Jake Foglia EE 551 Final Project Report

**Problem:**

Train a neural network to differentiate between pictures of cats and dogs.

**Solution:**

TensorFlow for python allows for the easy creation and training of machine learning models. I used the GPU version which has a CUDA C dependency. Essentially, it used my Nvidia GPU in order to accelerate the process of training the model. I found a [dataset](https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition/data) online with roughly 25,000 labeled images of cats and dogs. The python code I wrote generates two parallel arrays for the training set. One array holds the pixel information of each image (images were rescaled to fixed dimensions for consistency of input into the model). The other array holds Booleans of whether the image is a dog or cat. These Boolean values were determined based on the label contained in the image file names.

The model generation function has a few parameters. The first two parameters are lists containing the sizes of any number of convolutional layers, followed by the sizes of any number of dense layers. I found in my testing that [256, 128, 64] and [512] were relatively good values in terms of yielding a strong model. This meant that the neural network would contain a convolutional layer of size 256, followed by another of size 128, followed by another of size 64. After those layers would be a dense layer of size 512. The other parameter is the training data which I mentioned in the previous paragraph, that contains pairs of image data with their corresponding dog/cat booleans. Inside of this model generation function the model is trained using a variety of TensorFlow functions, which I relied on a YouTube series by sentdex for reference. The model is then saved as an .h5 file and stored in the logs directory. The model the gets returned to the main function and then the mail while loop is entered.

In the main while loop, the user is repeatedly asked to input a number corresponding to an image in the test directory, that they’d like the model to predict whether it’s a cat or dog. The program then outputs its guess: cat or dog, along with a confidence level.

**Highlights:**

In my experience, the confidence levels seemed to correspond well with how ‘easy’ of an image I thought it would be for a computer to predict. For example, for an image where the entire body of the animal is visible with its face pointed at the viewer, the model would typically have high confidence. On the other hand, the model would have low confidence and might guess wrong for an image containing a partially occluded animal or one that was shot at a strange angle.

Overall, the model had an accuracy value of 98.77% on the images it trained with. However, a more realistic accuracy level is the validation accuracy. Essentially, during the training process TensorFlow reserves some of the labeled training data strictly to test with. So the accuracy for those images is more realistic since the model had never seen those particular images before in the training process. The validation accuracy ended up being 82.13%, so roughly that percentage of real-world images would be predicted correctly by the model, assuming the image is of comparable quality to the images in my training set.