Crater Detection

- Maryam Labaf

1.1 The code

1.1.1 Custom modules used

- `crater_preprocessing.py`
- `crater_loader.py`
- `crater_data.py`: classes for data management. Similar to tensorflows datasets
- `crater_cnn.py`: the class for CNN implementation
- `crater_nn.py`: the class for NN implementation
- `network_blocks.py`: some methods to build CNN structure
- `crater_plots.py`: some methods to display plots from CNN
- `helper.py`: some miscelaneous helper functions

1.1.2 Scripts for experiment

- `train_cnn.py`: for model training. It generates a files that can be loaded
- `training_nn.py`: for model training. It generates a files that can be loaded
- `sliding_window_2networks.py`: loads both models and perform crater detection while sliding through the image
- `remove_duplicates.py`: processes csv outputs from sliding. Removes duplicated detections

1.1.3 Extra non-critical scripts

- `use_trained_model.py`
- `visualize_rectangles.py`
- `crater_slice_window.py`
- `xmeans.py`

1.2 Preproccessing Data:

Used the file created in phase I and resizded data into 50 by 50.

1.3 CNN:

CNN was implemented with code from Tensor Flow tutorial number 2 by Magnus Erik Hvass Pedersen-GitHub.
https://github.com/Hvass-Labs/TensorFlow-Tutorials/blob/master/
A new Network class was written using code from this repository, and customized as needed.

CNN Design: Consist of six layers:

- Input layer: 50 by 50.
- Convolutional and Pooling Layer 1: 16 filters with size 5 by 5. Pooling was done with 2 by 2 window size and picked the maximum value. Activation function was a relu function.
- Convolutional and Pooling Layer 2: 36 filters with size 5 by 5. Pooling was done with a 2 by 2 window size and picked the maximum value. Activation function was a relu function.
- Fully Connected Layer: Two fully layers, the first one with 64 neurons and second one with 16 neurons. Activation functions were both relu functions.
- Output: Two neurons. After sigmoid functions were used for each outputs, softmax method was used.
- Number of Epochs: 20
- Learning Rate: Adam Optimizer with starting learning rate 10^{-4}.

Trained this CNN by using craters and non-craters data of tile_24.

Accuracy on the Test Set:

- Trained CNN had more than 98% accuracy in various random train and test sets. There were only 2 missclasified images.

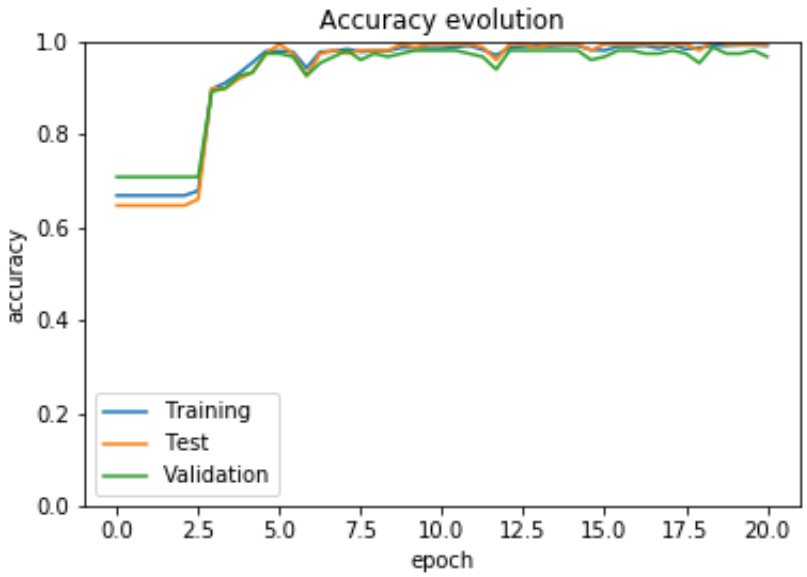
Accuracy on the Validation Set: -Trained CNN had about 97% accuracy on the validation set after achieving 98-99% accuracy on the test set.

Filter activations With the trained model, it is posible to visualize the activation values for the filters at the convolutional layers

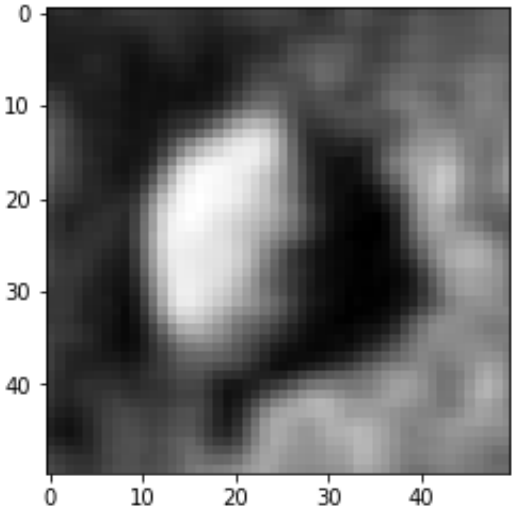
Here it can be appreciated that these filters have identified features such as the shade patterns in the crater

1.4 Image Pyramids and Scanning:

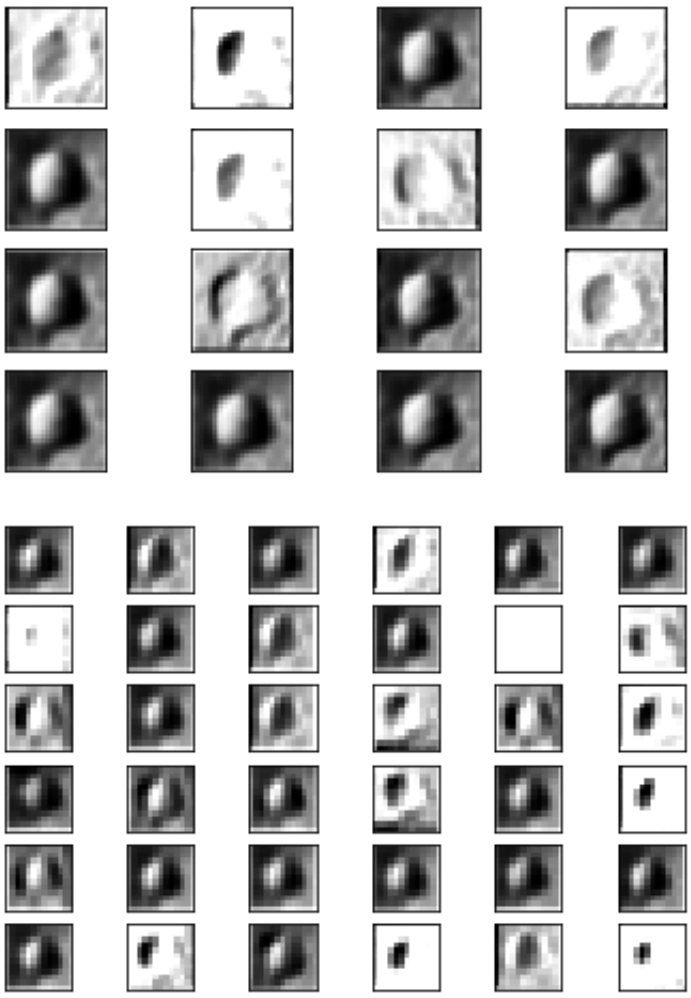
By modifying the sample code given, shrinked the original images with rate $\frac{2}{3}$. Since exponential shift of window sizes had too many gaps, smoothed the gap by using a set of window sizes,20, 22, 24, 26 and 28, in each level of the pyramid. A stepsize of the window was 2. Copied images taken from the windows were normalized into 50 by 50 so that they can fit into our CNN and NN.



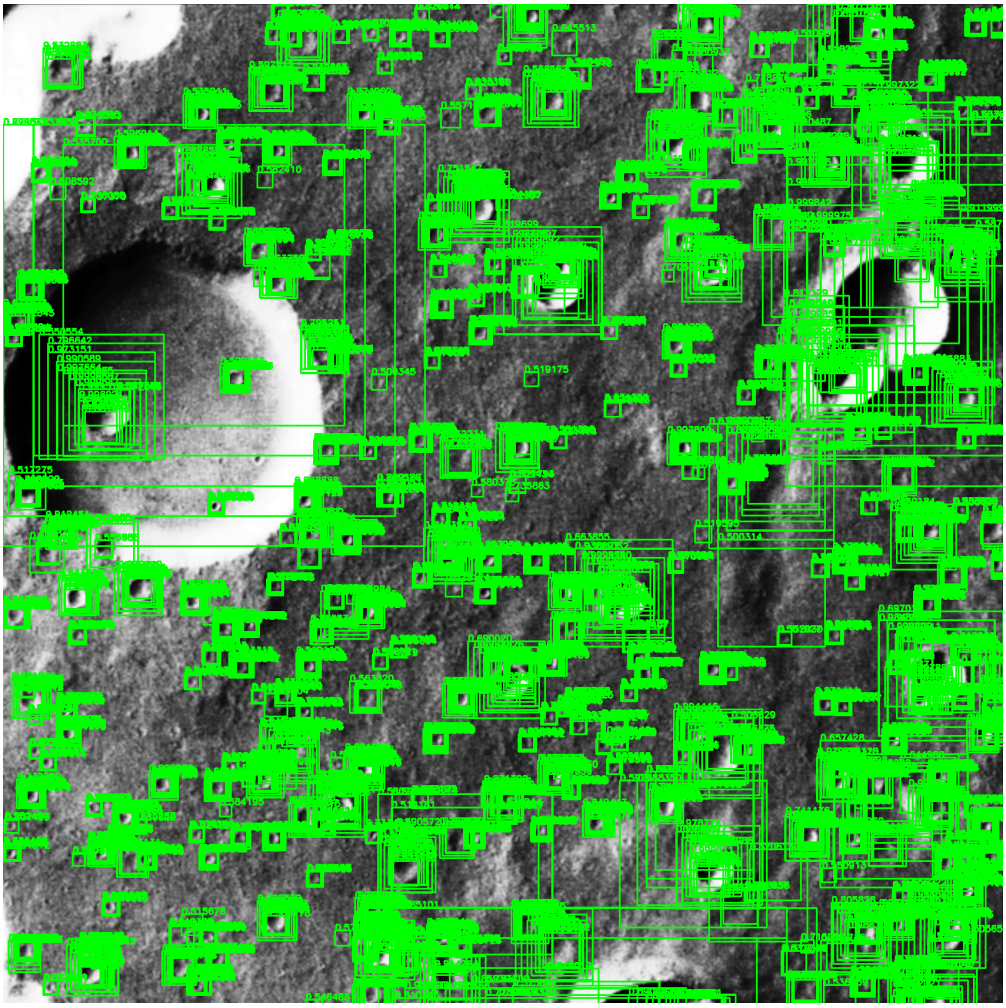
Accuracy evolution



Example crater visualization



Filter activations for example crater picture



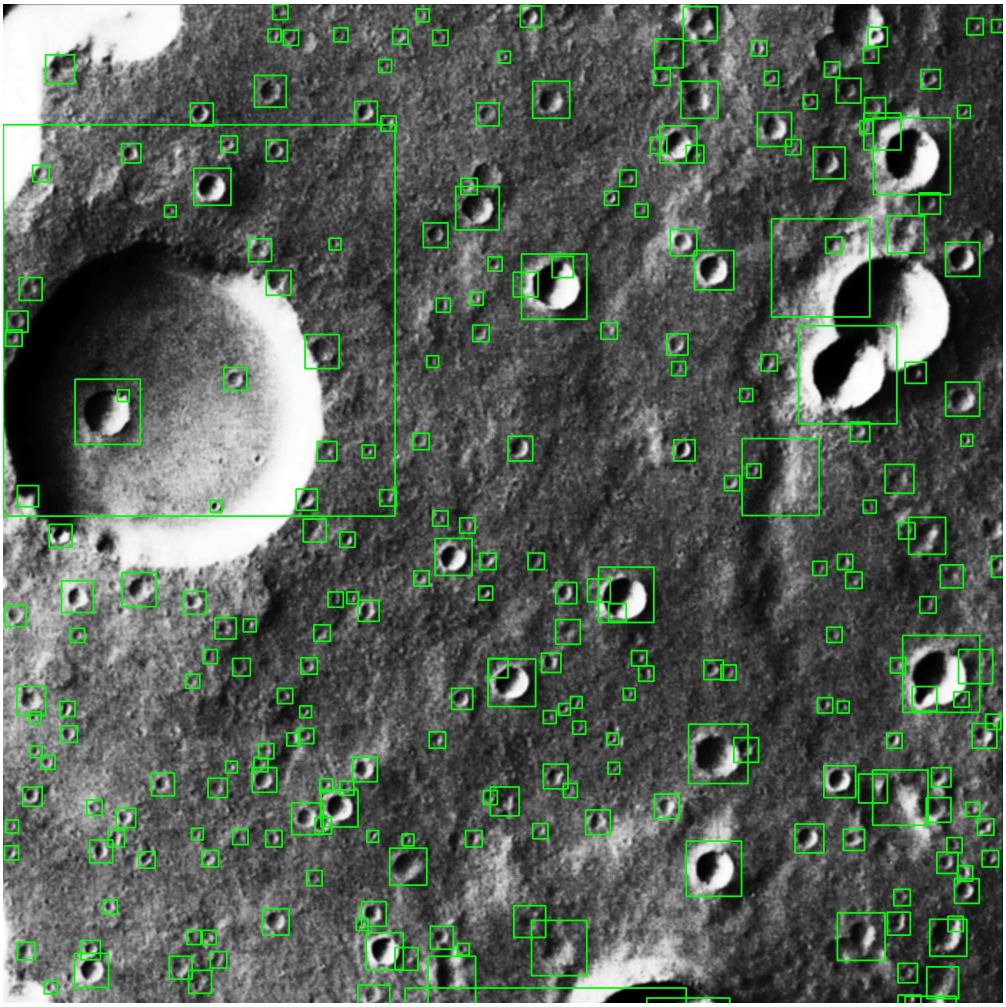
A sample output from tile3_24

1.5 Deleting Duplicates:

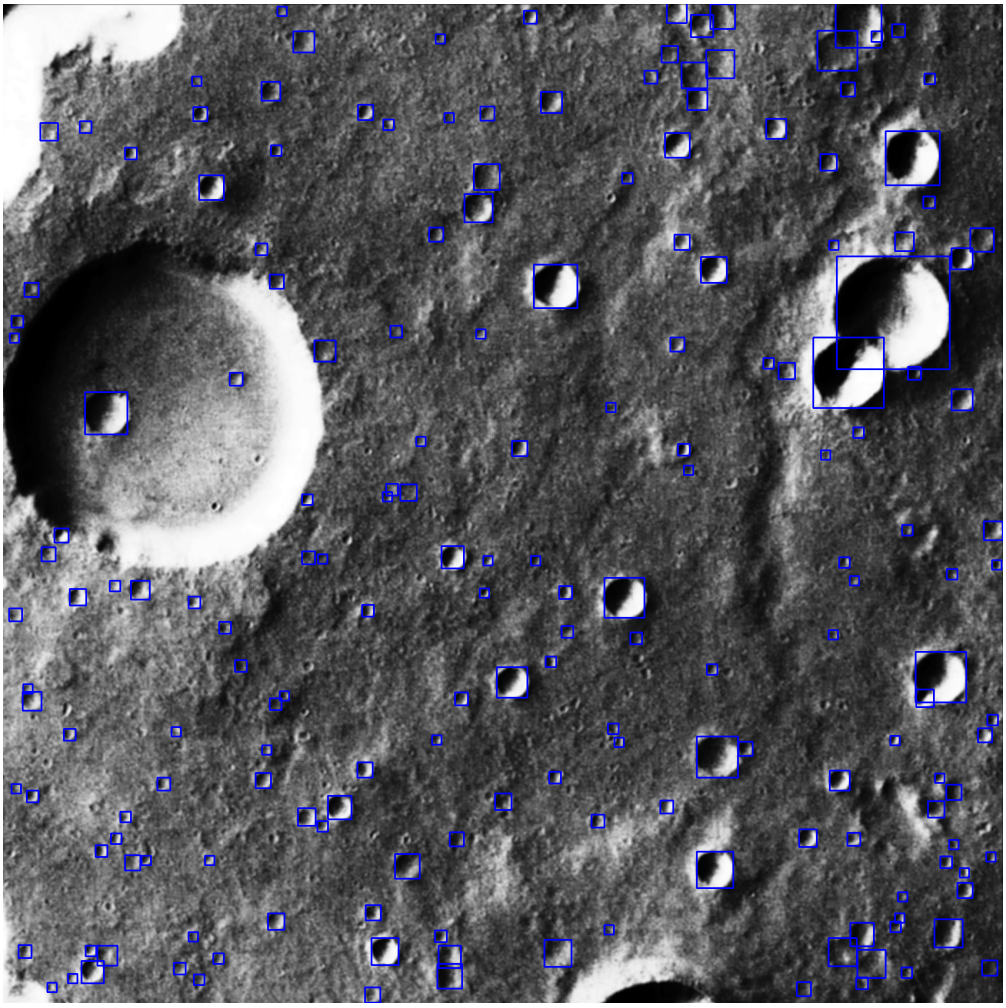
Decided to treat a detecting duplicates problem as a clustering craters problem.

Clustering methods for removing duplicate windows: - X-Means: The number of clusters was not necessary. Achieved to detect craters with large sizes, but failed to detect craters with small sizes. - K-Means: Achieved to detect craters with small sizes, but failed to detect craters with large sizes with a proper number of clusters. However, finding the proper number of clusters was the problem. - BIRCH: Balaneced iterative reducing and clsutering using hierarchies. Worked good to detect almost all the craters. Also, computational time was short. - Hierarchical Cluster based on Euclidian distance: Similar to BIRCH, gathered the set of datas which were closed to one randomly chosen data, put into a cluster and removed from the dataset in each iteration. - Extended hierarchical Cluster based on Euclidian distance: Used the probability given by CNN and NN to weight the importance of data in each cluster after the previous method.

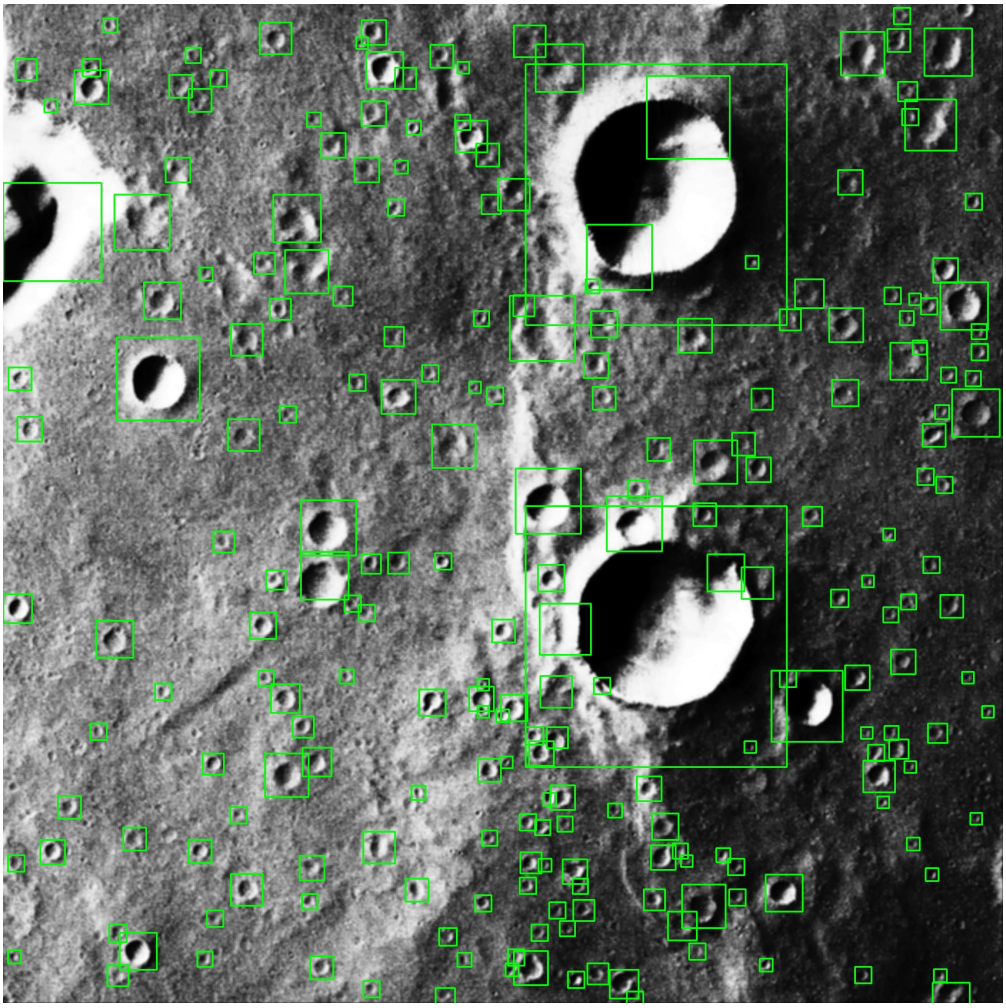
1.6 Object Detection:



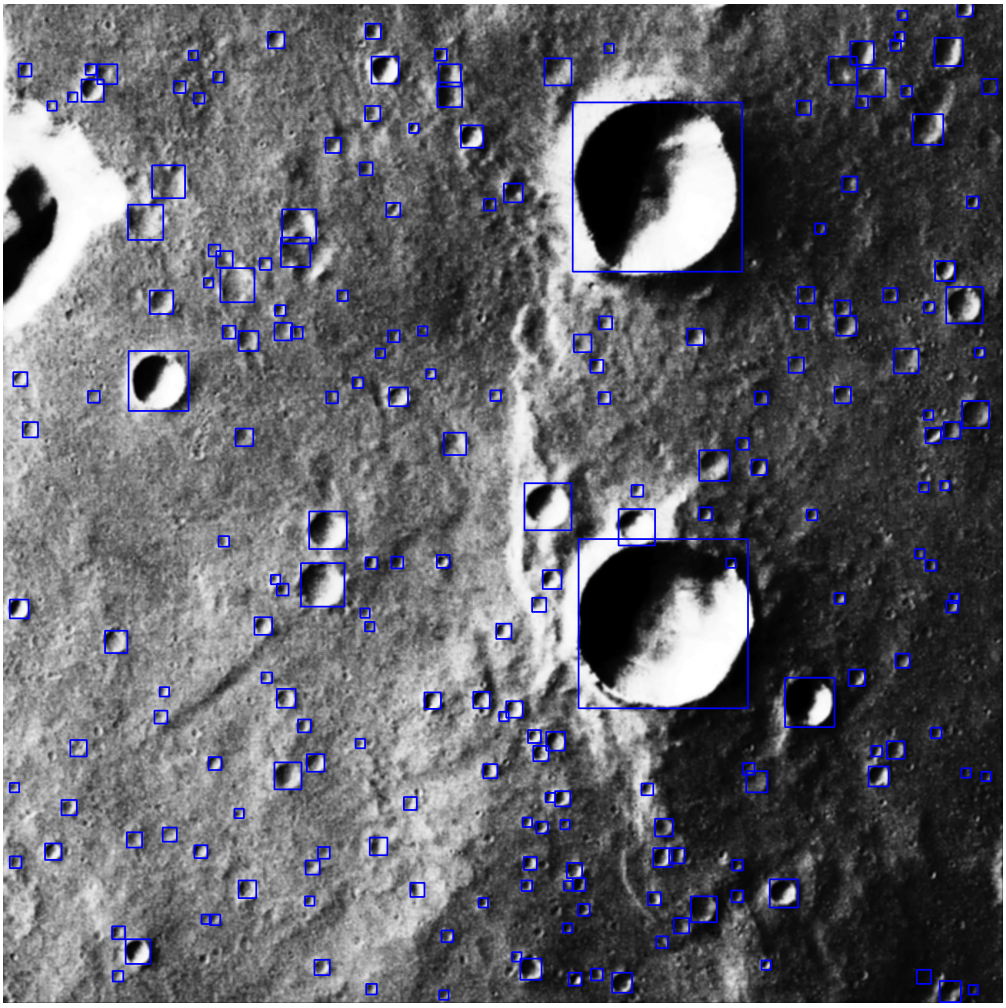
Final output from tile3_24



Ground truth for tile3_24



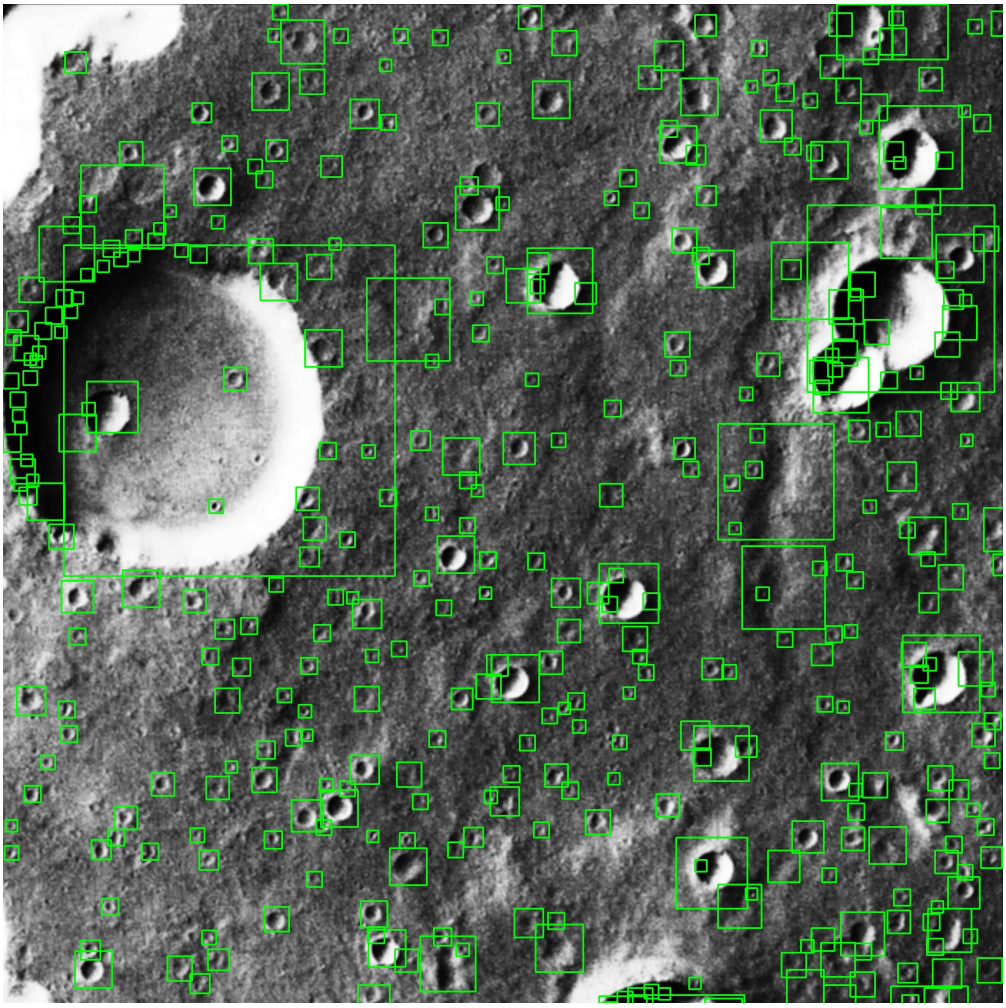
Final output from tile3_25



Ground truth for tile3_25

1.7 Analysis of CNN and NN:

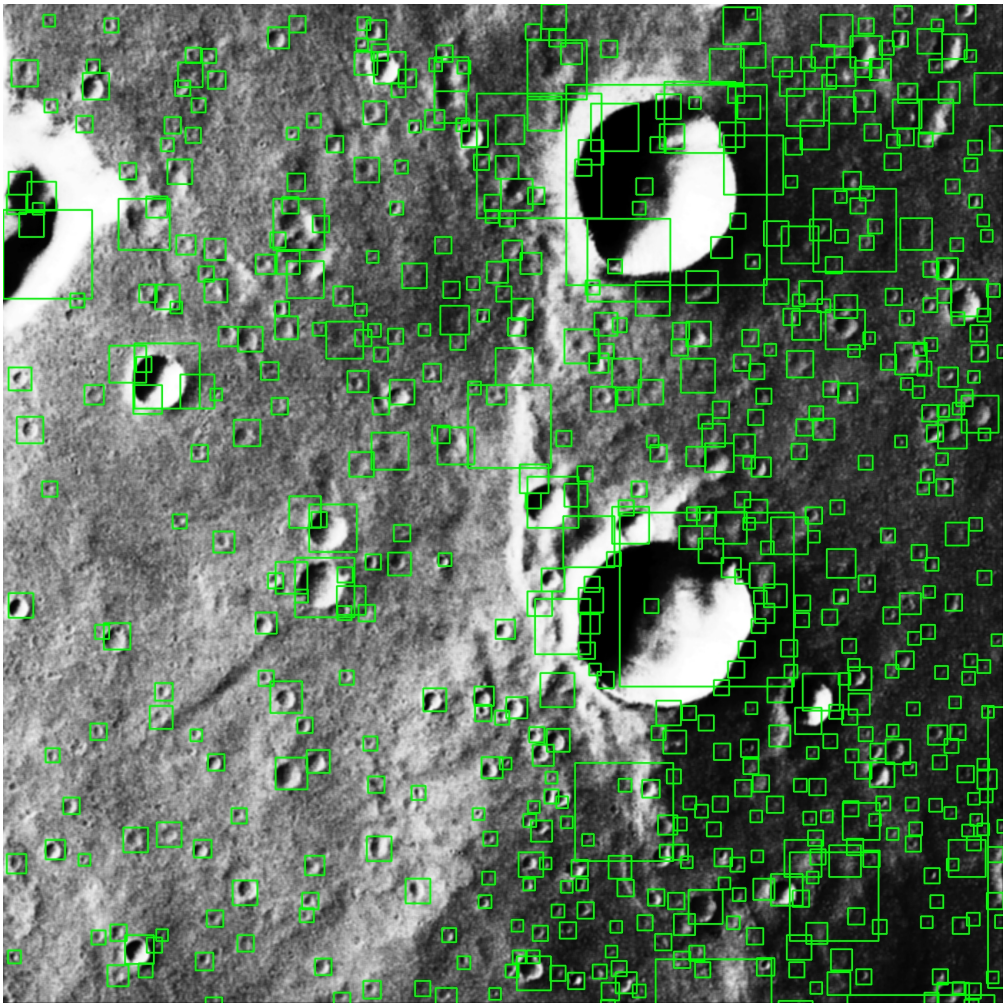
Both CNN and NN worked good for finding craters from the base pictures. However, NN had much more false positives than CNN.



Final output from tile3_24 by NN trained in phase 1

1.8 False Positive and False Negative Analysis:

Both CNN and NN detects almost all craters from base pictures, 24 and 25. However, it seemed difficult for them not to detect a non-crater as a crater. CNN has better eliminations of false positives, but there still exists a room for improvement.



Final output from tile3_25 by NN trained in phase 1