11-13-2023

An Investigation into Using a Neural Network and LIDAR for Hazard Detection on the Moon's Surface

Anjan Narayanaswamy

An Investigation into Using a Neural Network and LIDAR for Hazard Detection on the Moon's Surface

by

Anjan Narayanaswamy

An Honors Capstone

submitted in partial fulfillment of the requirements

for the Honors Diploma

to

The Honors College

of

The University of Alabama in Huntsville

11/13/2023

Honors Capstone Project Director: Dr. Matt Turner

11/28/2023

Anjan Narayanaswamy (Nov 28, 2023 16:11 CST)

Student (signature) Date

11/21/2023

Project Director (signature) Date

11/28/2023

Department Chair (signature) Date

Digitally signed by Sean M Lane
Date: 2023.11.29 11:01:55 -06'00'

Honors College Dean (signature) Date

HONORS COLLEGE
THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Honors College

Frank Franz Hall

+1 (256) 824-6450 (voice)

+1 (256) 824-7339 (fax)

honors@uah.edu

Honors Thesis Copyright Permission

This form must be signed by the student and submitted with the final manuscript.

In presenting this thesis in partial fulfillment of the requirements for Honors Diploma or Certificate from The University of Alabama in Huntsville, I agree that the Library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by my advisor or, in his/her absence, by the Chair of the Department, Director of the Program, or the Dean of the Honors College. It is also understood that due recognition shall be given to me and to The University of Alabama in Huntsville in any scholarly use which may be made of any material in this thesis.

Anjan Narayanaswamy

Student Name (printed)

Anjan Narayanaswamy (Nov 28, 2023 16:11 CST)

Student Signature

11/28/2023

Date

Table of Contents

Abstract

NASA’s Human Landing System (HLS) is the next US space mission to land astronauts on the moon. In this mission, the spacecraft will have to land safely on the moon and avoid hazards on the moon’s surface. These hazards can include craters which are not suitable for the lander to target for a landing. During the Apollo missions, astronauts had to manually land the lunar lander and rely on their vision to ensure hazards were not in the landing zone. Today, terrain-related navigation is being developed by NASA to automatically detect hazards using LIDAR and deep learning. The purpose of this project is to see if deep learning and neural networks can be implemented onto the HLS designed by the Fall 2023 Senior Design Team to identify crater hazards quickly and accurately on the moon’s surface to help astronauts identify viable landing zones. Research into neural networks was done and a prototype was created which used neural networks and a deep learning algorithm to scan input images of the lunar surface and identify lunar features. Using the knowledge from this prototype, an impact assessment of adding the hazard detection system to the final design was undertaken, and the results showed that the power, mass and cost impacts of the detection system to the overall lander specifications was minimal and that the technology was viable for the Senior Design project.

Introduction

The Human Landing System (HLS) is a part of NASA's Artemis program. The HLS will be designed for a manned mission to the moon in which astronauts can conduct crewed missions on the Moon's surface. For the HLS Mission Design and Development MAE/AE Senior Design, students were tasked with studying previous lunar missions and creating a new mission for the HLS, as well as creating a design solution for the lunar lander. With the last manned NASA operation being Apollo 17 in 1972, many technological changes were considered involving all aspects of the lunar lander.

One of the more recent advances in technology being studied is machine learning and how it can be applied to data instruments on spacecraft. Recently, NASA used LIDAR sensors to generate elevation images of terrain, then implemented machine learning technology to detect hazards on the terrain so that the lander could identify possible landing locations [1]. This technology is very relevant for the design of the lunar lander for the senior design project, as this lander would need to use this technology to safely land the manned lunar lander. The purpose of this thesis paper is to research academic information on the machine learning algorithms used to detect hazards, learn how these algorithms are applied to images provided by LIDAR sensors, and use this information to assess the impact of a hazard detection system on the final vehicle design. This impact assessment investigates mass, power, cost, and major mission changes, if any.

R-CNNs and Semantic Segmentation Background Research

Deep learning is a subset of machine learning which uses multi-layered neural networks to make predictions from large datasets. The function of neural networks is meant to mimic the neurons in the human brain. To do this, a neural network consists of connected node layers, of which there is an input layer, an output layer, and one or more intermediary layers. Each node in the layer has a weight and threshold. When a node has an output value that is higher than its threshold, data is then sent to the next node layer. More node layers can be added on to a neural network to improve optimization and filtering of the algorithm [2].

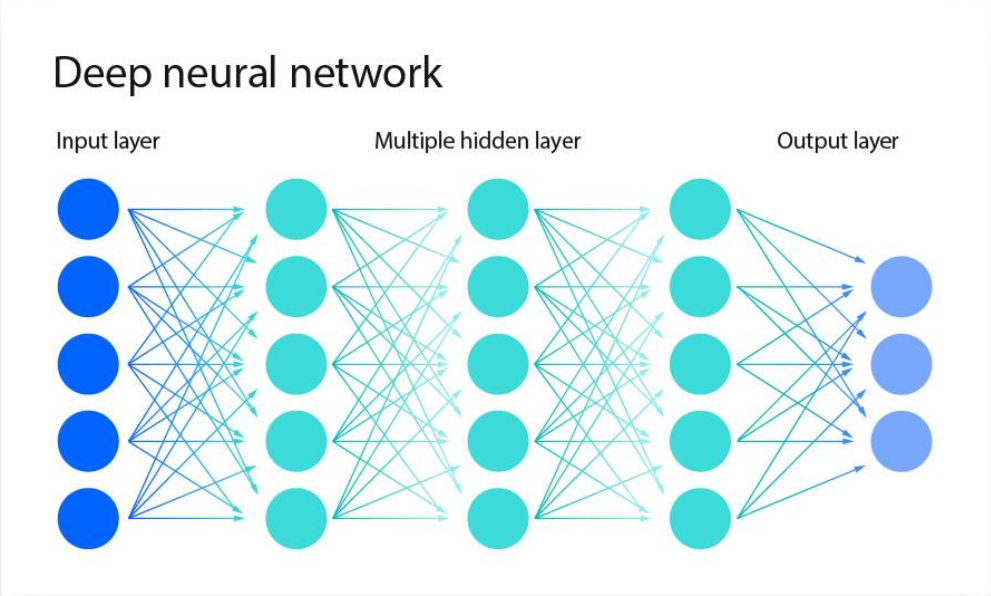


Figure 1. Deep Neural Network Diagram [2]

There are many types of neural networks, but the best neural networks for image input and analysis are convolutional neural networks, or CNNs. A CNN is composed of three layers: the convolutional layer, the pooling layer, and the fully connected layer. The convolutional layer is the first layer, then more convolutional or pooling layers can be added, and finally a fully

connected layer as the output layer. Each layer adds some complexity to the CNN. The first layers tend to identify simpler features like colors, then layers advance to identifying shapes and larger elements to find an object.

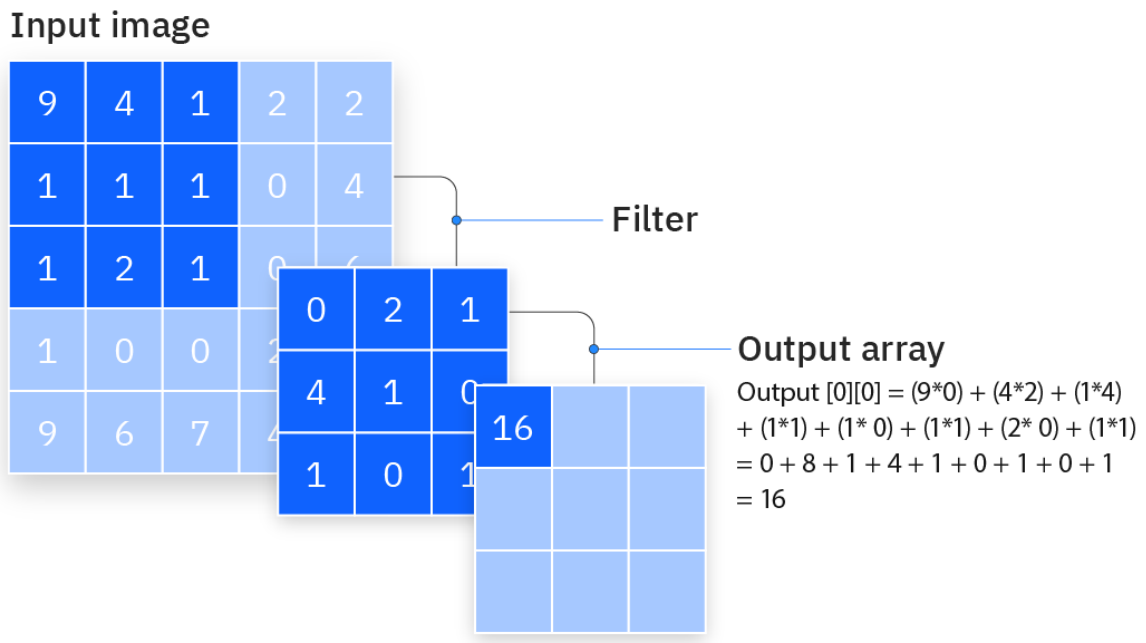


Figure 2. Convolutional Neural Network Diagram [3]

The convolutional layer takes in input data, and uses a filter, or feature detector, that scans images in strides to find certain features using a series of dot products between the input and filter. Once the feature detector finishes scanning the image, it outputs a feature map. By doing this, the convolutional layer creates a map of numeric values that align with the input image, which can then be used by the CNN to identify patterns in the image to identify objects.

The pooling layer works very similarly to the convolutional layer, but its main purpose is to remove parameters from the input. This is important as too many parameters in an input can lead to overfitting. Overfitting is when an algorithm has output which fits too closely with training data, and as a result does not work well with new data sets.

The fully-connected layer connects each of the nodes of the output layer to its previous layer. By doing this, features can be classified through all of the previous layers, which each use different feature detectors [3].

CNNs have been found to work effectively with images of lunar surfaces to identify craters. Previous research on using LIDAR imaging to detect craters on lunar surfaces uses a technique called semantic segmentation [4] [5] [6] [7]. Semantic segmentation uses a CNN and classifies each individual pixel using a label. This produces a label image as the feature map from the CNN, which is then used to identify features in the image. The specific architecture used with semantic segmentation is called U-net. U-net uses a more complex process than a traditional CNN, using an encoder network and a decoder network within the layers of the CNN. These networks not only classify each pixel of the image, but also project the features it learns throughout the process to the final feature map [8].

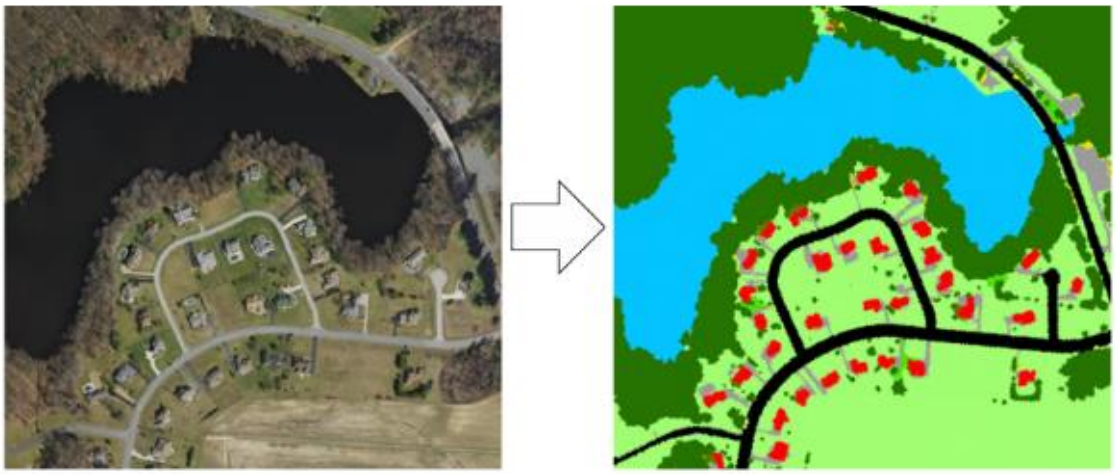


Figure 3. Convolutional Neural Network Example [8]

Deep Learning Detection Prototype Background

To further understand how a CNN can be implemented to detect hazards in images, a deep learning prototype was created using ArcGIS Pro. ArcGIS is a program used to map data. The prototype was created using ArcGIS Pro because the software has built-in Python scripting capabilities, as well as a built-in Deep Learning toolbox that can quickly integrate machine learning to provided data.

ArcGIS fortunately provides a sample program that applies a CNN to lunar imaging to detect craters [9]. The following research will apply and follow this program to further understand the mechanisms and workflow of the program.

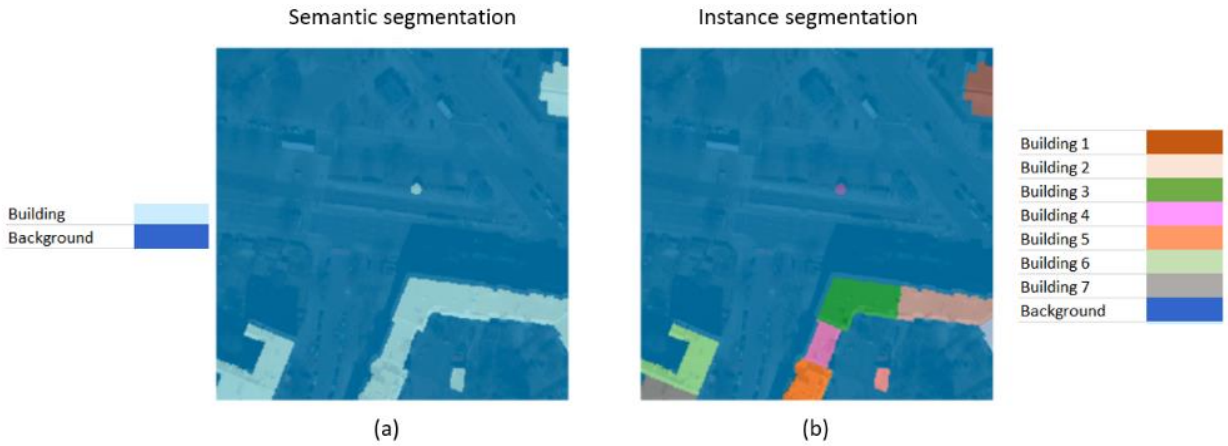


Figure 4. Semantic Segmentation vs. Instance Segmentation [10]

The sample applies a Mask R-CNN (region-based convolutional neural network) model to process the input images. This is a type of CNN, as the name suggests, and differs slightly in the implementation of its algorithm. Rather than using semantic segmentation, Mask R-CNN uses object instance segmentation, which integrates the pixel classification of the semantic segmentation with object detection. Object detection is an algorithm that detects an object class within a bounding box prediction. This not only allows the model to detect a feature but detect a

scalable qualifier of a feature. In our example of hazard detection of craters, object instance segmentation would not only be able to identify a crater, but also classify a crater by its size.

Mask R-CNN is an instance segmentation model built off another model called Faster R-CNN. Faster R-CNN uses two CNNs that return a feature map that highlights features using a bounding box and class label, as well as a confidence score. The CNNs consist of a backbone network and a region proposal network. Both of these networks provide a region proposal, which is a region of the feature map that contains the object. Then the model predicts bounding boxes for the objects in the proposed regions.

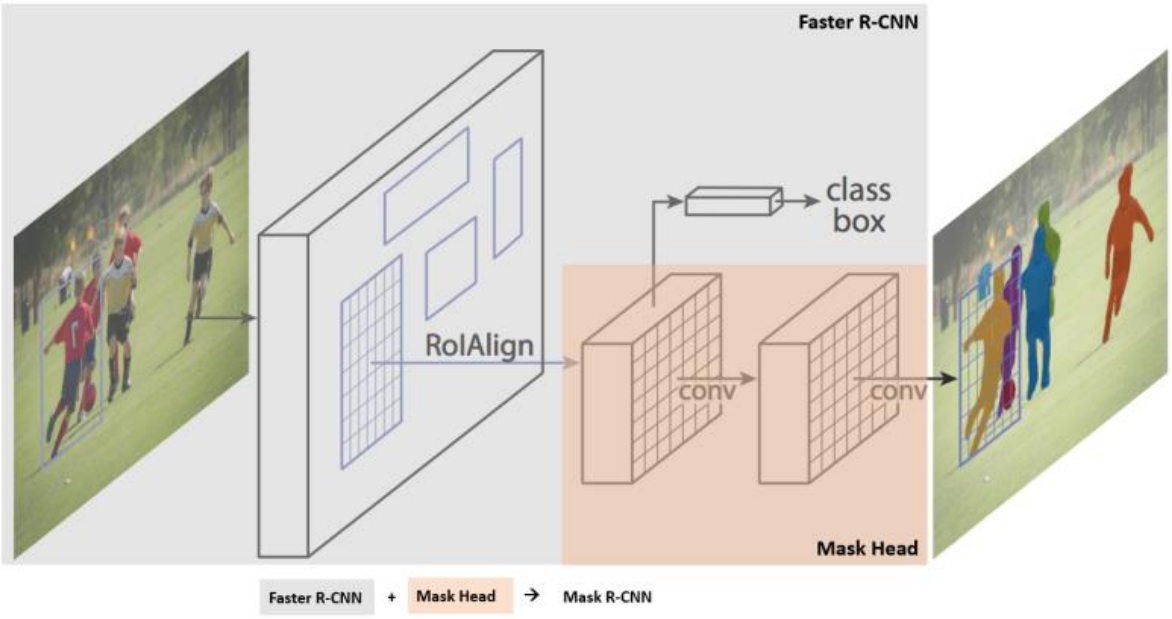


Figure 5. Mask R-CNN Diagram [10]

Mask R-CNN adds onto this process by predicting segmentation masks for each proposed region. The segmentation masks and feature map are then aligned to create a new, fixed size feature map, segmented per pixel [10].

Deep Learning Detection Prototype

As stated in the previous section, the deep learning prototype uses ArcGIS Pro, a Python Jupyter Notebook script, and the Deep Learning Toolbox provided with ArcGIS Pro. The code is shown and explained in this section.

In Figure 6, the required ArcGIS Python packages are imported. These include GIS and Mask R-CNN, which contains the deep learning model. The connection to GIS is also initialized. GIS is the geographic information system, which is used by ArcGIS to analyze and manage given mapping data.

```
In [1]:
import os
from pathlib import Path

from arcgis import GIS
from arcgis.learn import MaskRCNN, prepare_data
```

```
In [2]:
gis = GIS('home')
```

Figure 6. Section 1 and 2 of Prototype Code

In Figure 7, the training data is prepared to be exported. The variable lunar_dem contains a Digital Elevation Model, or DEM of the lunar surface at the South Pole of the moon. This is the input image for the Mask R-CNN model. The variables crater_5_20km and craters_more_than_20km contain the input feature layers for the model. These will help train the model to identify craters by size, either 5 – 20 km, or more than 20 km.

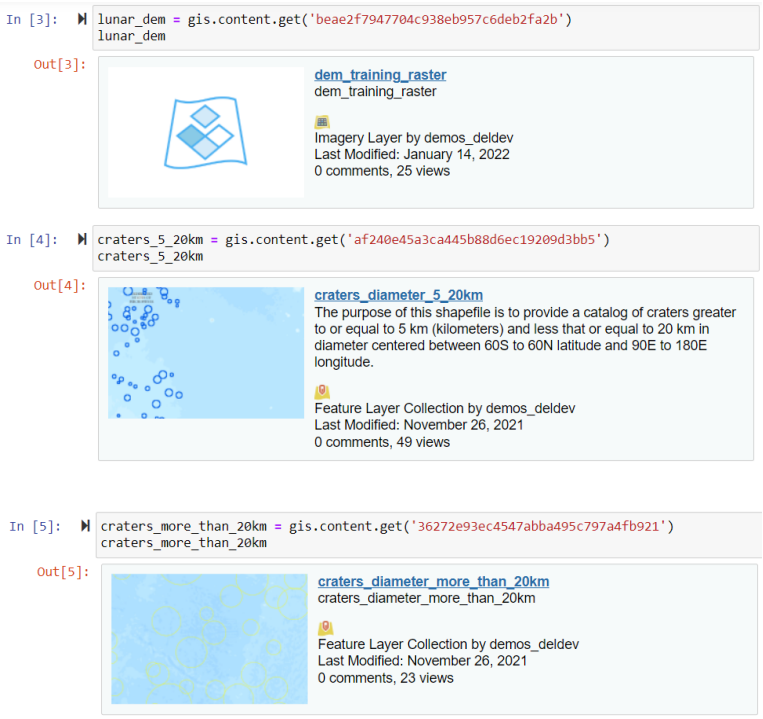


Figure 7. Section 3-5 of Prototype Code

Using the training data from the previous section of code, the Export Training Data For Deep Learning tool is used in ArcGIS Pro. The inputs are shown in Figure 8.

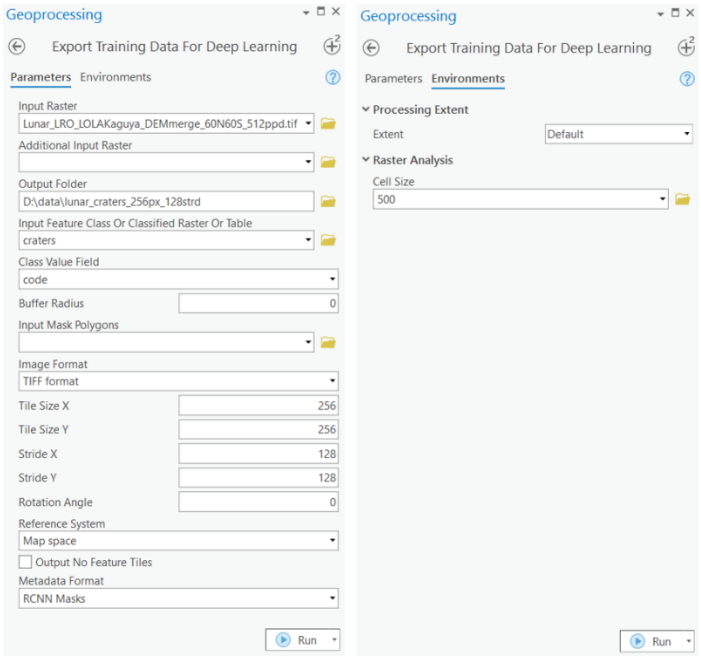


Figure 8. Export Training Data For Deep Learning Prompt

In Figure 9, another dataset of training data is provided with the next section of code and extracted to the same location as the previous data.

```
In [6]:
training_data = gis.content.get('0a4e2d4ad7bf41f6973c7e3434faf7d4')
training_data
```

Out[6]:

lunar_craters_detection_from_digital_elevation_models_using_de
lunar_craters_detection_from_digital_elevation_models_using_deep_le

Image Collection by api_data_owner
Last Modified: January 14, 2022
0 comments, 64 views

```
In [7]:
filepath = training_data.download(file_name=training_data.name)
```

```
In [8]:
import zipfile
with zipfile.ZipFile(filepath, 'r') as zip_ref:
    zip_ref.extractall(Path(filepath).parent)
```

Figure 9. Section 6-8 of Prototype Code

In Figure 10, the data is then prepared using an ArcGIS data preparation Python package.

```
In [ ]:
#Data prep
output_path = Path(os.path.join(os.path.splitext(filepath)[0]))
data = prepare_data(output_path, batch_size=8)
```

```
In [ ]:
#Initialize Data Visualiation
data.show_batch(rows=2)
```

Figure 10. Section 9 and 10 of Prototype Code

At this point, the training data is called to provide a visualization of the training data. Figure 11 shows what the model will receive as its input images.

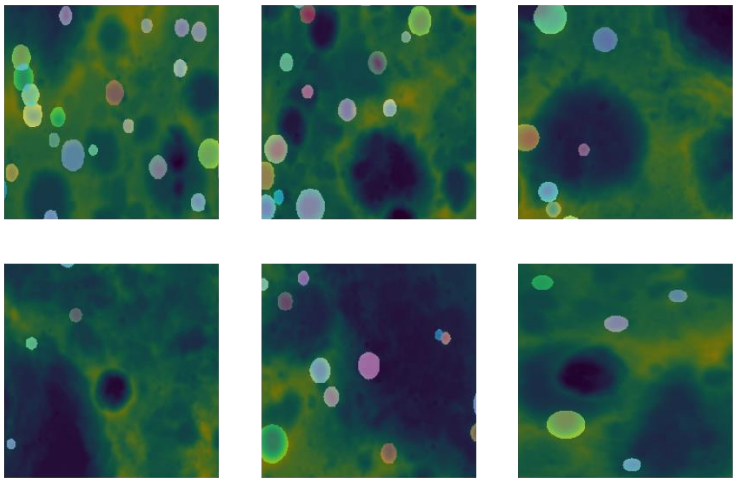
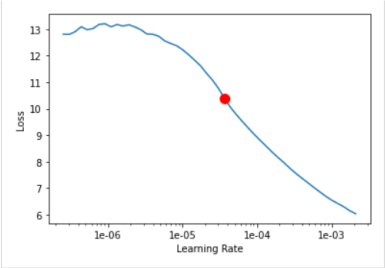


Figure 11. Input Images for Mask R-CNN Model

The code in Figure 12 prepares the Mask RCNN model and optimizes the learning rate of the function.

```
lr = model.lr_find()
lr
```




```
3.630780547701014e-05
```

Figure 12. Section 11 of Prototype Code

The next section of code in Figure 13 trains the model. Each epoch represents the model fully running through the training data. The training and validation error or losses are represented by the columns labelled train_loss and valid_loss. In each epoch, the algorithm calculates the amount of error it produces when identifying each feature and decreases the error each epoch by using the CNN workflow. Eventually, the model stops when the error ceases to change a significant value, which in this case is epoch 56.

```
model.fit(100, lr, early_stopping=True)
```

56.00% [56/100 1:09:59<54:59]

epoch	train_loss	valid_loss	time
0	6.944399	5.095363	01:00
1	3.083338	2.410944	01:04
2	2.264618	2.287425	01:08
3	2.102938	2.249945	01:09
4	2.032436	2.229557	01:15
5	2.009923	2.240452	01:15
6	1.982428	2.191599	01:12
7	1.960892	2.277536	01:17
8	1.991066	2.217761	01:17
9	1.938140	2.132267	01:17
10	1.868605	2.113236	01:19

Figure 13. Section 12 of Prototype Code

The model is then saved in Figure 14, and the results are visualized, as shown in Figure 15. An assessment of the accuracy is also recorded, shown in Figure 16.

In [ ]:
```
#Save model
model.save("moon_mrcnn_2", publish=True)
```

In [ ]:
```
#Results visualiztion
model.show_results(rows=2, mask_threshold=0.7)
```

Figure 14. Section 13 and 14 of Prototype Code

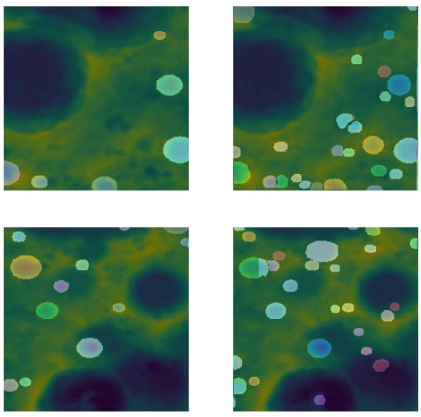


Figure 15. Output Images from Model Training

```
model.average_precision_score()
```

100.00% [10/10 00:09<00:00]

```
{'1': 0.4536149187175708}
```

Figure 16. Section 15 of Prototype Code

Now that the model has been trained, the Detect Objects Using Deep Learning tool is used to detect craters in the initial lunar DEM, as shown in Figure 17. The saved Mask R-CNN model is called using this tool as well.

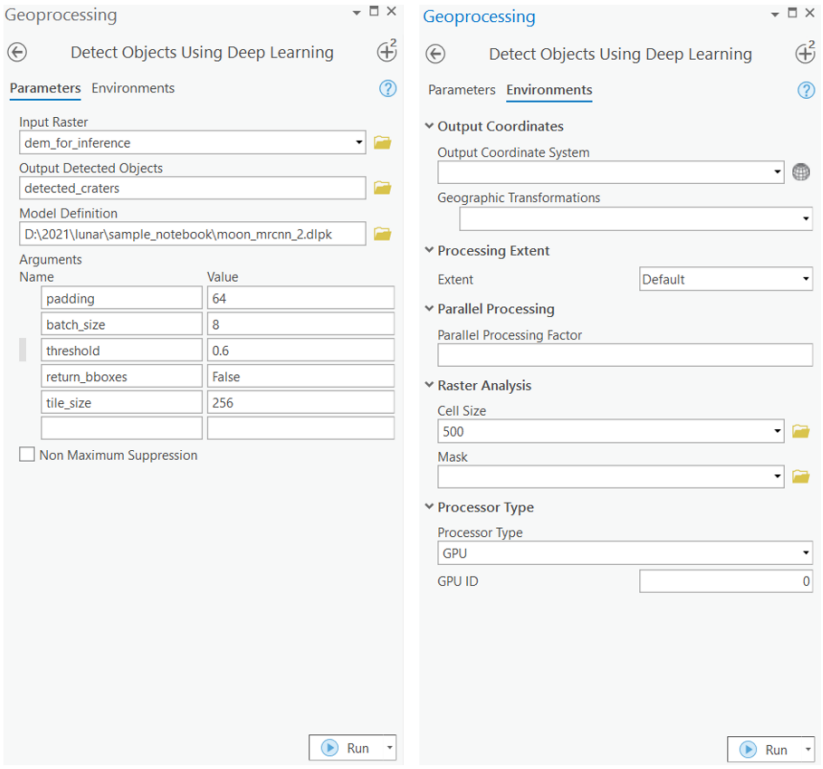


Figure 17. Detect Object Using Deep Learning Prompt

The final results visualization is shown in the last section of code, shown in Figure 18. The output images are shown in Figure 19 and Figure 20. Two areas of the full DEM are called in the code. The craters are outlined in red in the images.

```
In [ ]:
from arcgis.mapping import WebMap
## Visualize Craters
LunarArea1 = gis.content.get('1a76ed548cfc4e159d860ae253e5ecc3')
map1 = WebMap(LunarArea1)
map1
```

```
In [ ]:
## Visualize Crater in different area
LunarArea2 = gis.content.get('2ce5ad91fab649c7982dde750cce3390')
map2 = WebMap(LunarArea2)
map2
```

Figure 18. Section 16 and 17 of Prototype Code

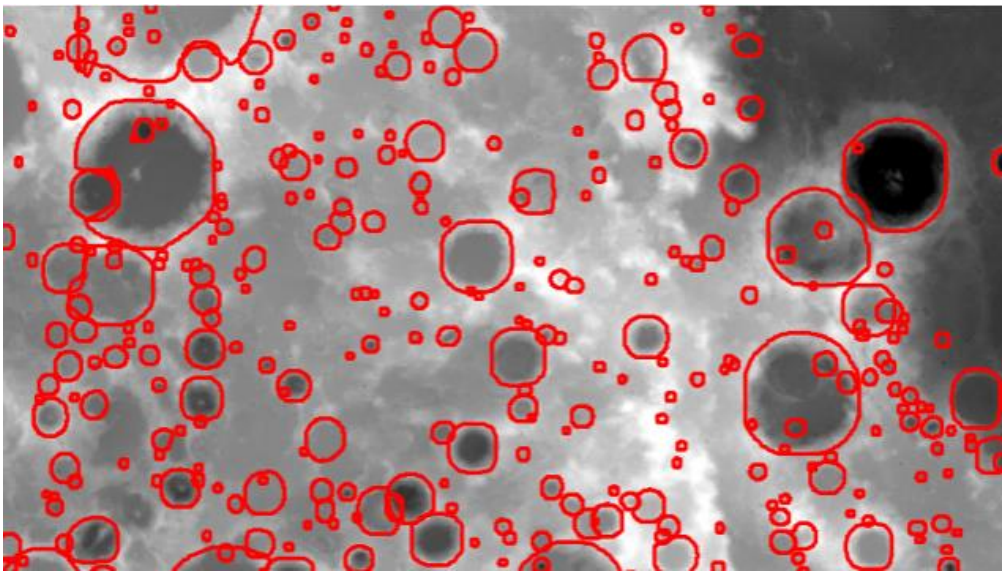


Figure 19. Area 1 Final Results Image from Trained Mask R-CNN model

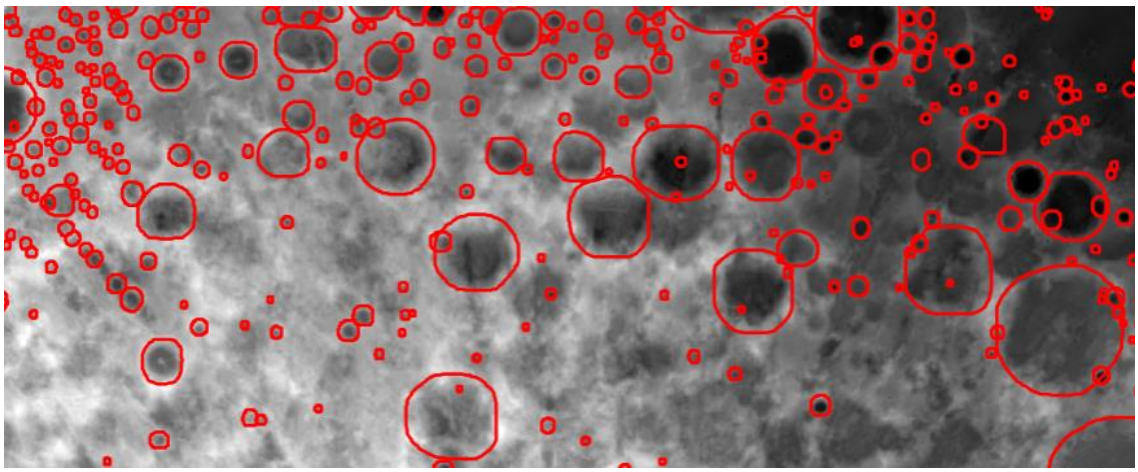


Figure 20. Area 2 Final Results Image from Trained Mask R-CNN Model

The prototype is able to find the craters and identify them by size as intended. In some images, there are some craters that are missed by the model. This could be attributed to the lower resolution images selected for the prototype, as well as a relatively smaller input dataset. More images can be used to train the model and improve the accuracy and precision of the detection. The quality of the training data provided to the model could also be improved, as it only contained craters shown as circles, when in reality craters can have more irregular shapes. Despite this, the model was still able to identify non-circular craters even though it can still be better trained using better quality input datasets. With these results, the prototype shows that the deep learning model should be effective on the lunar lander with more production level testing.

Impact of Hazard Detection System on Final Design

To have a fully functioning hazard detection system on the lunar lander design used for the senior design project, the equipment required should not add too much mass or power, as these are the strictest requirements for the lander. The cost is also taken into consideration, but its requirement is not as strict as mass or power.

Component	Count	Mass [kg]	Volume [cm^3]	Power [W]	Cost [USD]
Landing Sensors / Equipment					
[redacted]					
Landing Lidar	2	2.4	90	20	150,000
[redacted]					
Obstacle Processing Computer	2	0.6	6,000	10	150,000
[redacted]					

Figure 21. Hazard Detection Components

As shown in Figure 21, the components required are the LIDAR sensors and the obstacle processing computers. In total, these have a mass of 3 kg and 30 W of power. Both of these values are very minimal in the final mass of the lunar lander. The mass and power impact of the ADACS subsystem is shown in Figure 22 and Figure 23. The total cost estimate of $300,000 has a slightly larger impact, but the safety consideration helps warrant the decision to include a hazard detection system. The overall lander cost of $4,300,000,000 also puts in perspective the impact the system has on cost, which is very minimal.

Dry Mass Per Subsystem		
Subsystem	**Descent Calculated [kg]**	**Ascent Calculated [kg]**
ADACS	16.76	400.3
Payload	130	1,270
Power	-	118
Propulsion	247.1	367.9
Structures	730	1,030
Thermal	304.32	135.87
C&DH	-	40.8
Comms	-	28.24
Total	**1,428.2**	**3,391.1**

Figure 22. Total Ascent-Descent Vehicle Mass Breakdown

Subsystem Power Draw [W]	
ADACS	< 200
Payload	2800
Propulsion	0
Structures	0
Thermal	550
CD&H	95.2
Comms	62.5
Total	3707.7
Subsystem Power Generation [W]	
Power	5000

Figure 23. Total Ascent-Descent Vehicle Power Breakdown

Results

Background research into deep learning has provided a deeper understanding of convolution neural networks and how they can be used to classify objects in images. These CNNs can be used to identify craters on the lunar surface. The deep learning detection prototype provided more insight into the workflow for using a CNN, specifically the Mask R-CNN model. The results from the prototype are very positive, as the trained model can identify craters or all sizes on the input image. Although the prototype has some slips, this can be easily fixed by increasing the quality of training data, increasing the amount of training data, and adding complexity to the CNN through additional layers if necessary. The prototype shows that the model can work in a real life setting on the lunar lander design. The impact assessment of the hazard detection equipment has shown that the mass and power of the overall lander will not be greatly impacted and should be an affordable option for the lunar lander.

References

[1] Amzajerdian, F., Petway, L. B., Hines, G. D., Roback, V. E., Reisse, R. A., & Pierrottet, D. F. (2013). LIDAR sensors for autonomous landing and hazard avoidance. *AIAA SPACE 2013 Conference and Exposition*. https://doi.org/10.2514/6.2013-5312

[2] *What are Neural Networks? | IBM*. (n.d.). https://www.ibm.com/topics/neural-networks

[3] *What are Convolutional Neural Networks? | IBM*. (n.d.). https://www.ibm.com/topics/convolutional-neural-networks

[4] Moghe, R., & Zanetti, R. (2020). A deep learning approach to hazard detection for autonomous lunar landing. *The Journal of the Astronautical Sciences, 67*(4), 1811–1830. https://doi.org/10.1007/s40295-020-00239-8

[5] Ghilardi, L., D'Ambrosio, A., Scorsoglio, A., & Curti, F. (2021). Image-Based lunar landing hazard detection via deep learning. *ResearchGate*. https://www.researchgate.net/publication/349044028_Image-Based_Lunar_Landing_Hazard_Detection_via_Deep_Learning

[6] Furfaro, R. (2019, October 28). *Deep learning semantic segmentation for vision-based hazard detection*. https://hdl.handle.net/10589/149486

[7] Tomita, K., Skinner, K. A., Iiyama, K., Jagatia, B., Nakagawa, T., & Ho, K. (2020). Hazard Detection Algorithm for planetary Landing using semantic segmentation. *ASCEND 2020*. https://doi.org/10.2514/6.2020-4150

[8] *How U-net works? | ArcGIS API for Python*. (n.d.). https://developers.arcgis.com/python/guide/how-unet-works/

[9] *Lunar Craters Detection using Deep Learning | ArcGIS API for Python*. (n.d.). https://developers.arcgis.com/python/samples/lunar-craters-detection-from-dem-using-deep-learning/

[10] *How Mask R-CNN works? | ArcGIS API for Python*. (n.d.). https://developers.arcgis.com/python/guide/how-maskrcnn-works/

Appendix

Appendix A: Deep Learning Prototype Code

In [ ]:
```
import os
from pathlib import Path

from arcgis import GIS
from arcgis.learn import MaskRCNN, prepare_data
```

In [ ]:
```
gis = GIS('home')
```

In [ ]:
```
lunar_dem = gis.content.get('beae2f7947704c938eb957c6deb2fa2b')
lunar_dem
```

In [ ]:
```
craters_5_20km = gis.content.get('af240e45a3ca445b88d6ec19209d3bb5')
craters_5_20km
```

In [ ]:
```
craters_more_than_20km = gis.content.get('36272e93ec4547abba495c797a4fb921')
craters_more_than_20km
```

In [ ]:
```
training_data = gis.content.get('0a4e2d4ad7bf41f6973c7e3434faf7d4')
training_data
```

In [ ]:
```
filepath = training_data.download(file_name=training_data.name)
```

In [ ]:
```
import zipfile
with zipfile.ZipFile(filepath, 'r') as zip_ref:
    zip_ref.extractall(Path(filepath).parent)
```

In [ ]:
```
#Data prep
output_path = Path(os.path.join(os.path.splitext(filepath)[0]))
data = prepare_data(output_path, batch_size=8)
```

In [ ]:
```
#Initiail Data Visualiation
data.show_batch(rows=2)
```

In [ ]:
```
#Create model and find learning rate
model = MaskRCNN(data)

lr = model.lr_find()
lr
```

In [ ]:
```
#Fit model with learning rate
model.fit(100, lr, early_stopping=True)
```

In [ ]:
```
#Save model
model.save("moon_mrcnn_2", publish=True)
```

In [ ]:
```
#Results visualiztion
model.show_results(rows=2, mask_threshold=0.7)
```

In [ ]:
```
#Assess accuracy
model.average_precision_score()
```

In [ ]:

```
from arcgis.mapping import WebMap
## Visualize Craters
LunarArea1 = gis.content.get('1a76ed548cfc4e159d860ae253e5ecc3')
map1 = WebMap(LunarArea1)
map1
```

In [ ]:

```
## Visualize Crater in different area
LunarArea2 = gis.content.get('2ce5ad91fab649c7982dde750cce3390')
map2 = WebMap(LunarArea2)
map2
```