Kristina Poon

Robotics and Computer Vision: Homework 4

When **loading in MNIST**, the first command that imports MNIST from Keras' datasets is a database of handwritten digits that is comprised of a dataset of 60,000 grayscale images and a test set of 10,000 images. The two tuples returned include the both trained and test images and labels. To decide on a proper label y for an input image x, the classifier function must be defined. The commands found in the **Training and Test Data** appear to just show the details of what the training images and training labels are like in terms of size. For instance, train_images.shape returns three parameters describing that it contains 60,000 28x28 pixel images and there are 60,000 training labels, as shown by the following line. Afterwards, one is able to distinguish that there are 10,000 28x28 pixel test images and 10,000 test labels.

In the process of **setting up the network**, the architecture of the network is established. The first five commands are creating the neural net by adding in layers one at a time. The first parameter of Dense layers describe the output size of the layer; the connections between the layers are handled by Keras. The network is then configured by compiling it. In particular, the compiling method described in the command is for multi-class classification. The following commands preprocess both the input data and class labels prior to fitting the model network on training data in the **Train the Network** section of commands. This process is supervised learning since such a function is produced from imported labeled training data. Each training example has an input object and a corresponding output that a supervised learning algorithm analyzes to procure an appropriate classifier function. The classifier function should then, hopefully, be able to map new examples for inputs outside of the training set. Within the actual commands, the both the training images and training labels are being ran through to train the classifying function where they attempt to find parameters that minimize error e in $e = \Sigma[Y - f(X)]^2$. The mappings generate a multilayer neural network that traces attributes of inputs and selects the correct identifying output based on its training data. The evaluate function returns the loss value and accuracy value for the tested model. The lower the loss value, the better the model. Accuracy that is closer to 1.0 is desired; the printed value at the final command is the accuracy of the model at the end of the training.