASA [38]. Consisting of two cameras, the stereoscopic images provided a unique understanding of the depth and features of the surface regolith. The spatial resolution was significant, allowing for visibility of microscopic details of the lunar regolith in-situ. Images like this are expected to be collected once again over the next decade with various planned lunar rover missions to carry high-resolution imaging instruments onboard. The specifications of the camera are as follows in Table 2.1.

*March to the Moon* is an online digital repository created by members of NASA’s Johnson Space Center who scanned the original films from the Mercury, Gemini, and Apollo missions to be shared publicly online [38].
Table 2.1: Specifications of the ALSCC onboard the Apollo 11, 12, and 14 missions [38].

<table>
<thead>
<tr>
<th>Specification</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area photographed:</td>
<td>72 mm × 82.8 mm</td>
</tr>
<tr>
<td>Focus:</td>
<td>fixed range</td>
</tr>
<tr>
<td>Aperture:</td>
<td>f/22.6 fixed</td>
</tr>
<tr>
<td>Film:</td>
<td>30-ft S0368 112 stereo pairs</td>
</tr>
<tr>
<td>Surface Particle Identification:</td>
<td>as low as 0.004 inch</td>
</tr>
<tr>
<td>Resolution:</td>
<td>80 µm</td>
</tr>
<tr>
<td>Magnification:</td>
<td>0.33 times</td>
</tr>
<tr>
<td>Base-Height Ratio:</td>
<td>0.16 for stereoscopic photos</td>
</tr>
<tr>
<td>Stereo Angle:</td>
<td>9° convergent</td>
</tr>
<tr>
<td>Cycling Time:</td>
<td>10 sec</td>
</tr>
<tr>
<td>Object Plane Coverage:</td>
<td>9 sq. inch</td>
</tr>
<tr>
<td>Lighting:</td>
<td>flash illumination</td>
</tr>
</tbody>
</table>

The Apollo 11 mission ALSCC images were collected in the general vicinity of the lunar module landing site, Tranquility Base. Neil Armstrong and Buzz Aldrin took images of the surface with the instrument and logged their general observations. There were many instances where objects of interest within the regolith at their eye level were not able to be distinguishable, but more detail could be seen through the images. Observations from the images were made back on Earth after the mission and cataloged in the Apollo 11 Preliminary Science Report [1]. The Apollo 12 mission ALSCC images were also taken close to the lunar module landing site in Oceanus Procellarum [19]. The images captured during Apollo 12 were similar to Apollo 11. Apollo 14 landed at Fra Mauro and the images were taken further from the lunar module landing site to avoid any affects of the exhaust from the landing to affect the surface [51]. Images of astronaut boot prints were captured, showing prints as deep as 10 cm. Due to the angle of repose and the cohesive regolith, the boot prints are deeper than what would be expected on Earth. The Apollo 11 and 12 images represent regolith from the lunar mare, while Apollo 14 images represent ejecta Imbrium basin [21]. Figure 2.6 details the approximate
locations of the landing sites for Apollo 11, 12, and 14.

Figure 2.6: Approximate locations of the Apollo 11 (Tranquility Base), Apollo 12 (Oceanus Procellarum), and Apollo 14 (Fra Mauro).

2.1.5 Imagery of the Lunar Surface

The ALSCC images provided microscale images of the lunar surface. There are limited examples of detailed, high resolution images of in-situ regolith from missions over the past decades since Apollo. Since the mid-1900s, humanity has ventured closer to the lunar surface, and captured images from different scales and collected samples from the surface. The first mission to capture images of the Moon was the Luna 3 Soviet spacecraft in 1959. It took images of the lunar far side and returned them to Earth. The images were captured at approximately 63,500 km from the surface of the Moon and were noisy when transmitted to Earth, as can be seen from Figure 2.7a [52].
Since then, greater quality of images have been captured and provided understanding of the Moon. For example, the Lunar Reconnaissance Orbiter was launched in 2009, providing detailed images of the lunar surface from an altitude of 35–50 km. The image resolution could resolve to approximately 50 cm/pixel, showing details of the heavily cratered surface (Fig. 2.7b) [29].

The Chang’E-4 mission landed the Yutu-2 rover on the lunar far side in 2019 to explore the South Pole Atkien crater. It is the first uncrewed mission to land on the lunar far side, and as of writing this thesis, is the longest living lunar rover mission. Different instruments are onboard the Yutu-2, but of most interest is the panoramic camera (PCAM) which has 360-degree rotation and a spectral range of 420-700 nm, acquiring 3D stereoscopic images [24]. The camera was able to capture images of glass spherules on the surface of the Moon ranging from 1.5–2.5 cm in diameter (Fig. 2.8)[53]. This high resolution imagery of in-situ lunar regolith is limited and provides great significance for research of the lunar surface.

The Emirates Lunar Mission’s Rashid rover launched as a part of the Hakuto-R mission with ispace in December 2022. It will land at Atlas crater on the surface of the Moon in February 2023. The mission plans to capture high resolution images of the lunar surface, expecting to provide either similar or better images to the ones captured by the ALSCC. There are two cameras onboard, a compact close-up imaging camera called CAM-M which will capture small details of geological features on the surface, and a thermal camera [47]. The images that will
2.2. Machine Learning

Figure 2.8: Images of glass sperules on the lunar surface captured by the Yutu-2 rover [53].

increase the amount of data available of in-situ lunar regolith.

2.2 Machine Learning

Machine learning is a field of study within artificial intelligence focused on understanding and developing methods for computer systems to learn similar to how humans learn. Using a variety of statistical models and mathematical implementations, machines are able to extract important information from data and use that to complete complex tasks and recognize patterns.

The main focus of this thesis is the processing of image data using machine learning techniques for a specific task. As images are simply M×N matrices of pixels, these machine learning models can manipulate the matrices to reduce the size, extract important elements, and train models to learn the context of the images.

2.2.1 Learning Methods

Machine learning models learn using three different methods: supervised learning, unsupervised learning, and semi-supervised learning.

Supervised learning methods refer to providing a model with information about the specific class categories and objects in an image for training. In this case, the model learns what to look for and what is significant within the data based on what the model designer tells it by the means of a label. This requires datasets that are fully labelled, which can be time intensive
and difficult to find large dataset for new data. For example, with images of dogs, a designer would label the entire images as ”dog” and the model would be able to learn this relationship explicitly.

Unsupervised learning methods refer to providing a model with a dataset and having it cluster important features. These models will extract features from the data based on relationships between the features. In this case, models decide what is significant based on how the features of the data relate to each other. This requires unlabeled datasets and can find unique relationships in data. For example, with images of dogs, the model would provide a large dataset of dogs and the model would be able to find relationships between all the images to identify that a dog is present in a new image of a dog.

There are many benefits and trade-offs for using the different learning techniques and requires a model designer to understand the considerations relating to the problem to be solved.

2.2.2 Features

A feature within an image represents a characteristic that has a measurable attribute which can be extracted by a machine learning model. The model uses features to understand visual information in an image for classifying and clustering information. Features in image data for this research consisting of lunar regolith can include colour, shape, size, and texture.

Feature extraction is performed in machine learning to reduce the image data to feature vectors which describe the features in the data. Within convolutional neural networks (CNN), the convolutional and pooling layers are used to extract features and assign importance. In this research, since a transfer learning approach is used, the feature extraction for the new dataset is performed to fine-tune a pre-trained model.

2.2.3 Gradient Descent

Gradient descent is an important aspect of many machine learning models as it is the optimization algorithm that reduces the cost, or loss, function of the model. The cost function describes
the relationships between the predicted output and the actual output. The lower the cost, the closer the prediction is to the expected outcome. Given a function such as a simple convex function, the goal is to reach the local minimum value. An arbitrary initial position is selected, and the slope is calculated. To begin, large steps in the direction of the steepest portion of the function are taken. Once the slope begins to flatten, smaller steps are taking to avoid overshooting the minima. Functions in machine learning are more complex and have multiple local minima and maxima. It is not possible to know if one has obtained the global minima value, however, there are techniques to support greater optimization. The gradient descent algorithm is described by:

\[
\theta = \theta + \eta \nabla \theta J(\theta)
\]  

(2.1)

where \(\theta\) represents the training parameters, \(\eta\) represents the learning rate, the step size for the algorithm as it approaches the local minima, and \(J(\theta)\) represents the cost function to be minimized. The goal is to step towards the minimum value of the slope in the opposite direction of the gradient cost function (\(\nabla \theta J(\theta)\)) [42]. This is visualized simply in Figure 2.9.

![Gradient Descent Visualization](image)

Figure 2.9: Basic visualization of gradient descent of a cost function. Arbitrarily starting at point A, the gradient descent algorithm’s goal is to step towards the local minima at point B [26].

An important aspect of gradient descent is proper selection of the step size, otherwise known as the learning rate. The learning rate is a parameter that can be adjusted in a model to help a model to converge effectively. A large learning rate may help this convergence happen
quicker, but may lead to overshooting the minima resulting in a larger rate of error. A small learning rate would help ensure the local minima is achieved, however, could take much longer and not have a meaningful benefit. This is visualized in Figure 2.10.

![Figure 2.10: Comparison of a big learning rate resulting in the algorithm overshooting the local minimum, and a small learning rate which converges to the local minimum slowly [10].](image)

Batch gradient descent (BGD) computes the gradient for cost function over all training examples at each step, which makes it computationally expensive to perform with large datasets. Stochastic gradient descent (SGD) computes the gradient of the cost function for a random sample of training at each step, which makes it faster to perform over BGD.

As a NN is trained, the connections between the nodes at each layer are transferred to the next as a part of the training algorithm, from the input to the output. The cost function represents the error between the predicted and expected output, with the goal being to minimize the error. Backpropagation is an algorithm for calculating the gradient which is used to update the weights of the nodes after a pass through the network to reduce the error.

### 2.2.4 Neural Network (NN)

A neural network (NN) is a computer architecture modelled after a biological brain and the connections of nodes within it [3]. The nodes and layers within a NN allow for the modelling of the relationship between an input and an output to help the network learn and ultimately achieve a task such as classification and clustering. The most rudimentary version of this is an input layer, one or more hidden layers, and an output layer (Fig. 2.11). Data is fed into the
neural network from the input layer. The hidden layers pass information from one layer to the next, processing and extracting important information. The result from the neural network is outputted at the output layer. For example, for a multi-class problem, the output layer would consist of multiple options for the classification relating to each class. Common NNs are feed-forward networks and recurrent networks. Feed forward NNs pass information from the nodes from input to output without looping back. Recurrent NNs can have information passing backwards through the nodes. In this work, feed forward NNs using multi-layer perceptrons are used for their effectiveness in working with the relationship of input data to outputs, such as for object detection.

![Diagram of simple neural network with an input layer, at least two hidden layers, and an output layer](image)

Figure 2.11: Diagram of simple neural network with an input layer, at least two hidden layers, and an output layer [9].

Applications of NNs are in computer vision, speech recognition, natural language processing, search engines, and more.

**Convolutional Neural Networks (CNN)**

Convolutional neural networks (CNN) are popular architectures used in deep learning and have gained popularity. CNNs are particularly good at extracting features to generalize on the input data, are computationally efficient compared to classical models, and boast high accuracy for various applications. The architecture of a CNN consists of a convolutional layer, polling layer,
fully connected layer, and an activation function (Fig. 2.12).

Figure 2.12: Example of CNN architecture consisting of input image, convolutional layers with ReLu activation, pooling layers, fully connected layer, and softmax activation function for the classification output [45].

The convolutional layer consists of producing a feature map by applying a convolutional filter, or “kernel”, to the data to extract features. The kernel slides over the pixels of an image, generating the feature map through this convolution operation. The receptive field is the region where the kernel overlaps the image and where the convolution occurs. This continues for all pixels in the image. Images are represented as 3-dimensional with a height, width, and depth, so this convolution operation takes place in 3-dimensions, outputting a scalar value for the feature map [23]. The convolution formula is as follows:

\[
  a_{ij} = \sigma((W \ast X)_{ij} + b)
\]  

\(\sigma\) introduces non-linearity into the network by passing the convolution result to an activation function which helps the model fit more complex problems. \(b\) is the bias, a value added to adjust the weights of the feature map to help with fitting the data. \(W\) represents the kernel, or weights, that goes over the pixels in the image, and \(X\) represents the input.

The pooling layer implements a function to reduce the number of parameters by reducing the height and width of the output from the convolutional layer. Max pooling is a common technique. A similar method to the kernel window moving across the input image, the pooling window shifts across the input and takes the maximum value from the window.
Next, the fully connected layer consists of the output of the convolutional and pooling layers as an input. Here, the kernel weights matrix and the input matrix undergo the dot product to obtain the output.

The activation function introduces non-linearity to the neural network which ultimately helps the model fit the input data. Sigmoid and softmax had been used frequently in the past within CNNs. Softmax is an extension of sigmoid and is more suitable for multi-class classification, whereas sigmoid is used more in binary classification. However, rectified linear units, or ReLu, activation has gained more traction due to its simplicity and efficiency in implementation and training.

CNNs consist of two parts, a feature extractor made up of the convolutional and pooling layers, and the classification method made up of the fully-connected layers. They are powerful tools for deep learning.

**Region-Based Convolutional Neural Networks (R-CNN)**

R-CNN builds upon the foundations of classical CNNs. Initially, the ”selective search” method is implemented to scan an input image to identify potential regions where an object might exist. Approximately 2000 regions are proposed with this method. Next, a CNN is used to extract features from the image relating to the objects in each region, meaning 2000 CNNs are run for each of the 2000 proposed regions. And finally, classification using a support vector machine.
(SVM) and localization using linear regression is performed on the output image to identify bounding boxes and labels for possible objects (Fig. 2.14) [15].

Figure 2.14: Overview of R-CNN network for object detection. Regions are proposed over an input image. Then, for each proposal, features are extracted using a CNN before outputting the classification for the region [15].

2.3 Deep Learning for Object Detection

Training machines to understand the visual information of the world is a difficult task. Deep learning is a powerful technique in computer vision consisting of multi-layered, or ”deep”, networks which can be trained with large amounts of image data to perform tasks such as object detection and classification [54]. Object detection is a computer vision task which combines image classification and object localization. Image classification is the task of assigning a class label to an image. An image is fed into an image classification model and the label is outputted. Object localization is the task of locating an object within an image. The output for an object localization task is a bounding box around a single object in an image.

The labels take into consideration the context, size, and location of the object in an image. For object detection tasks, images can be labelled approximately with a bounding box, or exactly using pixel-wise segmentation. For this research, a bounding box labelling approach was used and will be described in further chapters. The number of objects and the size of objects in an image can vary, which makes it very useful for problems where a model needs to generalize on the input data, but also introduces additional complexity. A sliding window approach has
been used to shift over the pixels in an image to extract feature vectors for different positions in an image and formulate the location of objects.

Classical object detection uses hand-tooled algorithms to assign descriptor vectors to selected features. Some approaches include Histogram of oriented gradients (HOG) [7]. Simply put, HOG are used to extract relevant information from an image relating to the features by calculating the gradients of each pixel and representing the information in a histogram. Support vector machines (SVM) are also used in classical object detection. Building on the fundamentals of linear and logistic regression, SVMs are supervised learning models based on regression for classification problems. They fit a hyperplane, or decision boundary, within N-dimensional data consisting of labels that maximizes the distance between the classes.

The key distinction between classical approaches and deep learning is that in deep learning the network is allowed to learn the feature set as well as the classification. Since the introduction of deep learning, there have been great advancements in object detection as will be discussed further in this thesis. Various competitions have been held internationally to benchmark large datasets to process visual data for these computer vision tasks such as image classification and object detection. Some examples are the Pattern Analysis, Statistical Modeling, and Computational Learning Visual Object Classes (PASCAL VOC), ImageNet Large Scale Visual Recognition Challenge (ILSVRC), and Microsoft’s Common Objects in Context (COCO) [32] [43]. These competitions have lead to many successful models used in real-world applications developed from leveraging these large labelled datasets consisting of hundreds of thousands of images [13].

2.3.1 Transfer Learning

Transfer learning is a concept in machine learning which leverages pre-trained models that have access to large amounts of labelled data for training, and transferring the knowledge from the one task to a new task that has a smaller amount of data available. Large pre-trained networks can have their final layer fine-tuned on a smaller dataset for a specific task. This is analogous to how humans learn from one another, by adapting learning based on the learned
experiences of others that are shared. Generating large, labelled datasets are a difficult, resource intensive tasks, so transfer learning is particularly helpful. Many large, pre-trained networks have been created and annually benchmarked to improve performance of the models and size of the datasets.

2.3.2 Object Detection Models

There are various models that have been used for object detection tasks. Object detection models consist of two parts, a backbone network which is pre-trained on a dataset such as ImageNet or COCO, and the head consisting of either a one-stage or two-stage detector which localizes objects in an image with a bounding box and classifies objects using labels (Fig. 2.15). Two-stage models consist of two models: one to extract regions in an image, and one to classify and localize objects within the regions. In one-stage models, predictions are directly made without additional steps for region proposals like in two-stage models. You Only Look Once (YOLO) and Single Shot Detector (SSD) are examples of one-stage models, and Faster R-CNN and Mask R-CNN are examples of two-stage models. Common backbone networks are ResNet, EfficientNet, and VGG16 [4].

![Figure 2.15: One-stage vs. two-stage detectors [4].](fig2_15.png)

**Faster R-CNN**

The Faster R-CNN model architecture is based on its predecessors, R-CNN and Fast R-CNN [14]. Building on R-CNN, Faster R-CNN replaces the selective search method with regional
2.3. Deep Learning for Object Detection

Proposal network (RPN) for identifying regions where objects may exist and generates a feature map. For each proposed region, a region of interest (ROI) pooling layer extracts a fixed-length feature vector. The RPN requires two passes, one to generate the region proposals, and another to detect objects within the regions. The ROI window iterates over the feature map and uses max pooling to generate the feature vector. The feature vectors are fed into a Fast R-CNN network which classifies the information and proposes bounding boxes for each region[5][40].

Faster R-CNN models have been used in multiple applications and are a popular choice due to good accuracy and efficient performance. The Faster R-CNN presented by He et al. won first place in the COCO 2015 and ILSVRC 2015 object detection competitions [18][40].

Single Shot Detector (SSD)

Single shot detectors (SSD) use one-pass for object predictions, unlike models like Faster R-CNN, resulting in predictions to be made quicker (Fig. 2.17). A single CNN is used to generate bounding boxes around objects in an image and provides confidence scores for the presence of objects within the boxes. Non-maximum suppression (NMS) is a model post-processing step used to reduce the number of bounding boxes by removing the lowest scoring boxes around the same object and only maintaining the highest confidence box[34].
SSD has been tested on the PASCAL VOC2007 dataset, showing improvements in performance accuracy over Faster R-CNN[34].

ResNet

Networks like ResNet, AlexNet, and VGG are deep CNNs that have tens to thousands of layers. As networks get deeper, the vanishing gradient issue arises, where the values of the gradient reduces to zero making it hard to train a network. ResNet implements a technique called ”identity shortcut connection” which helps with the vanishing gradient problem by skipping connections without hurting the learning process [18]. Since this does not add further complexity to the network, it is a powerful solution to implementing deep networks without losing quality. In this work, ResNet is used as a backbone network for the models as a feature extractor. 50-layer, 101-layer, and 152-layer ResNets will be tested with the dataset produced.

2.3.3 Model Parameters

When designing a machine learning model, there are ‘hyperparameters’ that can be modified to change how a model performs.

Common parameters that require tuning are:

- Number of epochs (related to number of steps)

- Batch size
2.3. Deep Learning for Object Detection

The output from the previous layer is applied to the next layer [18].

- Learning rate
- Optimizer
- Loss Function

**Epochs, Batch Size, and Steps**

A batch describes the number of training samples before the parameters of the model during training are updated. An epoch is the number of passes, both forwards and backwards, for the training set for one cycle of training. To train a model which minimizes the cost function, or loss, effectively, hundreds to thousands of epochs are required. Say there are 1000 samples in the training set and the batch size is equal to 10, then the model would train on the first 10 samples, update the parameters, then train on the next 10, and so on. Doing this procedure once constitutes one epoch [16]. In this work, the number of steps will be updated in the model configuration file using the TensorFlow 2 Object Detection API (Section 2.4.1). A step refers to one operation for the optimizer to update the weights of the model. The relationships between the number of steps to epochs and batch size is:

\[
\text{num\_of\_steps} = \frac{(epochs \times \text{num\_of\_training\_imgs})}{\text{batch\_size}}
\]  

(2.3)
Learning rate

As mentioned when gradient descent was discussed in Section 2.2.3, the learning rate is a parameter that can be adjusted to change how quickly and effectively the gradient resolves to a minimum of the cost function [42]. A higher learning rate leads to larger steps to be taken, resulting in faster convergence, but can result in the algorithm overshooting, or missing, the optima. A lower learning rate allows for effective convergence, but can greatly increase training time. Tuning this parameter can be a good way to achieve better performance by evaluating the loss of the model after training and evaluation.

Optimizers

There are many optimizers that can be used for different problems in machine learning. Computing the gradients at each step to reduce the error, many optimizers can be used with SGD. Common ones include Momentum, Adam, AdaGrad, and RMSProp [42]. For this work, Momentum optimizer was used. Momentum helps the SGD algorithm converge to a minima faster by overcoming the problem of oscillations that occur as the gradient descent algorithm searches for a minimum. Momentum reduces the impact of potentially moving in a different direction for the gradient by implementing a historical value relating to the previous step as a value close to 1 to the calculation. The closer the momentum, or history, value is closer to 1, the more impact the previous step has on the calculation of the next. This parameter can be updated to see how different optimizers affect the loss function.

2.3.4 Training, Testing and Validation

In machine learning models, splitting the dataset into training, testing and validation sets is important for evaluating a model’s performance with unseen data. The training set consists of a larger portion of the dataset used for fitting the model to the data. The training set ranges between 50% to 90% of the dataset. The test set consists of a smaller portion of the dataset for unbiased testing of model performance after training on new, unseen data. The test set ranges
between 10% to 50% of the dataset. A validation set is used to evaluate the model during training to provide insight into hyperparameter tuning. The validation set also helps to prevent the model from overfitting to the dataset during training. The model is biased to the validation set as it sees it during training, so a test set is necessary to provide an unbiased evaluation of the model performance. The validation set ranges from 10% to 20% of the dataset.

Different train/test/validation split ratios are selected based on the dataset. For example, if there are not many hyperparameters to be tuned, a smaller validation set can be used. There is no optimal or standardized split ratio used for machine learning and requires understanding the dataset and the model requirements.

### 2.3.5 Data Augmentation

Data augmentation is used to increase the size of datasets. This is especially useful in deep learning which requires large datasets for training. Image processing techniques are used to vary the image data to create new images to increase the number of images. An example of data augmentation can be seen in Figure 2.19.

![Figure 2.19: Example of data augmentation on an example image from the MNIST handwritten digit dataset [12].](image)

Some common data augmentation techniques are Gaussian blur, rotations, horizontal flips, random flips, lighting changes such as contrast and brightness adjustment, colour changes, and more. For object detection, the labels are transformed along with the images that are being processed.
2.3.6 Evaluation Metrics

To understand if a machine learning model is performing well, various metrics are used to compare how well the model performs with new data compared to the ground truth labels. In object detection, class labels are assigned to objects in an image within the dataset, called ground truth labels. Once a model is trained, it can be tested on unseen data to see how well it predicts object locations and labels. By comparing how many predictions are true positives, false positive, true negatives, and false negatives can give insight into the overall performance of the model. The relationship between these metrics and the predicted and ground truth labels is referred to as the confusion matrix (Fig. 2.20).

A True Positive (TP) prediction is when the model predicts the location and class of an object in an image, and it matches the ground truth label. A False Positive (FP) prediction is when the model predicts the location of an object of a specific class where there is no ground truth label. A False Negative (FN) occurs when the model does not predict an object in an image where there is a ground truth label. A True Negative (TN) is when the model does not predict an object where there is no ground truth label, however, true negatives are not a part of the model evaluation in object detection. For this work, a positive class is a specific particle type of lunar regolith whereas a negative class is background or not a particle.

The accuracy score measurement provides an understanding of how the model performs for
2.3. Deep Learning for Object Detection

all classes.

\[ \text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \]  

(2.4)

The precision score describes the ratio of true positives to the total number of positive predictions. It is a metric for understanding how many of the positive predictions are correct. The precision is better when it is closer to 1 and means that the impact of false positives, or instances where objects that should not have been detected are, is lower.

\[ \text{Precision} = \frac{TP}{TP + FP} \]  

(2.5)

The recall score describes the ratio of true positives to the total number of positive results. It is a metric to understand how many of objects are correctly detected that match the ground truth labels. The recall is better when it is closer to 1 and means the impact of false negatives, instances where objects that should have been detected are not, is lower.

\[ \text{Recall} = \frac{TP}{TP + FN} \]  

(2.6)

**Intersection Over Union (IOU)**

Intersection over union (IOU) is a metric for object detection to determine if an object detection prediction correctly matches the ground truth label, related to the accuracy metric (Fig. 2.21). The predicted bounding box is compared against the ground truth bounding box for an object in an image. A higher IOU value closer to 1 is ideal as it means that the predicted bounding box and the ground truth bounding box are overlapping. An IOU value greater than 0.5 generally means that it has detected the object.

\[ \text{IoU}(A, B) = \frac{A \cap B}{A \cup B} \]  

(2.7)
Average Precision (AP) and Average Recall (AR)

The average precision (AP) represents the area under the precision-recall curve (Eq. 2.8). The precision-recall curve is obtained by plotting the precision against the recall for different thresholds. The AP is calculated as the mean over 11 equally-spaced values of recall from 0 to 1. The value ranges between 0 and 1, and is high when the precision is high for all values of recall. The AP is calculated per class.

\[
AP = \int_{r=0}^{1} p(r)dr 
\]  

(2.8)

The average recall (AR) is the average of the recall value calculated over different IOU thresholds between 0.5 to 1 and class categories (Eq. 2.9).

\[
AR = 2 \int_{0.5}^{1} recall \ast IoU_{d}(IoU)
\]  

(2.9)

Mean Average Precision (mAP)

Mean average precision (mAP) is a metric based on IOU, precision, recall and confusion matrix. It is a good metric for understanding the performance of an object detection model. The average of the AP for each class is calculated over different IOU thresholds. Equation 2.10 describes how the mAP is calculated. The AP for each class, \(i\), is calculated and averaged over the total number of classes, \(N\).
2.3.7 Loss

Loss is a metric that describes the error between the predicted output and the ground truth, related to the cost function. The goal of machine learning models is to minimize the average loss. A low value for the loss function indicates that a model is generalizing well.

A low training loss means that the model is fitting well on training data. Validation loss can be used to compare how the model performs on new data. A validation loss that is reaching a minimum similar to the training loss means that the model is performing well on new data. However, a higher validation loss than training loss, or a validation loss that decreases and increases again indicates that a model is overfitting. Overfitting occurs when a model is unable to generalize well, basically memorizing the training data and not being able to understand new data.

Different types of losses are calculated during training and evaluation of the object detection model using the COCO Evaluation API which will be discussed in Section 2.4.3. Classification loss and localization loss describe the error for each of the tasks respectively. Objectness loss describes the error for the model identifying whether an object exists within a region.

2.4 Tools

2.4.1 TensorFlow 2 Object Detection API

Developing a custom object detection model from scratch is a significant task which requires high computational power and time. Over the past decade, tools have been developed by organizations such as Google to reduce the resources required for developing and deploying flexible object detection models. The TensorFlow 2 (TF2) Object Detection API is an open-source framework for building and training object detection models for deployment [49]. The TF2
Object Detection API allows users to configure pre-trained models with architectures including Faster R-CNN, Mask R-CNN, and SSD for example. The models that are pre-trained and available for use with the TensorFlow 2 Object Detection API have been trained on the COCO 2017 Dataset. Hyperparameters such as batch size, input image size, number of steps, learning rate, and optimizer can be adjusted to try to improve model performance. While training the model, an evaluation job can be launched to see how the model performs on the validation set. The COCO API metrics are used to simultaneously evaluate the model for popular metrics such as mAP, AR, and loss and display them on Tensorboard, a user-interface to view the evaluation metrics.

2.4.2 LabelImg

To prepare the images for the object detection model, the LabelImg open-source, Python tool was used to label the glass particles with bounding boxes. Labeling supports two formats for bounding box labelling: Pascal VOC and YOLO. For compatibility with the TensorFlow 2 Object Detection API, Pascal VOC format was selected. Pascal VOC encodes bounding boxes with 4 values: xmin and ymin for the top-left corner of the box, and xmax and ymax for the bottom-right corner of the box. The labels for each image are saved in an XML generated by the LabelImg tool. The annotations are converted to TensorFlow records to prepare for object detection with the TF2 object detection API.

When preparing a dataset for object detection, it is important to be consistent, accurate, and exhaustive. Setting criteria for the labels to assign for each class helps with ensuring that objects in images are labelled well prior to model training.

2.4.3 COCO Evaluation Metrics

The COCO 2017 competition used metrics such as mAP and AR to evaluate model performance. It serves as a benchmark for other models. In the COCO evaluation metrics documentation, mAP is considered the same as AP, as is mean average recall (mAR) the same as AR.
Figure 2.22 shows the metrics used by COCO to evaluate the models and their descriptions [6].

The mAP is averaged over different IOU thresholds for all object categories. In the COCO API, 10 IOU thresholds are used between confidence score of 50% and 95% with a 5% step size. It is also evaluated at IOU equal to confidence scores of 50% and 75%.

The AR in the COCO evaluation metric represents the maximum recall for 1 detection per image, 10 detections per image, and 100 detections per image. It is averaged over different IOU thresholds and class categories.

The mAP and AR is also calculated for different scales of objects to understand how the detector performs for small (less than $32^2$ pixels), medium (between $32^2$ and $96^2$ pixels), and large (greater than $96^2$ pixels) objects.

### 2.5 Deep Learning Applications for Geology

Various systems on Earth implement deep learning for autonomous vehicles, face detection, security systems, and more. In recent years, deep learning and autonomy has begun to be explored for space applications such as satellite autonomy for refueling and collision avoidance, and rover path planning and traverse.

In the realm of geology, on Earth and in space, characterizing rocks helps with understanding the formation of the Earth, Moon, and the solar system as a whole. Additionally, the ability
to characterize rocks supports the exploration of possible useful resources. Deep learning with transfer learning has been used in various applications, testing various models, and obtaining reasonably good results for machine learning tasks. Due to the similarity between rocks to one another in samples, general approaches use individual grains for the machine learning tasks. Kim et al. proposed an approach to identify different terrestrial sand types from imagery [27]. Each class of sand was geometrically similar. The images consisted of individual grains of sand. CNNs and pre-trained models such as VGGNet, ResNet and Inception were used for the task. Liang et al. analyzed fine-grained rock images using a deep learning approach for an image classification task, observing high accuracy for their algorithm [31].

For lunar applications, Kodikara et al. applied different machine learning models to spectral data obtained of the lunar highland and mare from various missions [28] and Silburt et al. used a deep learning approach to identify and approximate the age of craters on the lunar surface [48]. There have also been promising examples of applications of deep learning and transfer learning for Martian geology as well. Li et al. described a method implementing transfer learning for Martian rock images [30]. Each image contained a distinct rock image. Models such as VGG-16 were compared with other models for Martian classification, resulting in overall high performance. The various applications of deep learning with transfer learning on both Earth and in space are promising for advancement in this field of research. This work builds on similar concepts discussed for rock particles within images where the particles are within the natural scene.

MoonNet from Mission Control space services was set to land in Atlas Crater onboard the Rashid Rover as the first demonstration of deep learning beyond low-earth orbit in March 2023[22]. MoonNet is an AI system developed to support mission operations and experimentation for rover missions [35]. However, the lunar module lost communication and was suspected to be unrecoverable. This is still proof of advancements in the field of AI and space exploration. There are likely to be more applications of machine learning and AI systems to process data from planetary missions on Earth and in space over the coming decades.
Chapter 3

Lunar Regolith Simulant Proof of Concept

This chapter outlines the procedure for manually developing a custom dataset of lunar regolith simulant to be used for training an object detection modeling using a transfer learning approach. The goal of this project was to create a preliminary procedure for preparing a small dataset of analogous images ahead of obtaining images of lunar regolith from the Apollo Lunar Surface Close-up Cam (ALSCC) missions.

3.1 OB-1 Dataset

For this work, it was important to develop a dataset of lunar regolith simulant that had a variety of different particles present in samples. The variation in the samples would need to represent what is most expected on the lunar surface such as glass, rock, and other fragments. As such, OB-1 simulant was used due to the distinct particles that can be observed from samples. JSC-1A lunar regolith simulant was also considered for this research, however, was not selected due to its composition. JSC-1A has a generally uniform composition as can be seen from Figure 2.4 and was not suitable for the task of identifying distinct particle types.

A vial of approximately 20 grams of OB-1 lunar regolith simulant was obtained. Various particles could be observed such as a combination of fine-grained elements, shards of light rock, and dark slag glass particles.
3.1.1 Data Collection

As there is no known dataset of labelled lunar regolith simulant imagery available, the images were captured manually by the author. The images were captured using specifications which attempted to simulate a lunar instrument onboard a rover for the purpose of capturing high-resolution imagery of the lunar surface. A Fujifilm X-T200 24.2MP compact system camera was used for the task of capturing high resolution images of the OB-1 lunar regolith simulant. Ideally, a DSLR camera mounted to a microscope would have been used for this task, however, one was not accessible. The camera was angled at approximately 45-degrees with overhead LED lighting. Approximately 3 to 4 grams of OB-1 simulant measuring approximately 5cm×5cm area and 2 to 3 mm depth was spread over a piece of parchment paper.

With this method, approximately 250 full-scale images were collected. About 2-3 grams of OB-1 lunar simulant was spread out over a small region of wax paper for each image. Once one image was captured, the sample was physically rotated, shaken, and/or a new sample of OB-1 was spread out and imaged. In this way, the images were varied. Due to the 45-degree angle, only a rectangular central region of the image was in focus, measuring approximately 500 pixel height and 1200 to 1800 pixels width (Figure 3.1). To prepare the data for the object detection model, the images were cropped to 500 pixels×500 pixels. This resulted in a dataset size of 863 square images.

![In-focus region of an image of sample of OB-1](image)

Figure 3.1: In-focus region of an image of sample of OB-1

It is important to note that the images are not accurate representations of the lunar surface, rather the best effort for replicating the conditions on Earth. Though the lunar regolith simulant
was created to simulate lunar regolith closely, the environment was not suitable for capturing these images. Additionally, the method with which the samples were laid on the parchment paper was not representative of the surface. A thicker layer of the simulant with more of the fine-grained dust may have helped to distribute the particles more as there were a larger proportion of the dark, glass/slag particles in the images sitting on top of the fine-grained material. Examples of actual lunar surface imagery will be discussed further in Chapter 4.

3.1.2 Data Labelling

To train the object detection model with a supervised learning approach, the dataset was manually labelled. The smallest visible particles in the OB-1 simulant images were approximately 0.3 mm in diameter. This spatial resolution was not sufficient to distinguish particles such as ones representing mineral fragments, or agglutinates. Within the samples, particles representing breccia and rock fragments looked very similar and not distinct enough to confidently distinguish. The most prominent particles were the glass particles represented by dark, black slag. For this reason, ‘glass’ was the one class used for labelling.

The first approach to labelling the images was to test a technique to isolate the glass particles. Since they were the darkest and most distinguishable particles against the background, the goal was to remove the background to see if the particles of interest would remain. First, the image contrast was shifted to the darker end of the image histogram to further darken the pixels consisting of the glass (Figure 3.2.a). Then, based on the output histogram, a threshold of 0.15 was selected for the background removal (Figure 3.2.b). The resulting background removal maintained many of the dark pixels relating to the glass particles (Figure 3.2.c) and finally, the image with the background removed was overlaid on the original image as a mask (Figure 3.2.d).

Though the background removal isolated many of the glass particles, there was a significant amount of error. The shadow regions in the images contributed to noise, ranging from small gaps between particles laying on top of others, to large regions consisting of shadows from rock fragments. Additionally, where glass fragments had reflections, the background removal
caused one glass piece with a large reflection to be split into two glass pieces. Despite adjusting
the contrast and threshold value, this problem could not be avoided without requiring additional
data processing to clean up noise associated to the shadows and reflections. This approach
would also not be easily translatable to other particle classes such as breccia fragments or
melts. As this process would be a heavily manual process to obtain fine outlines of objects, a
bounding box labelling approach was used instead to coarsely capture the important necessary
information relating to each object accurately.

The LabelImg python tool was used to label the images with bounding boxes. Particles
which were fully or partially visible were labelled while occluded or hard to distinguish par-
ticles were not. Per image, there were between 5 and 80 glass particles with varying sizes in
each image. There were no images in the dataset without an object in it. Many of the particles
were small, ranging from approximately 20 to 200 pixels large, and in certain images, many
particles were overlapping. In hindsight, this approach was flawed as it is a difficult task for
an object detector to learn with images that have a large number objects of interest especially
ones that are small and overlapping.
3.2 Model

3.2.1 Model Parameters

As previously mentioned, the lunar regolith simulant dataset was used as a proof of concept for object detection for microscale lunar surface imagery. Due to the small dataset, a transfer learning approach was used. The TensorFlow 2 Object Detection API has many pre-trained networks available which can be fine-tuned for use with a custom dataset to reduce training time and improve performance. Of the available models, the Faster RCNN with Resnet50 V1 backbone network was selected to provide a baseline. Faster R-CNN models are described to have high accuracy in object detection tasks while being slower than other models. This trade off is fine for the purpose of this work as time is not a constraint. The model configuration file was modified for one class and input image size of 500×500 pixels. The hyperparameters were set according to Table 3.1. These values were selected based on the default values available in the TensorFlow 2 Object Detection API for the Faster RCNN model. As this was a proof of concept, the values were used as is to provide a preliminary result.

Data augmentation is a powerful technique to increase the size of datasets, such as the OB-1 lunar regolith simulant dataset. Within the model configuration file, there are data augmentation options included such as random horizontal flips.

Gray scaling, changing contrast, and Gaussian blur are other techniques that were considered as all the images were quite uniform. Since the test data and training data would be from the same dataset with the same lighting conditions, this was not initially done to see how the model would first perform with the dataset as is.
Table 3.1: Hyperparameters for Faster RCNN with Resnet50 V1 model with OB-1 dataset

<table>
<thead>
<tr>
<th>Hyperparameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch Size</td>
<td>8</td>
</tr>
<tr>
<td>Number of Steps</td>
<td>25000</td>
</tr>
<tr>
<td>Learning Rate</td>
<td>0.04</td>
</tr>
<tr>
<td>Warm-up learning rate</td>
<td>0.01333</td>
</tr>
<tr>
<td>Optimizer</td>
<td>Momentum</td>
</tr>
<tr>
<td>Activation Function</td>
<td>Softmax</td>
</tr>
<tr>
<td>Training Set</td>
<td>90%</td>
</tr>
<tr>
<td>Test Set</td>
<td>10%</td>
</tr>
</tbody>
</table>

### 3.2.2 Proof of Concept Results

The model took approximately 3 hours to train and evaluate on an NVIDIA GeForce RTX 3090 GPU. The model was exported before performing inferences by loading an image of lunar simulant to see if the object detector could pick out the glass particles correctly. Figure 3.3 shows the original image labels compared against the model predictions. The model does reasonably well to predict particles on test data. However, it does not predict all the labels correctly.

Figure 3.3: Comparison of original, ground truth bounding boxes to predicted bounding boxes. Additional augmented images were tested to see how well the object detector would generalize.
3.2. Model

from the ground truth, and has a lower confidence score for some predictions.

(a) Plot of total loss generated during training and (b) Plot of mAP calculated on validation set for IOU validation. Orange = training, Blue = validation above 50%

Figure 3.4: Plots of loss and mAP of the Faster R-CNN ResNet50 model with the OB-1 dataset.

Different post-processing augmentations were performed to see how the model performs with varied data. Since data augmentation was not performed prior to training, variations in the test data showcase lower performance. The COCO metrics were used to evaluate the model during training and outputted to Tensorboard, TensorFlow’s visualization software for viewing the training job. The results show reasonably poor results (Fig. 3.4). This is likely due to multiple reasons such as:

- too many objects to detect per image
- not all objects labelled in each image
- overfitting due to lack of variation in data
- not enough data for training

The validation loss (Fig. 3.4a) increased during training and diverged from the training loss. This shows that the model overfitted to the training set, memorizing the results, and performed poorly on the validation set. This could be due to the learning rate being too high resulting in the optimizer pushing the model away from converging on a minimum value. However, a combination of the other highlighted potential problems is more likely the cause for poor
performance. Additionally, the mean average precision (mAP) settles around 44%, but seems to be decreasing as more training steps are taken. This could indicate that the model is worsening at detecting objects on the validation set as it is further trained due to the quality of the dataset. It is possible that the model is confused by the features of the glass particles compared to the background as there are many objects per image and dark shadows look similar to the glass.

Though the results were not as expected and did not provide a significant outcome, a great deal was learned from this proof of concept. This process helped with understanding the tools and the evaluation metrics for an object detection model using transfer learning applied to a geological dataset. A major issue with this project was that there was not enough data. Data augmentation would improve the size of the dataset, however, likely still would not have resulted in better performance. Labelling the total of approximately 800 images that were cropped and collected then augmenting them would have resulted in approximately a couple of thousand images. However, the process of labelling these images was an extremely lengthy and difficult process that was not prioritized. In Chapter 4 and 5, this procedure will be improved upon and applied to a dataset of in-situ lunar regolith from the Apollo missions. With the dataset not being representative of the lunar surface compared to the Apollo lunar regolith images, this model for the OB-1 dataset was not further improved upon, and the hyperparameters were not tuned.
Chapter 4

Lunar Regolith Dataset Methodology

This chapter outlines the methods for performing object detection with a dataset of microscopic lunar surface imagery. The dataset development, pre-processing, model selection, and results are detailed in this chapter. A significant amount of time was spent on developing the dataset for the object detection task and to understand the geological features within the microscale lunar imagery.

4.1 Apollo Lunar Close-up Camera (ALSCC) Dataset

4.1.1 Dataset Details

The Apollo Lunar Surface Close-up Cam (ALSCC) Lunar Regolith dataset was developed by the author for this research using images taken by the ALSCC instrument during the Apollo 11, 12, and 14 missions. The images captured by the handheld ALSCC instrument provide significantly high spatial resolution of the lunar surface regolith. The images from this instrument were a significant improvement over the OB-1 dataset, providing images of actual lunar regolith in-situ.
4.1.2 Data Collection

The ALSCC images for the Apollo 11, 12 and 14 missions were available on the *March to the Moon* online database. Since the ALSCC instrument was a stereoscopic camera, each scene captured has a stereo-pair image (Figure 4.1). Since there is a slight variation in colour and angle of the images in the pair, these duplicated images could be used to increase the number of images in the dataset. The quantity of images per mission and stereo-pair can be seen in Table 4.1. Some paired images are missing from the database for the second camera which is why there is an odd number of images for some cameras. The original images were downloaded from the database in raw TIF format to preserve the resolution. The images were padded with black bars as a result of being scanned from the original film and so, were cropped prior to processing them to remove the black bars. Each resulting image size was 2048 pixels $\times$ 2420 pixels.

![Figure 4.1: Example stereo-pair images from the Apollo 11 images (AS11-45-6709 A/B)](image)

Refer to Table 2.1 for the technical specifications of the ALSCC. In total, there were 86 high-resolution images. A smaller input image size was used to reduce the computational complexity, and to generate more training images by cropping the full-scale images. For this reason, the images were cropped to 512 pixels $\times$ 512 pixel images. For each of the original images, 20 cropped images were produced. The resulting dataset consisted of 1,720 images. The added benefit to this pre-processing was that the smaller image size made the task of
Table 4.1: Number of images for each mission

<table>
<thead>
<tr>
<th>Mission</th>
<th>Camera</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apollo 11</td>
<td>A</td>
<td>18</td>
</tr>
<tr>
<td>Apollo 11</td>
<td>B</td>
<td>17</td>
</tr>
<tr>
<td>Apollo 12</td>
<td>A</td>
<td>15</td>
</tr>
<tr>
<td>Apollo 12</td>
<td>B</td>
<td>15</td>
</tr>
<tr>
<td>Apollo 14</td>
<td>A</td>
<td>11</td>
</tr>
<tr>
<td>Apollo 14</td>
<td>B</td>
<td>10</td>
</tr>
</tbody>
</table>

labelling the particles easier in each image.

4.1.3 Data Labelling

As mentioned, for this work, a supervised learning approach was taken and required a fully labelled dataset. The task of identifying the particles within the lunar regolith images required professionals in the field of planetary geology to be consulted to support the identification of objects for the ground truth images. The classes that were identified from the ALSCC images were ‘glass’, ‘breccia’, and ‘melt’. ‘Glass’ consisted mainly of glass beads which are round with a shiny center due to the reflection from the light source, and generally small (Fig. 4.2a). ‘Breccia’ consisted mainly of rounded, irregular shaped fragments with varying speckles of different coloured materials embedded in them, ranging from half a centimeter to 2 centimeters in diameter, and sometimes larger (Fig. 4.2b). ‘Melt’ consisted of irregularly shaped, sometimes bubbly, shiny material that could be overlain on rocks or stand-alone melt particles, ranging from small fragments to covering larger regions (Fig. 4.2b). There are other objects in the images that were not labelled to first focus on a few broad classes. Mineral and rock fragments are two other classes that could have been considered, but since mineral and rock fragments can look similar to one another and to breccias, this would have been a more difficult labelling task for now.
Figure 4.2: Sample images of glass, melt, and breccia particles that were labelled in the ALSCC custom dataset.

Once again, the LabelImg open-source Python tool was used to label the images using Pascal VOC format bounding boxes. The process of labelling images in preparation for object detection requires consistency, accuracy, and to be exhaustive [11]. The ground truth labels were first identified with the assistance and expertise of planetary geologists Dr. Melissa Battler and Cosette Gilmour from Mission Control. This process was lengthy, requiring multiple passes viewing the images close-up, viewing the images with different perspectives, understanding the context of the images, and discussing the possible labels to be assigned to the visible particles. Criteria were defined for each class to assist in making sure the labelling procedure was consistent and as accurate as possible.

Objects were labelled in the images if there was high confidence of the ground truth label. Since the images were captured on the Moon where the environment results in many particles to be covered in a thin layer of dust, many particles were partially or fully occluded, making it difficult to distinguish features between different particles. For example, since ‘glass’ and ‘melt’ are both shiny and reflective in similar ways and can be of similar size, one would have
to rely on trying to distinguish the shape of the particles and the context of the scene to make a
decision on the ground truth label. If edges were occluded, this task was difficult, and particles
would not be labelled. The anaglyphs, generated by the stereo-pair images, could be viewed
to provide depth information of the particle features by wearing 3D glasses. This was helpful
in understanding if particles were on top of one another or next to each other. It also helped
to see the shapes of the particles better, for example, making it easier to see the rounded shape
of glass beads. Additionally, there were published reports detailing the results from the Apollo
missions that provided additional context which helped when identifying the different particles
in the images [1][19][51]. Moreover, breccias can have embedded glass and melt fragments,
however, they were not distinctly labelled.

While context of the full-scale scene and anaglyphs were not used in their full form to train
the model developed, they were mainly used to help ensure the particles were labelled with
confidence by the planetary scientists. The goal was to be consistent in the way objects were
labelled to see if the model could identify objects with information that was available in the
images used in training. For example, the anaglyphs are helpful for a human to distinguish
particles, however, the model might be able to identify the particles based on the shadows from
the single image in the stereopair.

The images were first labelled after discussion with the team and using annotations on
the original-sized, high-resolution images (see Figure 4.3). Once the ground truth labels were
decided, bounding box labels were assigned to the cropped images while referring to the full-
size images using the LabelImg tool. It is important to note that some objects were not labelled
while going through the full-size images because they belonged to an unknown class.

Overall, 813 ‘glass’, 479 ‘breccia’, and 506 ‘melt’ objects were identified. Since the ‘glass’
particles were much smaller, consisting of fewer pixels than ‘melt’ or ‘breccia’ fragments, the
larger number of identified ‘glass’ particles was beneficial for balancing the classes. However,
additional work could be considered to ensure that the size of labels and number of each class
could be more equalized to ensure the model performs well for all classes. For each of the
three Apollo missions, Table 4.2 details the number of objects per class that were labelled.
Figure 4.3: Example of object ground truth labels on the full-size images (AS11-45-6708A). Red = melt, Green = glass, Blue = breccia.

<table>
<thead>
<tr>
<th>Mission</th>
<th>Glass</th>
<th>Breccia</th>
<th>Melt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apollo 11</td>
<td>337</td>
<td>321</td>
<td>392</td>
</tr>
<tr>
<td>Apollo 12</td>
<td>330</td>
<td>91</td>
<td>111</td>
</tr>
<tr>
<td>Apollo 14</td>
<td>146</td>
<td>67</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 4.2: Number of objects per class for the ALSCC dataset

There were more images from the Apollo 11 dataset, and in general, more variation in the images. There were relatively balanced numbers of each object identified from the Apollo 11 images. The Apollo 12 images had less large particles present. There were a larger proportion of small, glass particles. Many of the images had objects that were occluded and covered in a layer of dust making it difficult to identify breccia particles. Apollo 14 had the smallest number of images. The images were also not as detailed. There were numerous glass particles present once again, however, very few melt particles. Glass and melt particles share similar characteristics, however, based on the criteria set out at the beginning of labelling, there were less melt particles that fit the description among the Apollo 14 images.
Of the 1,720 total images in the dataset, 908 of the images had at least one object in them while 812 images were unlabeled as they did not contain any objects in them. To improve the dataset and ensure there were more objects per class available for training, the dataset was augmented using image processing techniques.

### 4.2 Object Detection Model

The models were trained on a NVIDIA GeForce RTX 3090 GPU. Due to the high-computational cost of transfer learning, a GPU was necessary to efficiently train the models.

#### 4.2.1 Data Augmentation

Augmenting the dataset was important to increase the size, and improve variation in the image data. The `imgaug` Python library was used to perform the task. Each image in the training set underwent a randomized set of augmentations. As the images were transformed, the bounding boxes were transformed as well. The selected augmentations were:

- horizontal flips
- vertical flips
- rotations (90 to 270 degrees)
- additive Gaussian blur

The shadows on the Moon can result in confusion about the perspective of the objects in the image. Shadow regions can look like craters, and a particle jutting out can look sunken into the surface depending on the relative frame at which one is looking at an image. The horizontal and vertical flips, as well as rotations were meant to help the model train on the different possible perspectives that could be present in the lunar surface imagery.

Gaussian blur is a common augmentation that is performed on machine learning datasets. Blurring helps introduce imperfections in the images to help the model generalize on the dataset.
better. Many of the images have regions of blur due to the focus on the camera. If a rover were capturing images on the lunar surface, the images could be blurry due to movement while capturing or having blur introduced in the background due to the focus.

The images in the ALSCC dataset had varying contrasts and lighting conditions due to the nature of how the images were captured. The astronauts would capture images of different areas, sometimes within a crater or near the lunar lander vehicle, so, the lighting conditions would vary. For this reason, there did not seem to be a need to augment the dataset by adding images with different lighting and contrast. Shearing was also not used to augment the images. Shearing is the process of stretching and warping the image slightly. This distorts the objects in the images. Since the shape of the edges and the size of the objects in the image were important features identified by the team when labelling, shearing would cause unnatural distortions to the image. For this reason, the images did not undergo shearing.

After performing the augmentations, the dataset increased from 1,720 images to 6,260 images. Augmentations were only performed on images which had objects labelled within them. This was to ensure that more samples of the objects would be available for training rather than the unlabeled, background images. The augmentations were performed on the training data as well as the test data. This is not necessary for the testing set, however as mentioned, images of the lunar surface have varying perspectives and degrees of clarity, so it was important to see how the model would perform on augmented images in the test set.

4.2.2 Train, Test, Validation Split

The dataset was first split into training, testing, and validation sets. A split of 80% for training, 10% for testing, and 10% for validation was used. Generally, the training set is the only set that is augmented. However, as the dataset is small, the validation and test sets were also augmented to increase the quantity and variation of images that the model could be evaluated and tested on.

Table 4.3 details the number of objects in each set. As one can see, the augmentation increases the instances of each object class in the different sets to help with providing the
model more examples to train and test with.

Table 4.3: Number of objects per class before and after augmentation

<table>
<thead>
<tr>
<th>Class</th>
<th>Train</th>
<th>Validation</th>
<th>Test</th>
<th>Train (aug)</th>
<th>Validation (aug)</th>
<th>Test (aug)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glass</td>
<td>649</td>
<td>83</td>
<td>79</td>
<td>3894</td>
<td>498</td>
<td>474</td>
</tr>
<tr>
<td>Breccia</td>
<td>387</td>
<td>49</td>
<td>45</td>
<td>2322</td>
<td>294</td>
<td>270</td>
</tr>
<tr>
<td>Melt</td>
<td>383</td>
<td>60</td>
<td>63</td>
<td>2298</td>
<td>360</td>
<td>378</td>
</tr>
</tbody>
</table>

It is also important to understand how the different scales of the objects in each class impact the training and model performance. In the COCO Evaluation Metrics API, which will be discussed in Chapter 5, set scales are used for evaluating the metrics for different sized objects: small ($32^2$ pixels), medium ($32^2 - 96^2$ pixels), and large ($96^2$ pixels). Overall, there was a higher proportion of glass particles in the dataset, almost twice as much over the other classes. As Table 4.4 shows, the majority of glass particles are in the small scale under $32^2$ pixels, while the majority of the melt and breccia particles are in the medium scale between $32^2$ and $96^2$ pixels. This higher number of small scale glass particles helps to balance the number of pixels related to the class for the model to learn. However, there are still a significantly higher amount of melt and breccia particles for the model to learn, and this imbalance could impact the models’ ability to recognize small, glass particles.

Overall, there are fewer large particles about $96^2$ which makes sense as the particles on the Moon are generally small. Breccia fragments tend to be larger while melt particles vary quite a bit in size.

4.2.3 Model Selection

As mentioned, the TF2 Object Detection API has various pre-trained models available for use built on different architectures. The TF2 Model Zoo is an online database that holds the pre-trained models. The models that were considered for this work were Faster R-CNN with ResNet50 backbone, Faster R-CNN with ResNet101 backbone, Faster R-CNN with ResNet152
Table 4.4: Number of objects per scale in the training, validation and testing sets, and per object class (after augmentation)

<table>
<thead>
<tr>
<th>Scale</th>
<th>Train</th>
<th>Validation</th>
<th>Test</th>
<th>Glass</th>
<th>Breccia</th>
<th>Melt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small (32² pixels)</td>
<td>390</td>
<td>595</td>
<td>649</td>
<td>4710</td>
<td>198</td>
<td>636</td>
</tr>
<tr>
<td>Medium (32² – 96² pixels)</td>
<td>594</td>
<td>444</td>
<td>414</td>
<td>156</td>
<td>1956</td>
<td>1980</td>
</tr>
<tr>
<td>Large (96² pixels)</td>
<td>175</td>
<td>120</td>
<td>96</td>
<td>0</td>
<td>732</td>
<td>420</td>
</tr>
</tbody>
</table>

Table 4.5: Pre-trained models available on TF2 Model Zoo that were considered

<table>
<thead>
<tr>
<th>Model Name</th>
<th>Inference Speed (ms)</th>
<th>COCO mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Faster R-CNN ResNet50 V1 640x640</td>
<td>53</td>
<td>29.3</td>
</tr>
<tr>
<td>Faster R-CNN ResNet101 V1 640x640</td>
<td>55</td>
<td>31.8</td>
</tr>
<tr>
<td>Faster R-CNN ResNet152 V1 640x640</td>
<td>64</td>
<td>32.4</td>
</tr>
<tr>
<td>SSD ResNet50 V1 FPN 640x640 (RetinaNet50)</td>
<td>46</td>
<td>34.3</td>
</tr>
<tr>
<td>SSD ResNet101 V1 FPN 640x640 (RetinaNet101)</td>
<td>57</td>
<td>35.6</td>
</tr>
</tbody>
</table>

backbone, SSD with ResNet50 backbone, and SSD with ResNet101 backbone. These architectures were selected to compare the accuracy and speed of predictions for the Faster R-CNN against SSD for this object detection task. Table 4.5 details the models that were considered and the approximate time (in milliseconds) for an inference to be performed on one image and the mean average precision (mAP) of the model tested on the COCO dataset. Another model that was considered but not used was the Faster R-CNN with Inception backbone. Though it has high accuracy, it is much larger due to the Inception backbone and computationally expensive to effectively train.

For the purposes of this research, accuracy for identifying objects of different scales, especially smaller objects such as the glass beads and fragments of melt and breccia, is more important than speed. A model like this could be deployed on Earth to test images which are down linked from the Moon, meaning testing time and computational power is less of a constraint.
4.2. Object Detection Model

It is important to note that though the input image of the models selected size was 640 pixels×640 pixels, the input image size was reduced to 512 pixels×512 pixels to match the size of the images in the custom datasets. The smaller input image size would, in theory, be helpful to reduce the amount of time for training.

4.2.4 Configuring the Model

Within the TF2 Object Detection API, the selected models were configured to fine-tune pre-trained models on the custom dataset. Many of the default settings were kept the same to first test and understand which parameters needed to be tuned. Parameters such as batch size and input image size were modified. Since pre-trained models were used, only the feature layer was modified to train the model to detect objects based on the custom dataset. The models were set up with built-in feature extractors. By providing labelled images, the object detector extracted features automatically that were of importance for each class.

The model parameters were kept similar for all the models to make comparison easier and to identify parameters to tune. The parameters were set according to Table 4.6.

<table>
<thead>
<tr>
<th>Model</th>
<th>Faster R-CNN ResNet</th>
<th>SSD ResNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch size</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Number of Steps</td>
<td>25000</td>
<td>25000</td>
</tr>
<tr>
<td>Optimizer</td>
<td>Momentum, 0.9</td>
<td>Momentum, 0.9</td>
</tr>
<tr>
<td>Warm-up Learning Rate</td>
<td>0.01333</td>
<td>0.01333</td>
</tr>
<tr>
<td>Learning Rate</td>
<td>$4 \times 10^{-2}$</td>
<td>$4 \times 10^{-2}$</td>
</tr>
<tr>
<td>Activation Function</td>
<td>Softmax</td>
<td>Sigmoid</td>
</tr>
</tbody>
</table>

Once the model configurations were updated and the train, test, and validation sets were formatted correctly for object detection, the models could be trained with the evaluation running simultaneously and viewed on Tensorboard. Based on the results from the model evaluation,
hyperparameters could be tuned to try to improve performance.
Chapter 5

Results and Discussion

This chapter outlines the results for the object detection models tuned with the Apollo Lunar Surface Close-up Camera dataset. The models were evaluated during training using the COCO evaluation metrics API and viewed on Tensorboard. This chapter will go over the results from the evaluation, show examples of the models’ performance on the test set, and provide insight into the best performing models. Of the models selected, the SSD ResNet50 performed the best against the SSD ResNet101, and the Faster R-CNN ResNet101 performed the best against the Faster R-CNN architectures. These models underwent hyperparameter tuning to try to reduce the loss and improve performance.

5.1 Model Evaluation

For object detection, the most popular metrics for understanding how well the model generalizes are the mean average precision (mAP) and the average recall (AR). These metrics help to understand how well the model performs the two tasks of localization and classification. For this work, accurate classification and localization is important to optimize for, and so, the mAP and AR are used to evaluate the model. Additionally, the loss value details how well the model converges during training and with unseen data.

As mentioned, the parameters for the models were kept the same to compare and identify
which models would be suitable to tune (Table 4.6). The training time varied for the selected models with different architectures. The training time is referred to in Table 5.1.

Table 5.1: Summary of Model Training Time

<table>
<thead>
<tr>
<th>Model</th>
<th>Time (hrs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Faster R-CNN ResNet50</td>
<td>6.83</td>
</tr>
<tr>
<td>Faster R-CNN ResNet101</td>
<td>7.62</td>
</tr>
<tr>
<td>Faster R-CNN ResNet152</td>
<td>10.25</td>
</tr>
<tr>
<td>SSD ResNet50</td>
<td>5.37</td>
</tr>
<tr>
<td>SSD ResNet101</td>
<td>6.68</td>
</tr>
</tbody>
</table>

5.1.1 Mean Average Precision (mAP) Evaluation

The mAP metric compares the accuracy of a predicted object to the ground truth. In object detection, it is the most significant metric and commonly used. It considers the impact of false positives and false negatives as it calculates the AP over different recall values and number of classes. A high mAP value closer to 1 means the model is able to positively detect objects well with a low number of false detections (false positives and false negatives). Predictions with intersection over union (IOU) over 50% (mAP@IOU:0.5) are traditionally used as a standard metric for the mAP. While it is good to have predictions with IOU of over 75% (mAP@IOU:0.75), there are less of these represented generally since the box location and size can vary. For each of the models, the mAP evaluation is detailed in Table 5.2.

All models were able to achieve mid-range results for mAP for IOU over 50%. The SSD ResNet50 and Faster R-CNN ResNet101 architectures were the two highest performing models achieving 53.2% mAP and 49.6% mAP respectively. The models all performed relatively poorly for small scale objects, and better for large scale objects. Overall, higher mAP values would indicate better performance, but the impact of false predictions lead to a significant performance drop. The false predictions are likely attributed to the quality of the data labels,
and this will be discussed further in this chapter. In general, object detection models perform poorly on small object detection [54].

As noted in Table 4.4 from Chapter 4 of this thesis, there are a larger proportion of small scale objects which consist of glass particles, and the majority of the breccia and melt particles are of medium scale. For this reason, we are interested in the mAP of detections for these scales. The SSD ResNet50 and SSD ResNet101 models perform relatively better for small scale objects over the Faster R-CNN models, with 17.36% and 11.3% mAP respectively. This is still a fairly small value and indicates that the model has a hard time accurately detecting small objects such as the glass particles. For medium scale objects, the SSD ResNet50 and Faster R-CNN ResNet101 models have mAP scores of 40.5% and 29.4%. The Faster R-CNN ResNet50 is close behind with an mAP for medium scales of 29.0%. It is clear that models are better at detecting large scale objects, likely due to the greater number of pixels associated to each object for training, with Faster R-CNN ResNet101 and SSD ResNet50 achieving 50.0% and 44.9% mAP respectively.

The numbers in Table 5.2 may look low compared to metrics for classical machine learning where accuracy in the 90-100% range is considered high performance. To achieve an mAP of closer to 100%, the model would need to have nearly perfect precision and recall for all IOU thresholds and across all classes, which is not reasonable to expect. This is highly dependent on the dataset and the current state-of-the-art models being used for the task. To help understand the perspective, consider Table 4.5 which lists the mAP for each of the models selected. The mAP score is relatively low, ranging around ~ 30%. For example, He et al. reported an mAP of 48.4 for baseline Faster R-CNN ResNet101 using the COCO evaluation set [18]. Generally, a mAP around 70% is considered good, depending on the data. While the values are still considered low for this work, the justification and possible strategies for improvement will be discussed later in this chapter.

Additional plots of the mAP for all models can be found in Appendix A.
Table 5.2: Summary of Models: mAP after 25k steps

<table>
<thead>
<tr>
<th>Model</th>
<th>mAP@IOU:0.5</th>
<th>mAP@IOU:0.75</th>
<th>Total mAP (@0.5:0.05:0.95)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>(all scales / large / medium / small)</td>
</tr>
<tr>
<td>Faster R-CNN ResNet50</td>
<td>46.6%</td>
<td>17.0%</td>
<td>21.3% / 45.1% / 29.0% / 3.8%</td>
</tr>
<tr>
<td>Faster R-CNN ResNet101</td>
<td>49.6%</td>
<td>21.2%</td>
<td>24.1% / 50.0% / 29.4% / 8.3%</td>
</tr>
<tr>
<td>Faster R-CNN ResNet152</td>
<td>44.6%</td>
<td>16.2%</td>
<td>20.9% / 49.3% / 20.6% / 6.3%</td>
</tr>
<tr>
<td>SSD ResNet50</td>
<td>53.2%</td>
<td>22.3%</td>
<td>26.4% / 44.9% / 40.5% / 17.36%</td>
</tr>
<tr>
<td>SSD ResNet101</td>
<td>38.3%</td>
<td>12.5%</td>
<td>17.4% / 32.0% / 22.7% / 11.3%</td>
</tr>
</tbody>
</table>

5.1.2 Average Recall (AR) Evaluation

The average recall (AR) represents the proportion of true positive detections averaged over the number of classes and IOU thresholds between 0.5 to 1 calculated for maximum detections per image of 1, 10, and 100. Table 5.3 details the AR values after the models were evaluated. Since high recall, the detector’s ability to make correct predictions, is also important, this metric is used to evaluate the model. Overall, the SSD ResNet50 and SSD ResNet101 models have high AR values of 51.8% and 49.4% respectively. They perform better than the Faster R-CNN ResNet architectures across all scales and number of detections per image. The Faster R-CNN ResNet101 model has the best performance from its class of models, with an AR for maximum 100 detections for all scales of 44.4%. Like the mAP values, the AR should be higher to indicate that the model is performing well on new, unseen data, and the impact of false predictions is low.

Additional plots of the AR for all models can be found in Appendix B.
5.1. Model Evaluation

Table 5.3: Summary of Model Performance: AR after 25k steps

<table>
<thead>
<tr>
<th>Model</th>
<th>AR@100 / @10 / @1</th>
<th>AR@100 (large)</th>
<th>AR@100 (medium)</th>
<th>AR@100 (small)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Faster R-CNN ResNet50</td>
<td>42.2% / 36.0% / 22.5%</td>
<td>65.6%</td>
<td>48.6%</td>
<td>29.7%</td>
</tr>
<tr>
<td>Faster R-CNN ResNet101</td>
<td>44.4% / 38.8% / 24.7%</td>
<td>63.5%</td>
<td>47.8%</td>
<td>31.2%</td>
</tr>
<tr>
<td>Faster R-CNN ResNet152</td>
<td>41.2% / 36.0% / 22.0%</td>
<td>65.1%</td>
<td>38.7%</td>
<td>28.4%</td>
</tr>
<tr>
<td>SSD ResNet50</td>
<td>51.8% / 47.2% / 24.2%</td>
<td>67.2%</td>
<td>59.2%</td>
<td>43.9%</td>
</tr>
<tr>
<td>SSD ResNet101</td>
<td>49.4% / 42.6% / 20.2%</td>
<td>67.2%</td>
<td>58.2%</td>
<td>40.0%</td>
</tr>
</tbody>
</table>

5.1.3 Loss Evaluation

Figure 5.1 details the total loss for the selected models. In deep learning, a low value of loss closer to zero is ideal. A low value for training loss means that the model was able to reach a minimum and learn the training set well. If validation loss converges to a low value as well, this means the model was able to generalize well for new, unseen data. If the validation loss diverged, or was much higher than the training loss, this would indicate that the model was overfitting on training data, basically memorizing the training set, resulting in being unable to make accurate predictions on new data.

The training loss for the models was overall generally low meaning the model was able to successfully converge at a minimum value (Fig. 5.1). The validation loss varied and tended to be higher than the training loss for many instances, indicating that the model could be overfitting. However, the difference between the training and validation loss is small and during evaluation, and both training and validation loss decreased towards a minimum. Since the val-
ues are low, under approximately 1 for both training and validation, this indicates that there is low error in the predicted and ground truth bounding boxes, and does not necessarily mean the model is overfitting.

During evaluation of the Faster R-CNN architecture models, as the loss decreased towards a minimum, the mAP value began to saturate towards a maximum. This could indicate that a lower number of steps, or overall epochs, could be used to obtain similar results, and overall reduce training time. This occurred around approximately 16k steps. For the SSD ResNet architecture, the opposite was true. As the model was evaluated and the loss approached a minimum value, the mAP seemed to continue to increase even after 25k steps. This could indicate that more steps may be needed to result in the model achieving a high overall mAP with a low loss value.

In Section 5.2, these hypotheses will be tested to understand whether adjusting the parameters such as learning rate and number of steps improves performance.

Additional plots of the loss for all models can be found in Appendix C.
Table 5.4: Summary of Models: Loss after 25k steps. All losses are for validation unless otherwise stated.

<table>
<thead>
<tr>
<th>Model</th>
<th>Total Loss Train / Validation</th>
<th>Classification Loss</th>
<th>Localization Loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>Faster R-CNN ResNet50</td>
<td>0.0379 / 0.6017</td>
<td>0.1831</td>
<td>0.0698</td>
</tr>
<tr>
<td>Faster R-CNN ResNet101</td>
<td>0.0152 / 0.5299</td>
<td>0.1436</td>
<td>0.0533</td>
</tr>
<tr>
<td>Faster R-CNN ResNet152</td>
<td>0.0225 / 0.4727</td>
<td>0.135</td>
<td>0.0543</td>
</tr>
<tr>
<td>SSD ResNet50</td>
<td>0.4408 / 0.8692</td>
<td>0.4892</td>
<td>0.2274</td>
</tr>
<tr>
<td>SSD ResNet101</td>
<td>0.9022 / 1.207</td>
<td>0.253</td>
<td>0.3809</td>
</tr>
</tbody>
</table>

5.1.4 Inferences on Test Data

After training and evaluating the selected models, the models were exported and prepared for evaluating the performance on the test set of unseen image data. The models were tested on an Intel UHD Graphics 630 integrated GPU. Each model had varying inference time, which is shown in Table 5.5. As expected, the models with ResNet50 were faster than architectures with a higher number of layers. Since time is not a constraint, these results are more for interest than they are to evaluate models that are suitable for use.

Following from the discussion of the results for the mAP and AR, the models were evaluated with the test set. Figure 5.2 shows a sample image with only melt labelled in the ground truth image, and the performance of each model for the object detection task. The Faster R-CNN ResNet101 and Faster R-CNN ResNet152 models predict the bounding boxes for melt correctly with 100% confidence. The Faster R-CNN ResNet50 model is able to locate the melt in the image correctly, however, also makes false positive predictions for of other background
### Table 5.5: Inference time for selected models

<table>
<thead>
<tr>
<th>Model</th>
<th>Inference Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Faster R-CNN ResNet50</td>
<td>~2–3</td>
</tr>
<tr>
<td>Faster R-CNN ResNet101</td>
<td>~2–3</td>
</tr>
<tr>
<td>Faster R-CNN ResNet152</td>
<td>~3–5</td>
</tr>
<tr>
<td>SSD ResNet50</td>
<td>~1–2</td>
</tr>
<tr>
<td>SSD ResNet101</td>
<td>~2–4</td>
</tr>
</tbody>
</table>

Figure 5.2: Model predicted bounding boxes for ‘melt’ using test images (cropped portion of AS11-45-6709B image). Green = glass, blue = breccia, and teal = melt.

objects in the image, predicting the presence of breccia and glass. The SSD ResNet50 and SSD ResNet101 models were able to accurately detect melt as well, however with confidence of 73–85%. The SSD ResNet50 model made false positive predictions of breccia and predicted
5.1. Model Evaluation

Figure 5.3: Model predicted bounding boxes for ‘melt’ and ‘glass’ using test images (cropped portion of AS11-45-6701A image). Green = glass, blue = breccia, and teal = melt.

A piece of melt as possibly glass with low confidence. The SSD ResNet101 model predicted multiple regions within the predicted melt regions as additional melt with lower confidence around 33–42% confidence. Breccia was also detected with a low confidence of 36%. The Faster R-CNN ResNet101 and Faster R-CNN ResNet152 models were the best performing for this image over the other models. However, the false positive predictions of the breccia object may not be incorrect. It was not labelled initially due to low confidence during the ground truth labelling task, it may be a significant object that requires a second look.

Figure 5.3 is a sample test image with a few melt particles, a breccia fragment, and multiple small glass particles. All models performed poorly for detecting glass. The SSD ResNet50 and Faster R-CNN ResNet152 models were able to identify one of the glass particles in the image.
Figure 5.4: Model predicted bounding boxes for ‘melt’ and 'breccia' using test images (cropped portion of AS11-45-6700A image). Green = glass, blue = breccia, and teal = melt.

each, however, only with 33% confidence. The Faster R-CNN models accurately identified the breccia particle, but only one of the melt particles. The SSD ResNet50 model identified the breccia particle with low confidence, and made two predictions for one melt particle with low confidence. Similarly, the SSD ResNet101 localized a box around the breccia particle, however, incorrectly labelled the particle, and like the SSD ResNet50, made two predictions for one of the melt particles with low confidence.

Figure 5.4 is a sample test image with a couple melt particles, and a breccia fragment. The image has been augmented, flipped horizontally to test how well the model can generalize on the data when the perspective has changed. The SSD ResNet50 and SSD ResNet101 models made predictions for the objects with low confidence. Generally, they are able to localize ob-
subjects of significance, however, are not able to resolve for the correct label with high confidence. The predictions are also focussed on the bottom half of the image, not identifying the lighter, breccia particle in the top-left. The Faster R-CNN architectures are able to accurately identify the center-left melt particle. However, the Faster R-CNN ResNet50 model incorrectly identified a breccia fragment with 100% confidence. The Faster R-CNN ResNet101 model identified the melt particles correctly with high confidence, however, incorrectly identified the same incorrect breccia object, but with a lower confidence of 55% than the Faster R-CNN ResNet50. The Faster R-CNN ResNet152 performed the best, identifying the melt fragments with 100% confidence, and identifying the breccia in the top-left with high confidence. It does however falsely identify a glass particle with medium confidence. It is possible that the false detections the models picked up could be true positive, but not labelled, due to the difficulty of confidently identifying some particles.

Furthermore, it is clear that the detector has a hard time detecting small, glass particles, performs decently well for breccia fragments, and very well for melt particles. The glass particles tend to be missing some crucial features relating to their shape in the images which make it hard for the human eye to distinguish particles, let alone the object detector. The glass particles most prominent features are the shiny reflection, which can be confused with melt and also can be detected as embedded within breccia, both predictions that are possibly valid. A small amount of glass particles and possible false labelling due to human error of the glass particles are likely for the low evaluation results for this class. For this reason, it is understandable that the model performs poorly on the glass class. It is possible that the problem of glass detection may need to be addressed separately from larger, more distinct objects like melt and breccia, since small object detection is a known problem in this field.

5.2 Hyperparameter Tuning

Hyperparameter tuning can be done to improve model performance after model evaluation. The approach to hyperparameter tuning was to adjust one parameter and maintain the rest.
This way, each modification could be done in isolation to see how it improved the overall performance before combining with other parameter tuning. It is important to note that combining tuned hyperparameters should be further tested and adjusted before implementation as they may improve performance when isolated but may degrade performance when combined. The Faster R-CNN ResNet101 and SSD ResNet50 models were selected for tuning due to their high evaluation performance against other models with the same architecture.

5.2.1 Tuning Learning Rate

From the results, the loss cost between the evaluation loss and training loss could indicate that the models were overfitting. The Faster R-CNN ResNet101 model validation loss looked to be slightly diverging from the training loss. The SSD ResNet50 model validation loss converged well with the training loss, however, the mAP seemed to continue increasing though the loss was just about saturating at a minimum.

The first hypothesis was that the models could be improved by reducing the learning rate. For this reason, a few experiments were run with lower learning rates to see how the models performed. The learning rate was reduced by a factor of 10 and a factor of 100, and the model was run for the full 25k steps. Both models used a cosine decay learning rate, with $1.333e^{-2}$ warm-up learning rate followed by $4e^{-2}$ learning rate.

Figure 5.5 shows the mAP and loss functions over the training and evaluation time over 25k steps. Table 5.6 and 5.7 show details of the mAP, AR and training time for the Faster R-CNN ResNet101 and SSD ResNet50 models.

For the first trial, the learning rate was reduced to $4e^{-3}$ with $1.333e^{-3}$ warm-up learning rate. The mAP of the Faster R-CNN ResNet101 model did not significantly improve for confidence scores over 50% or 75%. Detections for small objects saw an increase of 2%, and medium scales saw an increase of approximately 8%. This result was promising since as previously noted, most of the glass particles are small scale, and most of the melt and breccia particles are medium scale. Seeing improvements for these metrics was good.

For the SSD ResNet50, the mAP value did not significantly improve either, though showed
improvement for overall mAP of 2.6%. The mAP for small and medium scales reduced by 1.76% and 1.4% respectively. The mAP for large scale improved by 7.6%, pushing the mAP for all scales to improve by 1.2%. Overall, this is not a significant improvement, however, the mAP value curved towards a maximum, so this is an improvement with the higher learning rate.

For the second trial, the learning rate was reduced to $4e^{-4}$ with $1.333e^{-4}$ learning rate. The results of tuning this parameter were not better than the first results, and show that the $4e^{-3}$ learning rate is the better value to be used for learning rate for both the Faster R-CNN ResNet101 and SSD ResNet50 models.

### 5.2.2 Tuning Number of Steps

As mentioned, a hypothesis for why the mAP for the SSD architectures seemed to not have yet converged at a maximum could have been because training had not concluded yet. Additionally, the Faster R-CNN ResNet101 seemed to have converged at a minimum loss and maximum mAP earlier than 25k steps. As such, adjustments to the number of steps was made to see how the models’ performance would change. The learning rate was kept the same as the original run, at $4e^{-2}$ with warm up learning rate of $1.333e^{-2}$ to see how only changing the number of steps would impact performance.

Figure 5.5: Plots of loss and mAP for Faster R-CNN ResNet101 and SSD ResNet50 for tuned learning rates of $4e^{-3}$ and $4e^{-4}$.
Table 5.6: Summary of Faster R-CNN ResNet101 performance after tuning learning rate.

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Learning Rate</th>
<th>Learning Rate</th>
<th>Learning Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$4e^{-2}$ (original)</td>
<td>$4e^{-3}$</td>
<td>$4e^{-4}$</td>
</tr>
<tr>
<td>Training Time</td>
<td>7.62</td>
<td>9.37</td>
<td>8.55</td>
</tr>
<tr>
<td>(hours)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mAP @0.5 / @0.75</td>
<td>49.6% / 21.2%</td>
<td>49.7% / 19.1%</td>
<td>43.4% / 12.7%</td>
</tr>
<tr>
<td>mAP all / large / medium / small</td>
<td>24.1% / 50.0% / 29.4% / 8.3%</td>
<td>23.3% / 43.4% / 37.0% / 10.3%</td>
<td>18.6% / 41.3% / 33.9% / 3.9%</td>
</tr>
<tr>
<td>AR@100 all / large / medium / small</td>
<td>44.4% / 63.5% / 47.8% / 31.2%</td>
<td>44.0% / 63.1% / 50.7% / 35.3%</td>
<td>42.8% / 58.3% / 50.5% / 30.9%</td>
</tr>
</tbody>
</table>

Table 5.7: Summary of SSD ResNet50 performance after tuning learning rate.

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Learning Rate</th>
<th>Learning Rate</th>
<th>Learning Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$4e^{-2}$ (original)</td>
<td>$4e^{-3}$</td>
<td>$4e^{-4}$</td>
</tr>
<tr>
<td>Training Time</td>
<td>5.37</td>
<td>5.92</td>
<td>5.60</td>
</tr>
<tr>
<td>(hours)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mAP @0.5 / @0.75</td>
<td>53.2% / 22.3%</td>
<td>55.8% / 23.3%</td>
<td>52.8% / 22.1%</td>
</tr>
<tr>
<td>mAP all / large / medium / small</td>
<td>26.4% / 44.9% / 40.5% / 17.36%</td>
<td>27.6% / 52.5% / 39.1% / 15.6%</td>
<td>26.2% / 49.9% / 37.3% / 14.9%</td>
</tr>
<tr>
<td>AR@100 all / large / medium / small</td>
<td>51.8% / 67.2% / 59.2% / 43.9%</td>
<td>47.3% / 65.8% / 56.0% / 33.1%</td>
<td>47.7% / 63.9% / 52.5% / 34.6%</td>
</tr>
</tbody>
</table>

steps effected the model performance.

The SSD ResNet50 model was adjusted to have 30k and 50k steps. After 30k steps, from Figure 5.6, we can see that the mAP was still rising, was lower than the original run with 25k steps, and the loss was quite high. This was an odd outcome, as it was expected that the trend would continue and the mAP would be expected to rise. After 50k steps, the model looked
as though it was beginning to saturate. From Table 5.8, despite the increase to 50k steps, one can see that the loss was still similarly as low, and the mAP remained relatively the same, in fact seemingly reduced across all scales except for an approximate 3% increase in accuracy for large scales. It was also a significantly longer training time, approximately 9 hours longer than the 25k steps. This shows that there is likely no need to train the SSD for more epochs to achieve better performance.

![Figure 5.6: Plots of loss and mAP for Faster R-CNN ResNet101 and SSD ResNet50 after tuning the number of steps.](image)

The number of steps or the Faster R-CNN ResNet101 model were reduced from 25k to 16k steps to see if the model achieved similar performance with less training steps. From Table 5.9, across the board, there were negligible to slight increases in all metric values with the lower number of steps. Additionally, the loss remained similarly as low as with the higher number of steps meaning that the error was still low (Figure 5.6). With the benefit of lower training time, the lower number of steps would be good to use for the Faster R-CNN network.
Table 5.8: Summary of SSD ResNet50 performance after tuning the number of steps.

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Number of Steps</th>
<th>Number of Steps</th>
<th>Number of Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>25k (original)</td>
<td>30k</td>
<td>50k</td>
</tr>
<tr>
<td>Training Time</td>
<td>5.37</td>
<td>6.18</td>
<td>14.62</td>
</tr>
<tr>
<td>(hours)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mAP @0.5 / @0.75</td>
<td>53.2% / 17.7%</td>
<td>50.9% / 14.5%</td>
<td>53.1% / 22.3%</td>
</tr>
<tr>
<td>mAP all / large / medium / small</td>
<td>26.4% / 44.9% / 40.5% / 17.36%</td>
<td>24.0% / 44.7% / 34.4% / 7.8%</td>
<td>26.2% / 48.0% / 35.4% / 15.7%</td>
</tr>
<tr>
<td>AR@100 all / large / medium / small</td>
<td>51.8% / 67.2% / 59.2% / 43.9%</td>
<td>51.0% / 66.1% / 59.6% / 39.6%</td>
<td>49.6% / 64.2% / 58.7% / 37.0%</td>
</tr>
<tr>
<td>Total Loss Train / Validation</td>
<td>0.4408 / 0.8692</td>
<td>0.47 / 0.8486</td>
<td>0.3392 / 0.8859</td>
</tr>
</tbody>
</table>

5.2.3 Inference on Test Set After Tuning

The tuned models were tested on the sample test images to evaluate performance on unseen data. The inference times for the models after tuning stayed the same as in Table 5.5.

Learning Rate Tuned Inferences

From Figure 5.7, the Faster R-CNN ResNet101 and SSD ResNet50 models with the lower learning rate predictions for test images are compared to one another and the ground truth images. The first image shows large pieces of melt overlaying rocks. The second image shows small glass beads on a flat dusty surface with a breccia fragment in the bottom-left and two pieces of melt in the bottom-right. The third image has been augmented with a horizontal flip to test the detectors’ ability to detect particles that have a different perspective. It shows a breccia fragment in the top-right and two pieces of melt in the middle to bottom-left.

The Faster R-CNN ResNet101 model with learning rate $4e^{-3}$ performed well with identifying melt particles with high confidence between 98–100%. However, from Figure 5.7, in the third image, the Faster R-CNN ResNet101 model is unable to detect the melt particle in the
Table 5.9: Summary of Faster R-CNN ResNet101 performance after tuning the number of steps.

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Number of Steps 25k (original)</th>
<th>Number of Steps 16k</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training Time (hours)</td>
<td>7.62</td>
<td>5.1</td>
</tr>
<tr>
<td>mAP @0.5 / @0.75</td>
<td>49.6% / 21.2%</td>
<td>51.5% / 22.9%</td>
</tr>
<tr>
<td>mAP all / large</td>
<td>24.1% / 50.0% / 29.4% / 8.3%</td>
<td>25.4% / 53.2% / 30.7% / 10.46</td>
</tr>
<tr>
<td></td>
<td>24.1% / 50.0% / 29.4% / 8.3%</td>
<td></td>
</tr>
<tr>
<td>Total Loss Train / Validation</td>
<td>0.0152 / 0.5299</td>
<td>0.0351 / 0.4038</td>
</tr>
</tbody>
</table>

bottom-left of the image. The model is reasonably good at identifying breccias, identifying the breccias in the second and third images with 98–100% confidence. In the first image, it falsely predicts a breccia, getting confused by the shiny features of the dust covered rock that could be a breccia or a dusty rock with melt under. The model is not very confident with predictions of glass, reporting low confidence predictions of two glass beads in the second image, approximately 36% confidence. It also predicts a glass bead in the third image which is not in the ground truth image. However, it is possible this is actually a true positive that was not labelled in the ground truth. Since the image was flipped, the perspective highlights the shadow under the possible glass bead more, and the rounded edge of the particle is more prominent. It is convincing that this may in fact be a particle of glass that the detector caught, that the author did not when labelling.

The Faster R-CNN ResNet101 with learning rate $4e^{-4}$ also was able to identify melt well,
however, with lower confidence at 99-100% for large melt objects and 45-99% for medium to small particles of melt. It does not do as well for breccias, identifying the one in the second image correctly, but missing the one in the second image and falsely identifying a breccia in the third image with 45% confidence. It is able to identify more glass, however, with low confidence of 39-57% for majority except for one which was predicted with 92% confidence in the second image. Furthermore, it also falsely detects a glass particle in the first image, likely confusing glass with melt. Following from the observations of the mAP, AR and loss values from Section 5.2.1, the Faster R-CNN ResNet101 model makes more accurate predictions with a learning rate of $4e^{-3}$, however, these improvements are not as significant compared to the original learning rate of $4e^{-2}$.

The SSD ResNet50 model with learning rate of $4e^{-3}$ performed reasonably well with predicting large melt between 93-97%. However, it did not predict small and medium scale objects
very well, missing one of the particles in the second image, and mislabelling the melt in the third image as a breccia. The breccia predictions were also generally low, at 87% and 52% for the second and third images respectively. It did not do very well with glass, overall identifying only one in the second image. The SSD ResNet50 model with learning rate $4e^{-4}$ performed similarly to the model with $4e^{-3}$ learning rate with respect to identifying melt and breccias, however, with overall lower confidence. It also had a false detection of breccia with a 37% confidence in the first image. Overall, it did not perform well on small scale glass, having not identified any in the second image. Overall, the SSD ResNet50 performed quite well with the change in learning rate compared to its initial results. A reduction of the learning rate by a factor of 10 to $4e^{-3}$ was a good adjustment to the parameter. The Faster R-CNN ResNet101 with learning rate $4e^{-3}$ performed similarly to the model with the learning rate of $4e^{-2}$, however, other parameters could be adjusted to further improve the model performance in combination with adjusting the learning rate.

**Number of Steps Tuned Inferences**

Figure 5.8 shows the Faster R-CNN and SSD ResNet50 model predictions for test images with tuned numbers of steps compared to one another and the ground truth images. The learning rate was not adjusted for these models to maintain only one parameter tuning to see how the models perform.

The Faster R-CNN ResNet101 model with 16k steps overall detected melt well, however, missed a few small and medium scale melt objects. Glass was harder as has been the case for other variations of the Faster R-CNN ResNet101 in this work. Two were detected in the second image, with lower confidence of 59% and 35%. In the second image, the breccia was detected, however, was not in the third image. It is clear that the model works well with 16k steps, however, requires additional improvement to the dataset to reduce the impact of false negative detections and improve glass detections. The SSD ResNet50 with 30k and 50k steps performed fairly similarly and will be discussed together. For the image of the melt objects close together, the SSD models outputted multiple bounding boxes for the objects with varying
confidence scores. As the models use non-maximum suppression with an IOU threshold of 60% in post-processing, the overlapping bounding boxes of the same object below this threshold should be removed. This outcome was unexpected and could be attributed to the longer training time resulting in overfitting the model. Overall, the SSD ResNet50 with higher number of steps resulted in the model becoming more confused about distinctions between melt and breccia. The performance was not improved by training the model for more iterations, and in fact degraded the performance. Retaining the original number of steps with the modified learning rate would lead to a better outcome to help the model achieve higher performance,
5.3 Limitations of Implementation and Improvements to the Results

rather than adjusting the number of steps.

Overall, the Faster R-CNN ResNet101 model performed well with the reduction in steps. As there was a time saving benefit in training the model while also maintaining the performance, even increasing it slightly, this parameter could be adjusted for the Faster R-CNN ResNet101 model in the future. The SSD ResNet50 did not significantly improve with the increase in number of steps and resulted in poor performance on the test set.

The Faster R-CNN ResNet101 model performed well with the reduction in steps. As there was a time saving benefit in training the model while also maintaining the performance, even increasing it slightly, this parameter could be adjusted for the Faster R-CNN ResNet101 model in the future. The SSD ResNet50 did not significantly improve with the increase in number of steps and resulted in poor performance on the test set.

5.3 Limitations of Implementation and Improvements to the Results

The poor overall performance showing mAP of around 50% on the higher end could be attributed to the small size of the dataset, the difficulty with accurately labelling the ground truth data, and the general limitations object detection models have for detecting small objects [54]. Without an active presence on the Moon, the understanding of the lunar regolith data is limited. To develop a highly reliable dataset of ground truth images, more knowledge about the regolith is needed which would be improved with further lunar exploration. Additionally, there are a limited number of images available to train the model due to the lack of active missions on the Moon. While the data augmentation helped with increasing the size of the dataset, more data would greatly improve model performance by providing more pixels associated to objects of each class. Additional data augmentation techniques could also be explored.

The metrics could be improved with another pass through the dataset to label the images again. For example, the SSD ResNet50 model displayed false predictions of breccia and glass particles in the image compared to the ground truth labels. However, from close inspection, it is possible that these particles are in fact what the model is suggesting. The labels were generated based on best effort in providing ground truth labels with the help of experts in the field of planetary geology. A powerful outcome could be using the false detections from these models to consider taking a deeper look at the particles the model is identifying. The model could very well be generalizing on features that are accurate to the classes, but ones that the
human eye may have missed. This could basically be another "set of eyes" for supporting labelling the ground truth by including the object detector in the conversation for labelling the images. It would be important to ensure not to bias the model and to critically consider the suggested labels. Additionally, increasing the number of people involved in the labelling task could improve the quality of the labels. A possible improvement to the technique for labelling could be to crowd-source the image labelling process to the wider planetary science community and assign labels based on majority voting.

Additionally, the ‘difficult’ label was not used for this data as the labelling philosophy was to only label the objects that were most likely associated to the labels, and consider the rest of the objects in the background as lunar dust. However, small glass particles are fragmented within the dust, but the resolution is not great enough to properly observe the features. Aside from the shine, the rounded feature of glass beads and the jagged features of glass shards are difficult to observe on small scales. By labelling these background small particles as ‘difficult’ glass labels, they would be removed from the scoring of the object detector and would likely improve the models learning of the glass particles, especially in comparison to the shine of the melt and embedded glass in the breccia.

While these suggestions could help improve the metrics, small object detection is a known limitation of object detection models. Further work would need to be done to understand this limitation and understand techniques that are currently being developed to tackle this problem [54][8].

5.4 Significance of the Results

These results provide a good baseline for working with small datasets of microscopic lunar surface imagery data for object detection tasks. State-of-the-art models such as Faster R-CNN and SSD with ResNet backbones networks have been highly successful for terrestrial implementations and can be adapted for use in planetary geology contexts. These architectures are able to reasonably associate features distinct classes of lunar regolith particles for detection.
The real test will be providing the model an image of the lunar surface from a lunar rover mission where these types of images are being captured. While the environment of the Moon captured in the ALSCC images is similar to what would be expected to be captured from a lunar rover equipped with a microscopic camera instrument, it will be a true test to see if the model can generalize on a new scene of lunar surface imagery from a different region on the Moon. These results provide a starting point for further in depth research into selecting a suitable deep learning model for lunar particle classification and identification, and is likely to only improve as more data becomes available over the next decade.
Chapter 6

Conclusion

This research presents a procedure for preparing a dataset of microscopic lunar imagery for an object detection task using a transfer deep learning approach. The procedure for collecting, labelling, and augmenting images of the lunar surface regolith was described. Using the TensorFlow 2 Object Detection API, various models with Faster R-CNN and SSD architectures with ResNet backbone networks were fine-tuned with a custom dataset of lunar regolith images to understand whether features of lunar regolith could be extracted and used to train an object detection model. The Faster R-CNN ResNet101 and SSD ResNet50 models showed the highest performance of their architectures, and underwent hyperparameter tuning, resulting in overall mAP of 49.7% and 55.8% respectively. These results showed that state-of-the-art deep learning object detection models trained on terrestrial objects can be fine-tuned for planetary geology purposes and can extract features reasonably well. Further, it could support geologists with the complex task of classification and identification of in-situ lunar regolith. Additional work can be done to increase the size of the dataset, improve the quality of the data labels, and overall, improve the confidence and accuracy of the models for identifying objects of interest. As more lunar exploration missions are launched and ultimately collect data of the surface of the Moon over the next decade, more data will become available. There will be more opportunities to increase the size and variation of the data for deep learning, and for implementing this type of solution to assist humans with identifying objects of interest on the lunar surface. This
work shows promising preliminary results for applying deep learning with transfer learning to support planetary scientists with characterizing particles of lunar regolith.

6.1 Future Work

There are a few potential directions to take with this research that begin with building on the techniques presented. A clear path forward would be to try increasing the number of particle classes beyond glass, melt and breccia. Classes such as mineral fragments, and separate classes for different types of breccias such as fragmental breccias and regolith breccias could be included.

To improve accuracy of particle labelling over coarse bounding box labels, an image segmentation approach could be taken to finely outline objects within images. This is a highly manual task and would require more time to accurately do. From there, a Mask R-CNN architecture could be tested to extend on the current Faster R-CNN approach [17].

Other model architectures can be explored to improve object detection for small objects such as glass beads. Potentially looking into the approach Deng et al. described in their paper discussing Extended Feature Pyramid Networks (EFPN) which have improved metrics for small object detection [8].

This work could be adapted for deployment onboard lunar rover missions to support scientific objectives. The TensorFlow 2 Object Detection API and object detection tasks in general require large computational resources and time to train the models, so the adaptation of this solution for a planetary mission would require a lighter weight adaptation. It may be of interest to look into mobile versions of these models that are available for implementation for more portable purposes.
Bibliography


[19] Grant H Heiken, W David Carrier, and By H Grant He ike. Lunar surface closeup stereoscopic photography on the ocean of storms (apollo 12 landing site). n/a, 1971.


Appendix A

Training and Validation mAP for Models
(Section 5.1.1)

Figure A.1: All mAP plots for Faster R-CNN ResNet50.
Figure A.2: All mAP plots for Faster R-CNN ResNet101.

Figure A.3: All mAP plots for Faster R-CNN ResNet152.
Chapter A. Training and Validation mAP for Models (Section 5.1.1)

Figure A.4: All mAP plots for SSD ResNet50.

Figure A.5: All mAP plots for SSD ResNet101.
Appendix B

Training and Validation AR for Models
(Section 5.1.2)

Figure B.1: All AR plots for Faster R-CNN ResNet50.
Figure B.2: All AR plots for Faster R-CNN ResNet101.

Figure B.3: All AR plots for Faster R-CNN ResNet152.
Figure B.4: All AR plots for SSD ResNet50.

Figure B.5: All AR plots for SSD ResNet101.
Appendix C

Training and Validation Loss for Models

(Section 5.1.3)

Figure C.1: All loss plots for Faster R-CNN ResNet50.

Figure C.1: All loss plots for Faster R-CNN ResNet50.
Figure C.2: All loss plots for Faster R-CNN ResNet101.

Figure C.3: All loss plots for Faster R-CNN ResNet152.
Figure C.4: All loss plots for SSD ResNet50.

Figure C.5: All loss plots for SSD ResNet101.
Curriculum Vitae

Name: Hira Nadeem

Post-Secondary
McMaster University
Education and Hamilton, ON
Degrees:
Western University
London, ON
2021 - 2023 M.E.Sc.

Honours and Global Opportunities Award
Awards: 2022

Related Work Mitacs Intern
Experience:
Mission Control Space Services
2022
Teaching Assistant
The University of Western Ontario
2021 – 2022

Presentations:
H. Nadeem, K. McIsaac, M. Battler, M. Cross,
"Lunar regolith particle classification using a deep learning approach”,
International Astronautical Congress Conference Interactive Presentation, 2022.