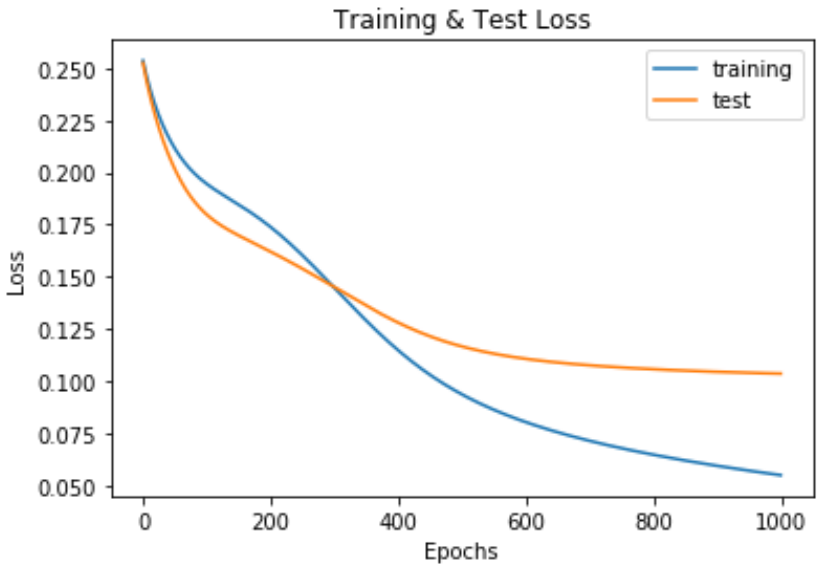
Assignment 5 Report

Description:
This assignment asks to implement the Back Propagation algorithm for Feed Forward Neural Networks to learn the down gestures from training instances available in the downgesture_train.list. Then this trained network is to be used to predict the labels for the gestures in the test images available in downgesture_test.list. For the error function, the assignment asks us to use the standard squared error and to output the predictions and accuracy on the test images.

I begin this assignment by writing a preprocessing function to read in the pgm image files (using library functions from opencv2) and prepare them as a matrix to be fed into the input layer of the network. Preliminary data exploration revealed that the dimensions of each image were 30x32 which implied our input layer would be of size 961 since we would have N images by 30x32 pixel features plus the 1 pad term for logical completion. In addition to this, the preprocessing phase also builds the appropriate labels using a list comprehension and a conditional mapping of 1.0 to filenames containing the indicator "down" and 0 .0 for all other filenames per the instructions.

After having processed the training and test data and building their respective labels, I write a class for my feed forward network. All weights are randomly initialized with values between -0.01 and 0.01 and given the appropriate dimensions as numpy arrays. Then a few helper functions are written in for padding, sigmoid activations for the perceptrons in the forward pass, and the sigmoid derivative for the back propagation. The forwardfeed method of the network is then implemented by carefully tracing the dimensions of the input (184,960) along with the needed "x0 = 1" column padding at each layer, ultimately returning a vector of probabilities from the final activation of the output perceptron. Finally, the backpropagation method is implemented by using the ½ squared error loss function so that the square term cancels out in the derivative. The cost updates are then computed and stored in costgrads 1 & 2 and applied to each respective layer's weight parameters after scaling by learning rate eta. The main challenge I dealt with in this assignment was tracing the dimensions of the input through the forward pass and also ensuring that the dimensions of the cost updates in the backward pass were all aligned appropriately for the hidden layer of weights.



After these were implemented, I then added train and prediction methods for the network and trained the network for 1000 epochs with a learning rate of 0.1 using numpy random seed 16 for the weight initialization. I also experimented with alternative initialization strategies such as using a standard gaussian, but ultimately I felt that the uniform initialization tended to see better performance. Since the input X matrix is constructed such that it contains the entire training set of images on input, each epoch of training is achieved by simply calling the back propagation method of the network within a for-loop for as many epochs as desired (in this case 1000).

After the training, I used the predict method to return the network's 1-0 predictions on the test images and found that the network achieved 86.75% accuracy on classifying the test images. Additionally, I used pandas for a confusion matrix to assess the extent of type 1 and type 2 errors. The network appears very good at identifying the negative class 95% of the time (61/64), but only gets ~58% of the positive class (11/19).

Requested Output:

```
Accuracy:  86.75 %
NN Predictions:
 [[1. 1. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0.
  1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 1. 0. 0. 0. 0. 1. 1. 0.
  0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0.
  0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0.]]
Test Labels:
 [[1. 1. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 1.
  1. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0. 1. 1. 0.
  0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0.
  0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0.]]
```

Predicted	0.0	1.0	All
True Label			
0.0	61	3	64
1.0	8	11	19
All	69	14	83