Lab 7 Report: Digits

by Cherise Malisa

The data used for this training was taken from The MNIST Database, the following is a link to the website: http://yann.lecun.com/exdb/mnist/.

The data consists of handwritten digits, with an already separated set of training set and test set, of size 60,000 and 10,000 respectively. The images of the digits have been size-normalized and centered in a fixed size. The original black and white images were normalized to fit in a 20X20 pixel box, the resulting images contain gray levels. The MNIST training is composed of 30,000 patterns from SD-3 and 30,000 from SD-1, the test was composed of 5,000 patterns from SD-3 and 5,000 patterns from SD-1.

Part 2:

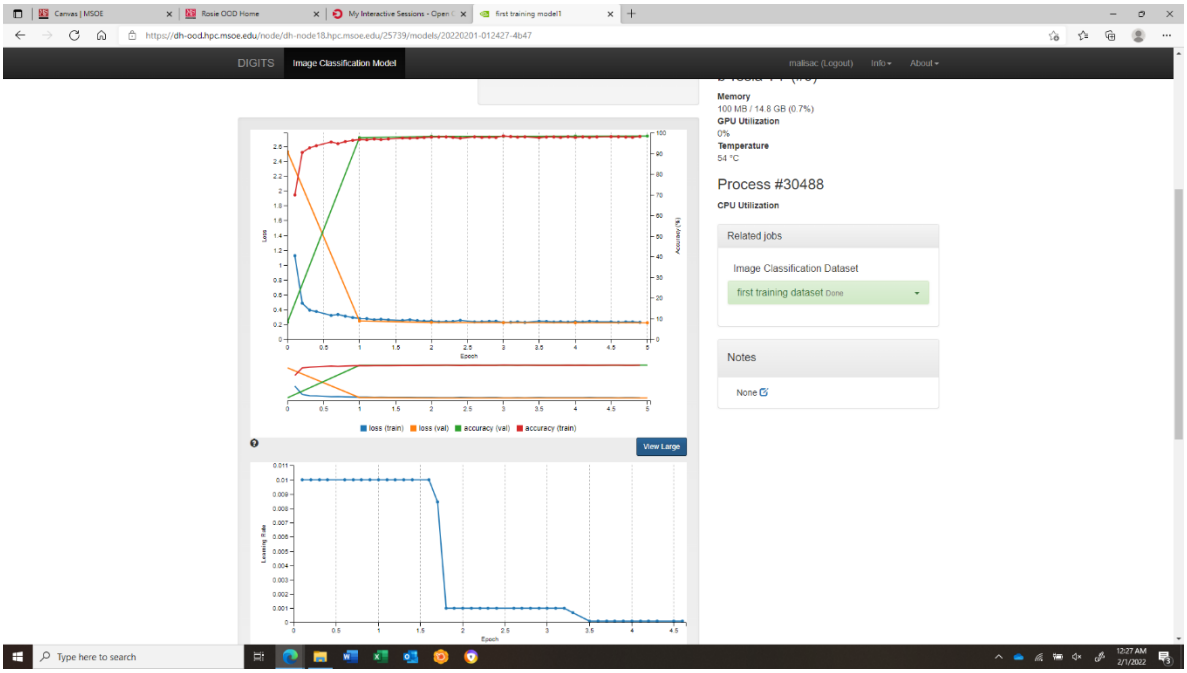


Figure 1: Contains the first training model done.

Part 3:

for this part of the lab one image was classified and the trained model was tested, from the set chosen. The image was of a one and the predictions were very accurate with the top one being a 99.91% of the number being identified as a one. A screenshot of this is pasted and can be viewed below.

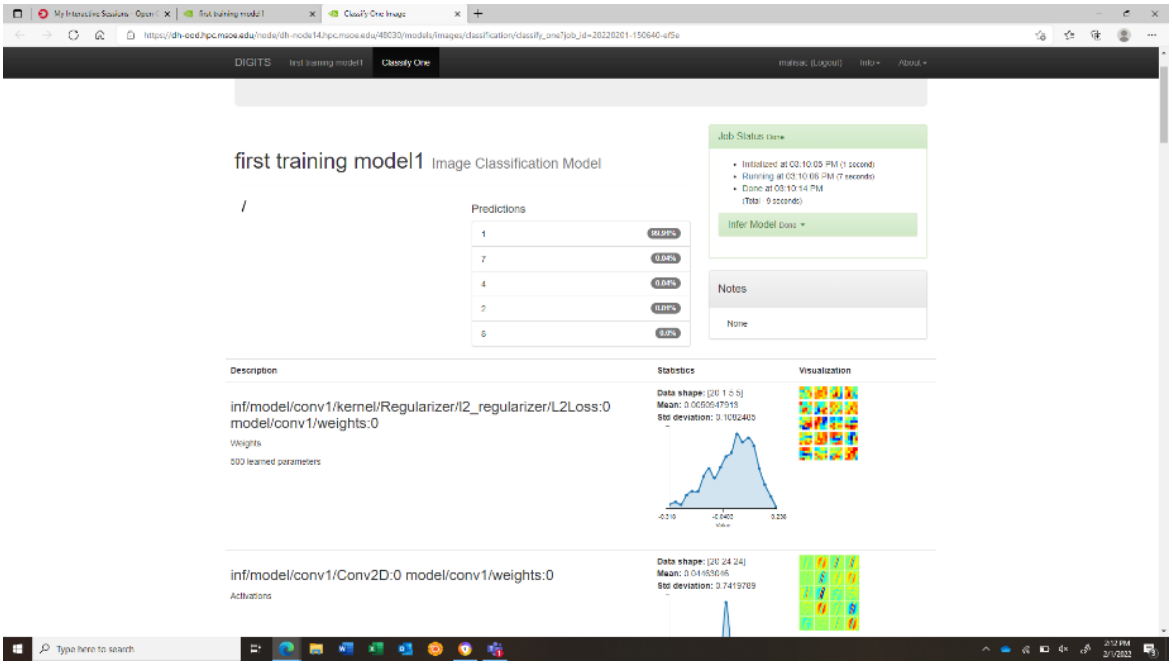


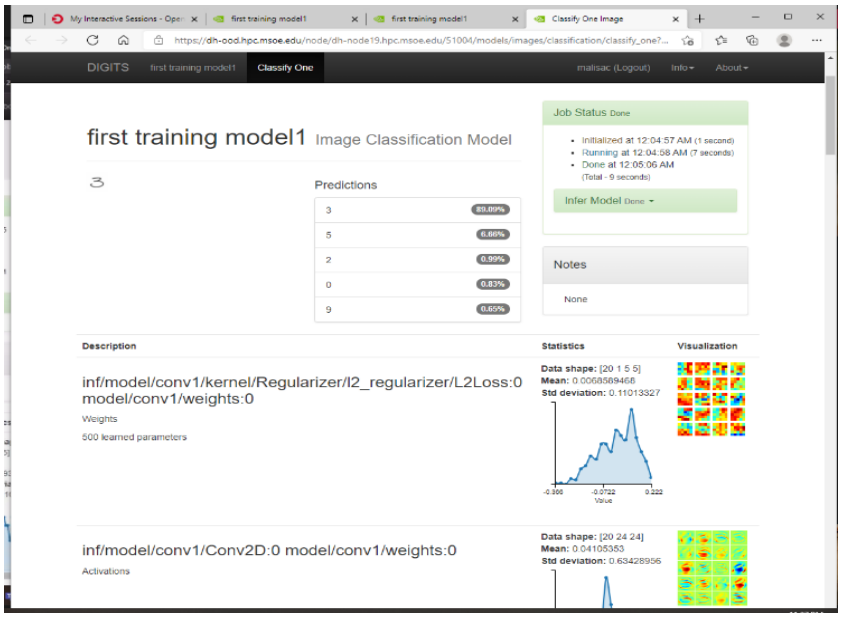
Figure 2: This is a screenshot of the first image tested on training model one which had 5 epochs.

I think the result is very correct, even though the image looks like an italicized tick, it also looks very much like a 1 so it is not surprising that the result was very accurate.

From online number:

The following image contains the results of the handwritten number 3 that I downloaded from the internet and its predictions from the model.

In this one it did well, the prediction that the number is 3 is very high. I think that is because the 3 given looks very similar to the



3 that was used for training as the 3 in the image looks very similar to the 3 in the image predictions box. The percentage prediction that the number in the image is a 3 was an 89.09%. Much lower than the prediction from a dataset that it was trained by which is reasonable, the

prediction is very high still.

Part 4:

In this part of the training, I created a new model that only trained for a single epoch. The images tested were the same as the ones mentioned in part 3 of the document.

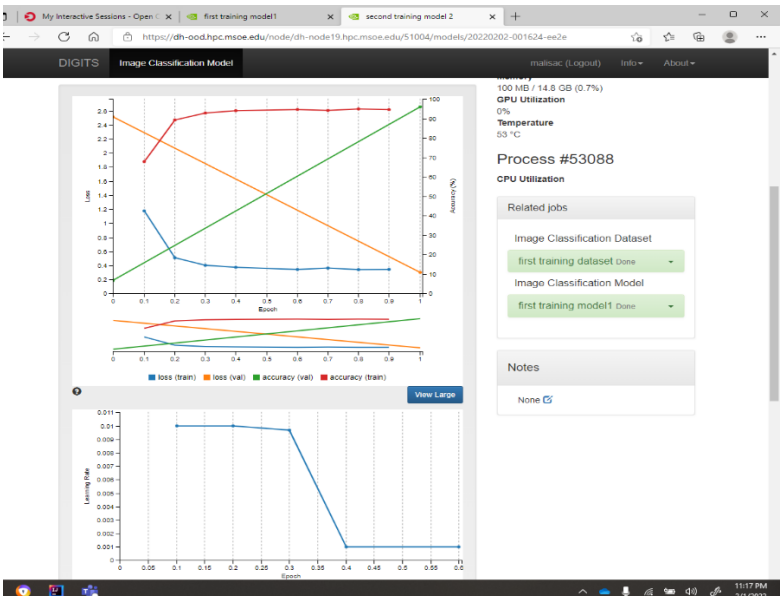
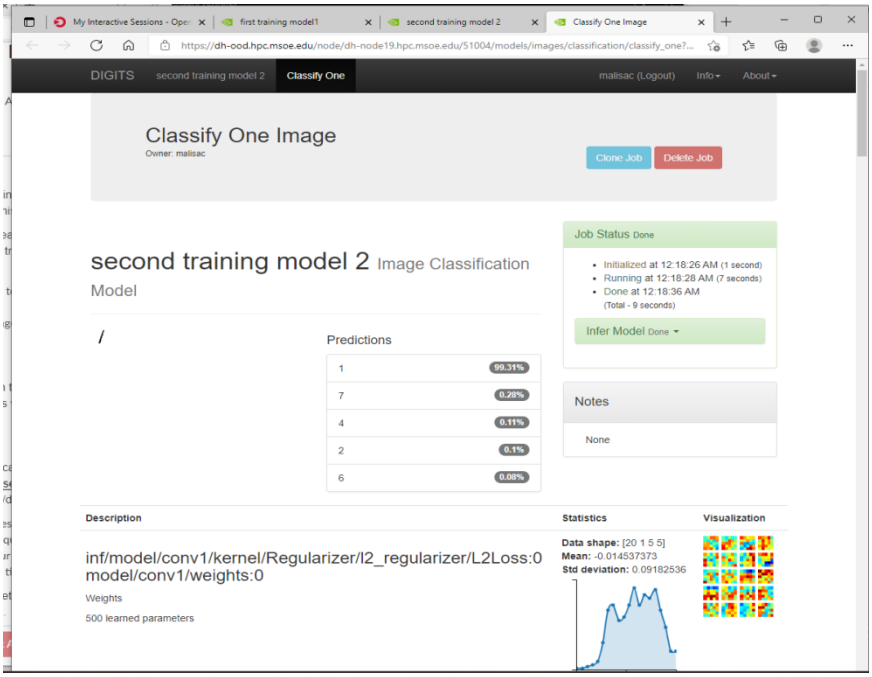


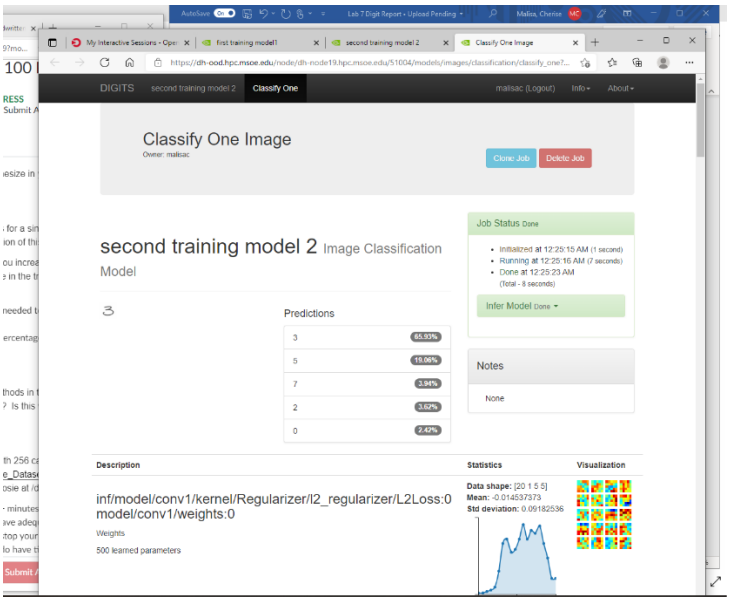
Figure 3: The following contains the data from the model that was trained with one epoch.



The result from using a single epoch was still very successful and the prediction that it is a 1 was 99.31%. which is a slight drop from the prediction made when the

epoch was 5.

Figure 4: Is an image of the number 1 predictions from the dataset that was used to train the model



The prediction was a lot lower this time than what it was when the epoch was 5 and not one. The prediction value was at, 65.93% for the number 3. I think this is because having it train once means its accuracy decreases for predictions of images that it did not use to train with such as the one in the picture. Hence the major drop in prediction value.

Figure 5: Is an image of the predictions made for the number 3 that was downloaded from the internet.

- What is the effect on the number of epochs needed to train?

  -It is no effect to the number of epochs needed to change.

- What is the resulting effect on accuracy?

  -The accuracy decreased significantly, but the training accuracy decreased noticeably.

- What do you notice about GPU utilization (percentage)? Why might this be affected?
  - The GPU utilization was much faster than when the epoch value was 5, this is probably because it was only used once when the epoch value was now one.

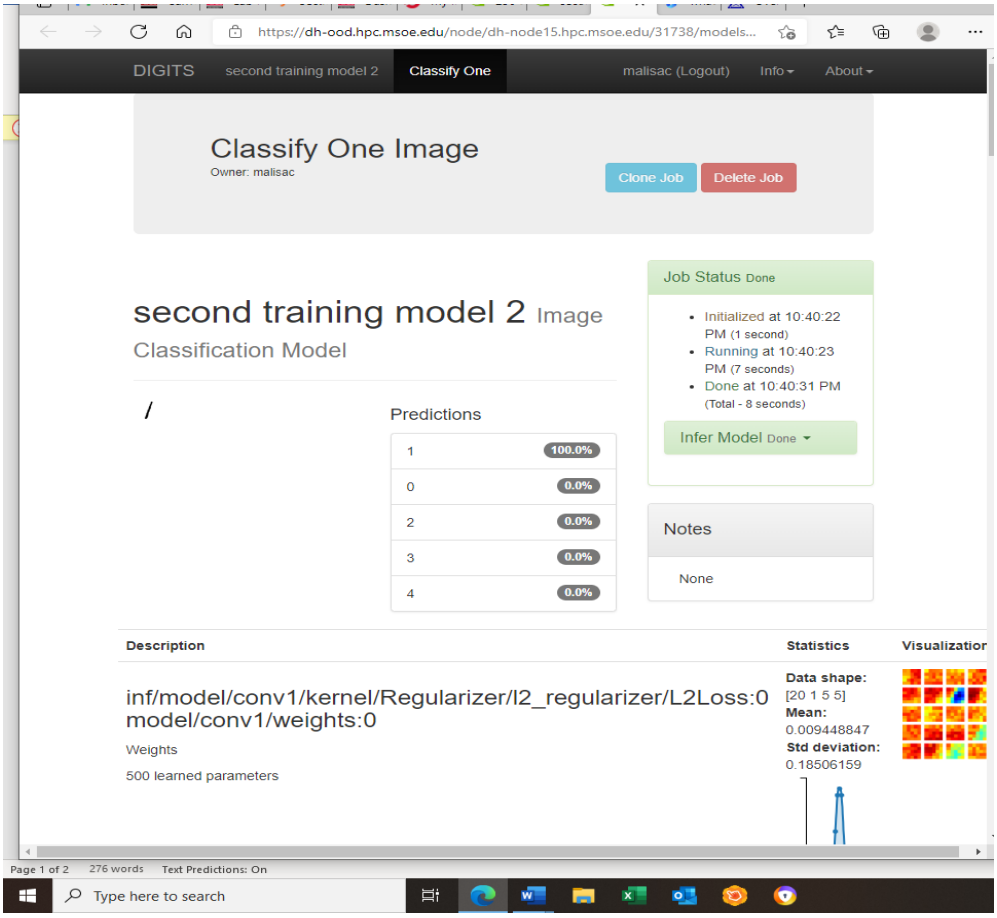
Part 5:

What is the effect on the network? Is this what you expected? Does this help with more "realistic" images?

In this part of the lab, I had to test and see the results of adding augmentation to and how those affected the network.

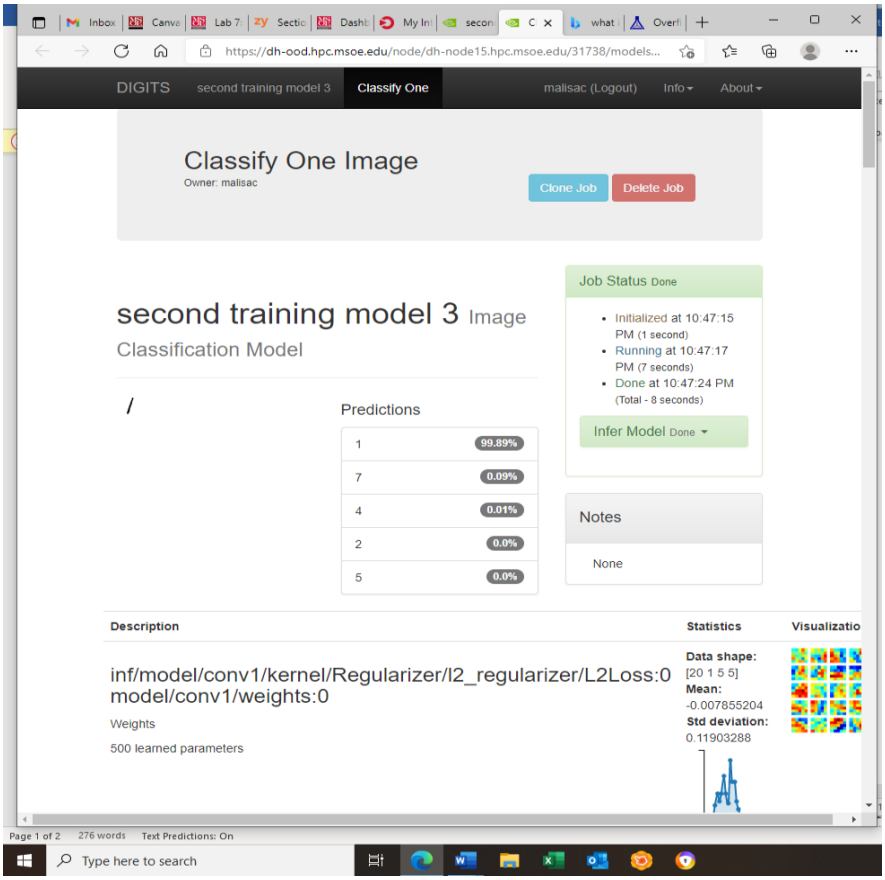
The following is an image from an experiment that had 5 as the epoch size and leNet network, the augmentation done to this data was whitening. And the effect on the network was good the prediction

percentage was the highest for the experiment. I tested the image of the number one and it predicted 1 100%. It is not what I expected so I was surprised in a good way. The following image has the evidence of the information mentioned.



The next image has the same parameters and the augmentation I chose for this was the vertical ….

And the effect on the network were slight. The prediction decreased to 99.89% from the prediction outcome with no data augmentation.

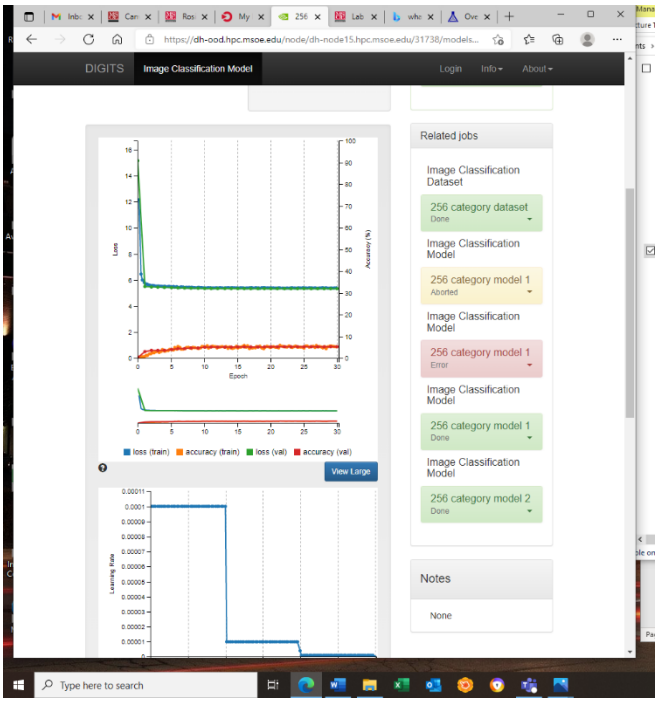


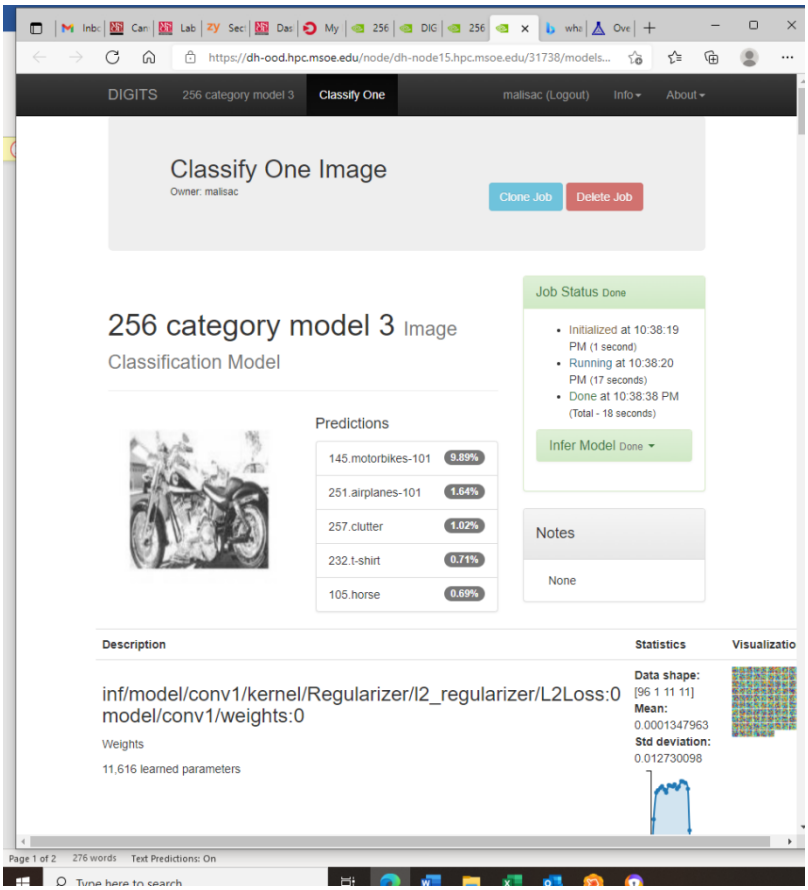
Part 6:

In this section a new data set was used, and this data set contains 256 category classifications. Hyperparameters where altered for each model run to try and get the model to converge. Below is a table of the different parameters used and the results acquired.

My data set was a grayscale 256X256 category classification, with 128 for the batch size, the learning rate was 0.0001 and the number of epochs was 30. I did not add a data augmentation because it distorted my first try. The accuracy curves started at 0 and increased slightly with the number of epochs, overall, the accuracy level was extremely low as compared to the first dataset that was experimented on. The loss curves started off high and plunged, then remained consistent throughout the rest of the classification.

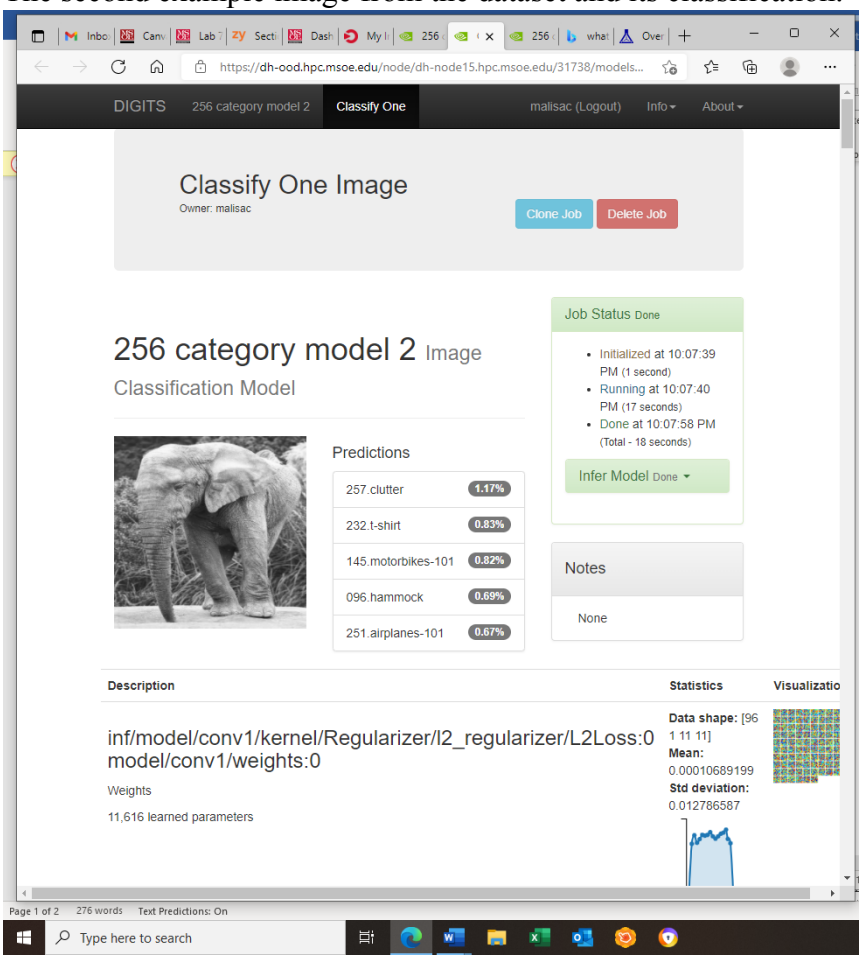
Below is a screen shot of the data set described above:





one example image and its classifications.

The second example image from the dataset and its classification.



The time taken to train the data set was 36 minutes. What I observed is that the success of the classification is based on the dataset and how it was trained and arranged and set up. No matter how many parameters you change it is difficult to find the perfect balance to get accurate results.