Term Project Phase I Report

Apply Neural Network to Real World Application (Crater)

CS 480

Our goal in this project was to implement a neural network with optimal parameters such that our network could recognize a crater in an image. A dataset of crater and non-crater images was supplied to us, as well as sample code used to train a neural network to recognize handwritten digits.

Our first task in approaching the problem was deciding between the two types of preprocessing and normalizing methods. We initially decided to utilize padding as the preprocessing network. But when we began training our network with the normalized and padded dataset, we quickly began to notice unpredictable behavior when attempting to analyze and assess the parameters. We implemented code to create a directory after a test’s final epoch in which images of both False Negatives and False Positives were duplicated. What we noticed right away is that most images that were failing to be detected properly were those such that dimension of the original image was smaller relative to the normalized size of 200 x 200 pixels. We concluded that the MINMAX method might produce better results, and it was clear to see that padding was causing difficulties in attempting to optimize our parameters.

Next, after implementing a loading script for our normalized data, we sought out to find optimal parameters to train our neural network to produce the best results. We focused on a parameter at a time, and noticed first that as the value for epochs rises the results improve. We decided on training our network with a small epoch’s value due to time constraints, but planned on doing a final run for a large value of epochs once we determined our most optimal hyper-parameter values.

We realized that lower batch sizes were producing far better results that larger batch sizes. Our analysis leads us to think this would make sense considering the dataset we are using isn’t that large relative to say for example the handwritten digit dataset. We decided on using batch size of both 1 and 2 on the majority of our tests. Batch size 2 would for the most part produce better results than batch size 1 on some parameters and tests but this was not always the case and we therefore never dismissed either value (1, 2).

We also noticed that the higher the number of neurons in the hidden layer, the more consistent our values would be (less deviation from the mean) and the closer the value of the Quality Rating was to our Detect Rating. We figured that if our Quality Rating was optimal, the Detect Rating could be even better. This would make sense considering the first layer contains 40,000 neurons. Mathematically, to go from dimension 40,000 to say 50 or 100, could produce a great loss in desired computational value. We began testing using 400 or 500 layers, although we wonder if time was not limited if a hidden layer size of even 2000 would produce better results from our network.

Final we sought to find the optimal eta value.