Kade Boltjes | kboltjes@purdue.edu

Here is the link to my git repo: https://github.com/Kboltjes/CS390Lab1

Here are the links I used during this lab:

- https://medium.com/ai%C2%B3-theory-practice-business/a-beginners-guide-to-numpy-with-sigmoid-relu-and-softmax-activation-functions-25b840a9a272
- https://keras.io/guides/sequential_model/
- https://keras.io/api/losses/

Completed

The parts of the lab I have completed are:

- Custom Neural Net
  - Has working sigmoid and sigmoid derivative functions
  - Training functions gets greater than 85% accuracy
  - Has a fully working 2-layer neural net
  - EC – all completed here
    - I implemented it so it is N-layer, where N can be 2, 3, or greater.
    - I implemented ReLu
- TF Neural Net
  - Implemented a Keras 2 layer neural network
- Pipeline & Misc.
  - Preprocessed image data values to be in the range [0.0, 1.0]
  - Created confusion matrix that gets printed after accuracy

Summary

I made the 2-layer neural network by following the class slides in calculating the layer adjustments from the layer deltas and errors. That adjustment is done for every batch during each epoch.

To convert that to an N-layer neural network, I converted the layer outputs, weights, and adjustments to be arrays where they are of length N. For the backpropagation of it, I start by calculating the all the layer outputs, then the final output's error. After that, I go backwards and calculate the layer adjustments from the layer output to the left and its layer delta until I reach the input layer. Then I use the input X instead of trying to get output from layer -1 to calculate the adjustment for the first layer.

For the Keras neural network, I started from the class slides. I wasn't sure if the Keras Flatten() layer counts as a layer, so I just used two dense layers with sigmoid activation functions. After that, I just looked through Keras' documentation to try to improve the accuracy.

Outputs

If you want to run any of my code, I have a section above all my code that allows you to alter global variables like NUM_EPOCHS_CUSTOM_NET. Those globals are fed directly into the constructors of my classes, so you won't have to try to find where to make the changes.

Here is an example of my Keras neural network output

```
Epoch 10/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.0043 - accuracy: 0.9965
Testing TF_NN.
Classifier algorithm: tf_net
Classifier accuracy: 98.310000%
Confusion matrix:
        truth-0  truth-1  truth-2  truth-3  truth-4  truth-5  truth-6  truth-7  truth-8  truth-9
pred-0      974        0        6        0        1        2        5        1        4        1
pred-1        0     1125        0        0        0        0        2        3        1        2
pred-2        1        2     1012        2        1        0        1        8        4        0
pred-3        1        1        2      996        0        9        1        1        4        5
pred-4        1        0        1        0      965        1        1        0        4        6
pred-5        1        1        0        2        0      872        3        0        3        1
pred-6        0        2        2        0        6        4      944        0        1        1
pred-7        1        1        5        5        1        2        0     1011        2        6
pred-8        1        3        4        2        0        1        1        0      946        1
pred-9        0        0        0        3        8        1        0        4        5      986
```

Here is an example of my neural network running with 2 layers

```
Epoch 200 of 200
Testing Custom_NN.
Classifier algorithm: custom_net
  with 2 layers
Classifier accuracy: 87.520000%
Confusion matrix:
        truth-0  truth-1  truth-2  truth-3  truth-4  truth-5  truth-6  truth-7  truth-8  truth-9
pred-0      966        0        8        2       19        5        7        0        5        4
pred-1        1     1123        1        1       14        0        3        7        1        5
pred-2        0        2      984        8       69        2        6       17        7        2
pred-3        3        1       12      971       33       21        1        2        9        9
pred-4        0        0        0        0       68        0        0        0        0        0
pred-5        4        2        3       11       53      843        7        2       12        6
pred-6        5        4        4        1      151        6      931        0       10        2
pred-7        0        1        7        6       92        3        1      981        8        9
pred-8        0        2       12        7       53        8        2        7      918        5
pred-9        1        0        1        3      430        4        0       12        4      967
```

Here is an example of my neural network running with 7 layers

```
Epoch 30 of 30
Testing Custom_NN.
Classifier algorithm: custom_net
  with 7 layers
Classifier accuracy: 92.550000%
Confusion matrix:
        truth-0  truth-1  truth-2  truth-3  truth-4  truth-5  truth-6  truth-7  truth-8  truth-9
pred-0      948        0       15        5        3        9       14        2        7        4
pred-1        0     1116        3        4        3        4        2        7        2        6
pred-2        2        3      938       21        3        1        7       13       17        2
pred-3        2        3       16      903        0       36        0        3       19        9
pred-4        3        0       17        0      930        5       11        6       12       39
pred-5       11        0        7       33        1      797       20        1       22       15
pred-6        5        2        7        1       11       16      898        0       11        3
pred-7        5        2       10       13        0        3        1      964        8       30
pred-8        0        8       19       23        4       14        5        6      868        8
pred-9        4        1        0        7       27        7        0       26        8      893
```