Garrett Alston, Flavio Andrade, Nikith Anupkumar, Eucli Barahona

Term Project Phase II Report

Apply Neural Network to Real World Application (Crater)

CS 480

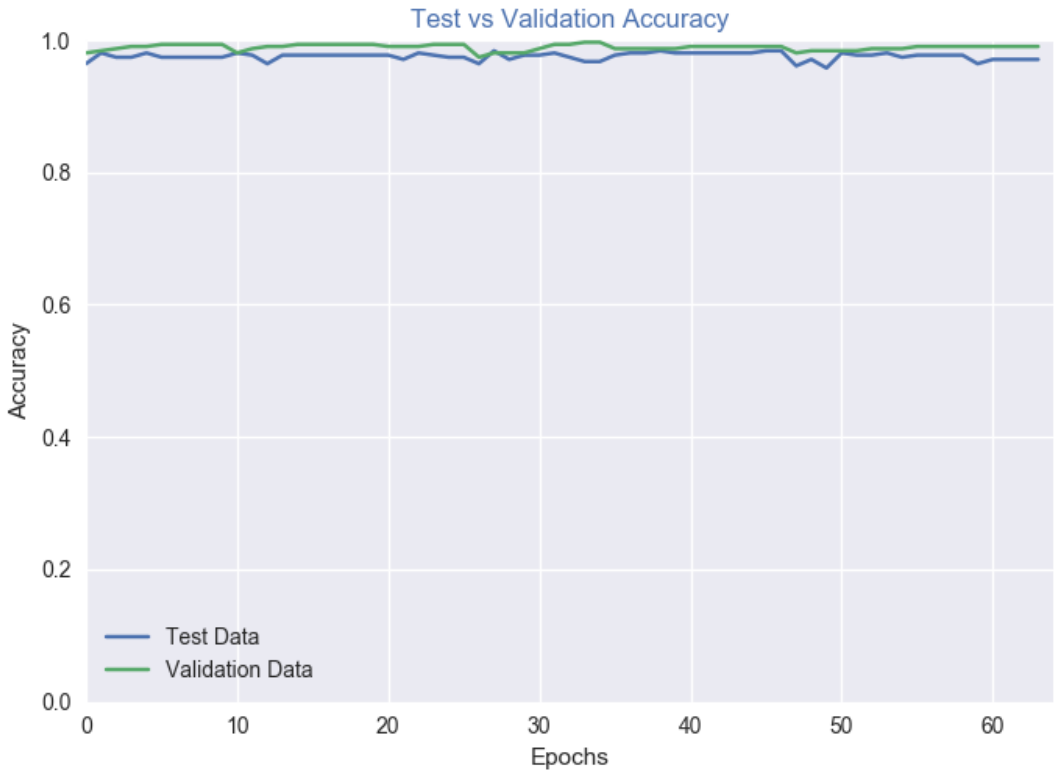
For this phase of the project, we are tasked to implement a Convolutional Neural Network to detect craters on a given dataset, and then utilized a trained model to detect craters on a largescale image to detect craters through a sliding window method.

The first task was to develop a dataset. Using all the original images from both tile3_24 and tile3_25, as well as the normalized images from Phase I, we created one large dataset. From this dataset, we then created a partition of the dataset such that the partition consisted of a training subset, a validation subset and a test subset. The subsets contained 70%, 15%, and 15% of the original dataset respectively. From this we ran into our first problem. We were not passing shuffled subsets and thus the Network was consistently performing poorly. Once we realized that the data was indeed being passed in a linear format, we ensured a shuffling occurred and thus the Network began to learn efficiently.

From here our next task was to develop and train an efficient model. We tried various input sizes, ranging from 28x28 to 200x200. Towards the end of our analysis we concluded that for runtime efficiency, a smaller sized dataset would produce faster results. Using 28x28 as our input size, we developed a Network consisting of 3 ConvPool Layers, 2 Fully Connected Layers, and a SoftMax layer for our output layer. We utilized a Leaky ReLU activation function. We chose this over ReLU due to the fact that ReLU creates a diminishing gradient problem for negative outputs. By utilizing Leaky ReLU with a small return value on negative activations, we can utilize a small gradient rather than utilizing 0, thus causing dead neurons in our network that stop being modified. We tested various feature maps on both layers but found optimal results in the range of 10-12 in the first layer and 20 in the second.

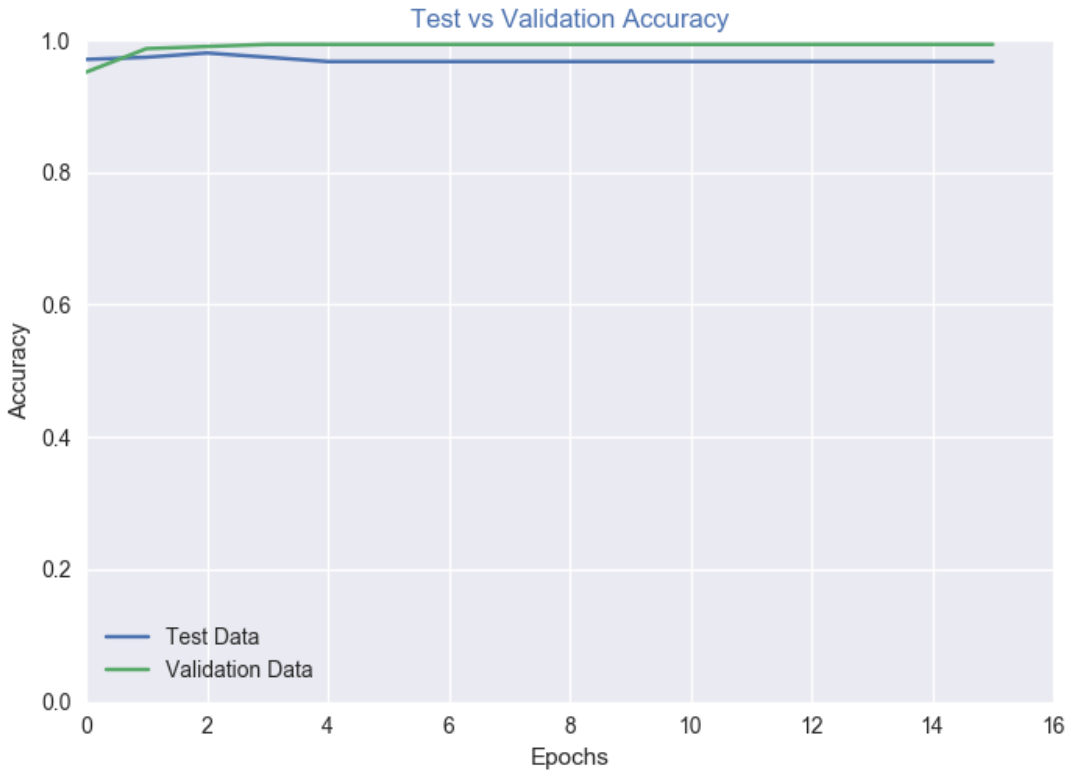
When it came to creating a network that ran faster and trained better we also had one other approach. First we ran the network using all the originals lambdas, but this took a long time to complete due to the fact we had 7 lambdas to run through. As a result, we decided to eliminate some lambdas that were not causing the network to produce good results. One way we discovered lambdas that were bad was by plotting the validation and test accuracies. Since we have N epochs and 7 original lambdas, the total number of x values would be 7N. By plotting these values, we could see where there were bigger dips in the validation and test accuracies since the x axis can be split into groups that were around size 7. Another use of the graph was to determine if our network was performing just as well on the test data as on the validation data.

This graph represents the best test and validation accuracies achieved by the network using 20 Epochs with the Leaky ReLU. Both the test and validation accuracies are relatively close to each other, resulting in very little overfitting.



We also tested our networks using images of size 101 by 101. These were also able to produce very good validation accuracies up to 99.35%, but test accuracies around 96 to 97% and sometimes 98%. The issue with images this size was that it took a long time to classify images using the sliding window approach (discussed further down). This is because we were required to keep scaling the 1700x1700 image up in order to classify smaller craters and have them fit in the 101x101 sliding window size. However, this was good when we were looking for bigger sized craters. On the other hand there were a lot more smaller craters than larger craters, so this is where the issue with speed came into play.

The graph below represents the best validation and test accuracies from the 101x101 network with 25 Epochs, an Eta of .0075, the Leaky ReLU, along with the best lambda that was found.

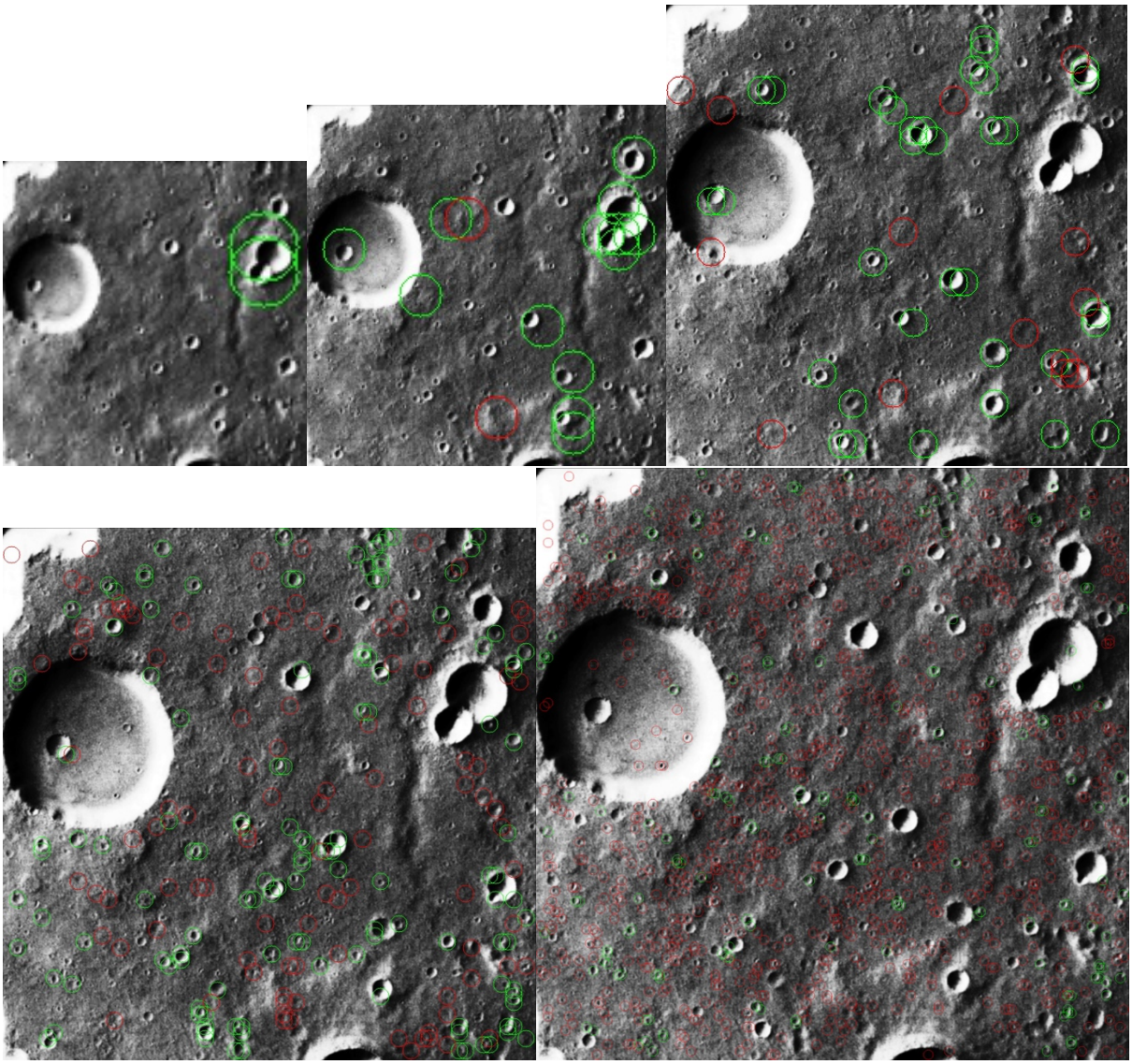


Next, using some of our best trained models from our network, we set out to implement a sliding window to scan the original larger crater images. Using a logarithmic function (shown below) computing scaling factors, we scanned the scaled images using 5 values of *i* (-4 to 0), which effectively resulted in scaling windows ranging from 28x28 to 387x387. These numbers were targeted to capture every possible size crater found in the dataset as well as scaling images in a manner consistent with how our training data was scaled.

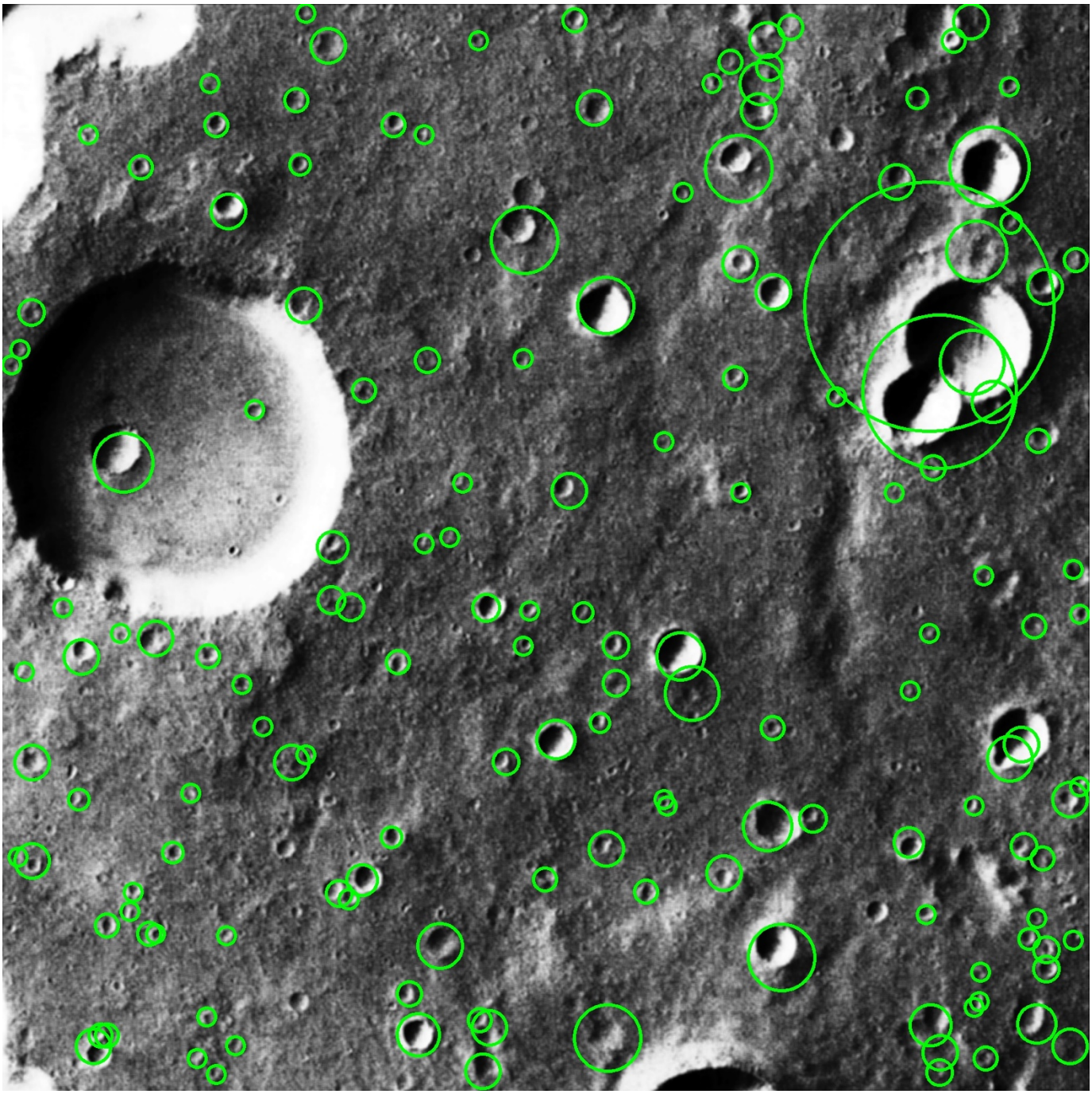
$$scalefactor_i = e^{.657 \times i}$$

In terms of differentiating True Positives vs False Positives during the image scanning process, each crater and each "hit" (or classification as a crater) was modeled as a circle. The crater circles were given by the ground truth csv files and a "hit" circle was determined as the center point of the scaling window and $1/4^{th}$ of its width, both divided by the current scale factor to normalize the "hit" with the given ground truth data. If a "hit" intersected with a crater, the positive identification was considered a true positive, otherwise it was considered a false positive.

We found that larger our image got scaled to (and in turn smaller our effective window size) got, the more false-positives that were produced by the network. The next few images show our results as the image sizes increase.



The next picture shows the combined averages for each crater positively identified across all scales of the original image.



As shown, the network performed reasonably well for some craters, outlining them accurately, however some outlines were skewed to one side, or to small or too big and some craters were not detected at all. The following is a table of our results

Tile_3_24 total craters present	185
Craters Detected	157
False Positives per Scaled image	208.2

Tile_3_25 total craters present	214
Craters Detected	189
False Positives per Scaled image	185.2

TOTAL craters present	399
Craters Detected	346
False Positives per Scaled image	393.4

False rate	53.2%
Detect rate	86.7%
Quality rate	43.7%

Phase 1 results:

False rate	6.7%
Detect rate	95.8%
Quality rate	89.6%

Compared to phase 1, it seems that the Convolution network did not perform well. However, we believe that it does perform a lot better than the network from phase1 because we could produce validation accuracies up to 99.35%, and test accuracies up to 98%. The reason for these differences between phase 1 and 2, is that the data above is produced from the sliding window. This is because of the following:

As shown above, we could detect a majority of the craters, however with very poor quality, most notably when the images were scaled up. We believe this is due to the amount of detail or noise that is visible to the sliding window with large size images, it appears despite high validation and test accuracy during training, the network had trouble differentiating small craters and surface features. In the future, to mitigate this issue, we would consider altering the contrast of the sliding window images and possibly the training, validation and test data as well to eliminate some of the noise when zoomed in.