CS390 FALL 2021

Lab 1

Name: Divay Gupta

Email: gupta576@purdue.edu

Repository Link: black-tul1p/CS390

Completed Lab Sections:

- Custom Neural Net
  - Accuracy – 93.23%
  - Implemented the sigmoid and sigmoid derivative functions.
  - Implemented the train function using backpropagation properly.
  - Fully functioning 2-layer neural net with minibatches of size 100.
- TensorFlow Neural Net
  - Accuracy – 98.27%
  - Implemented a 2-layer neural net using Keras
  - Used the sigmoid and SoftMax activation functions, in that order
  - Used the Adam optimizer
- Pipeline
  - Normalized image data values from 0-255 to a range of 0.0-1.0
  - Implemented a F1 score table and a confusion matrix with accuracy values

Resources Used:

- **Lecture Slides**
  - Slides 2 – Neural Network and Backpropagation basics
  - Slides 3 – Backpropagation and TensorFlow/Keras
  - Slides 4 – Tuning hyperparameters
- **API Documentation Links**
  - TensorFlow
  - Keras
- **Miscellaneous Links**
  - Backpropagation using Python
  - One Hot Encoding
  - MNIST Image Classification using Keras
  - What is a Confusion Matrix in Machine Learning?

Lab Implementation Summary:

- **Guesser Algorithm**
  - This algorithm just randomly guesses the labels with a set seed of 1618 for deterministic behavior and it achieves an understandably low accuracy of 9.68%.
  - The confusion matrix and F1 score table for this algorithm are:

```
Classifier algorithm: guesser
Classifier accuracy: 9.680000%

############################## Confusion Matrix ##############################
       0     1     2     3     4     5     6     7     8     9
0     96   107    95   104    96    97   103   102    83    97
1    106   101   105   138   110   103   113   130   114   115
2     90   110   100   110   109   127    97    84    96   109
3     99    96    89    99    93    95   116   102   104   117
4    107   107    87    88    95   100    80   120    97   101
5     88    80   102    69   109    93    81    81    97    92
6     92    97    94    98   106    82    89    87   100   113
7    110    98    97   107   108   103    99    96   114    96
8    114    99   105    75   102    91    89    93   104   102
9     89    84    85   120   121    99   118   108    90    95

############################### F1 Scores ################################
      0      1      2      3      4      5      6      7      8      9
[0.0974 0.0956 0.1005 0.0981 0.0935 0.0988 0.0916 0.0945 0.1054 0.0929]
```

- **Custom Neural Net**
  - The custom neural net was implemented with 2 layers – one hidden layer and an output layer. The sigmoid function used was implemented and used as the activation function. Backpropagation was correctly implemented and the outputs of the network are then one-hot encoded to get the predictions.
  - The neural net was trained for 30 epochs at a learning rate of 0.01 with a 100 mini-batches and was able to achieve an accuracy of 93.23%.
  - The confusion matrix and F1 score table for this neural net are:

```
Testing Custom_NN.
Classifier algorithm: custom_net
Classifier accuracy: 93.230000%

############################## Confusion Matrix ##############################
       0     1     2     3     4     5     6     7     8     9
0    965     0     0     3     1     4     6     0     0     1
1      0  1121     2     1     1     2     4     0     3     1
2      8     9   942    25     7     2    11    10    16     2
3      1     3    15   936     1    23     1     8     9    13
4      1     1     7     1   927     1     9     0     3    32
5     10     6     0    17     6   830     7     0     5    11
6     13     3     4     0     7    14   910     1     5     1
7      2    15    21     5    14     9     0   916     1    45
8     15     9     9    21    11    39    11     6   839    14
9      7    10     2     5    23    10     1     8     6   937

############################### F1 Scores ################################
      0      1      2      3      4      5      6      7      8      9
[0.964  0.9697 0.9263 0.9249 0.9364 0.9091 0.9489 0.9267 0.9017 0.9071]
```

- **TensorFlow Neural Net**
  - This neural net was implemented using Keras, with two layers – one hidden layer and an output layer. The hidden layer has 512 neurons (same as the input image size) and used the sigmoid function as an activation function and the output layer has 10 neurons (same as number of classifiers) and used the SoftMax function as its activation function. Those two activation functions were chosen since that combination gave the lowest loss and hence the best accuracy. The Adam function was used for gradient descent and the categorical cross entropy function was defined as the loss function due to its high performance in categorical data classification. The outputs of the network are then one-hot encoded to get the predictions.
  - The neural net was trained for 20 epochs at a learning rate of 0.001 and was able to achieve an accuracy of 98.27%.
  - The confusion matrix and F1 score table for this neural net are:

```
Testing TF_NN.
Classifier algorithm: tf_net
Classifier accuracy: 98.270000%

############################## Confusion Matrix ##############################
       0     1     2     3     4     5     6     7     8     9
  ┌──────────────────────────────────────────────────────────────┐
0 │   972 |   0 |   1 |   0 |   0 |   1 |   3 |   1 |   2 |   0 │
1 │     0 | 1127 |   1 |   1 |   0 |   1 |   2 |   1 |   2 |   0 │
2 │     4 |   2 | 1014 |   2 |   1 |   0 |   3 |   3 |   3 |   0 │
3 │     1 |   0 |   3 | 998 |   0 |   3 |   0 |   1 |   1 |   3 │
4 │     1 |   0 |   1 |   1 | 966 |   0 |   4 |   0 |   2 |   7 │
5 │     2 |   0 |   0 |   8 |   2 | 872 |   3 |   1 |   3 |   1 │
6 │     5 |   2 |   0 |   1 |   1 |   5 | 942 |   0 |   2 |   0 │
7 │     1 |   5 |   7 |   0 |   1 |   0 |   0 | 1005 |   3 |   6 │
8 │     3 |   1 |   3 |   4 |   5 |   2 |   2 |   3 | 947 |   4 │
9 │     1 |   4 |   0 |   3 |   7 |   3 |   0 |   5 |   2 | 984 │
  └──────────────────────────────────────────────────────────────┘

################################ F1 Scores ################################
      0      1      2      3      4      5      6      7      8      9
[0.9868 0.9903 0.9835 0.9842 0.9832 0.9803 0.9828 0.9814 0.9758 0.9772]
```