**CSE 574**

**Project 2 Report**

**Group 7**

**Mihir Bose - mihirbos**

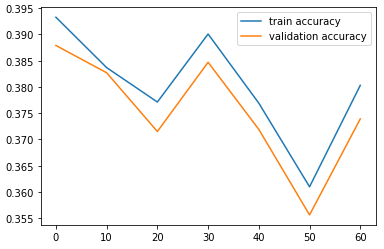
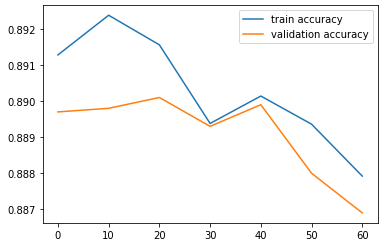
**Yonghui Lin – yonghuil**

**Xin Jiang – xjiang26**

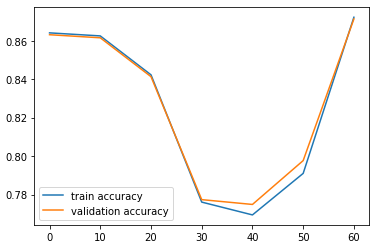
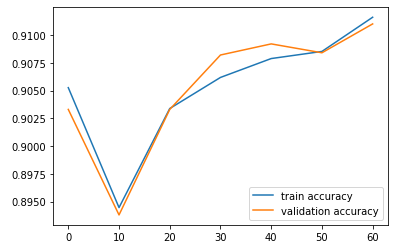
**A) MNIST Dataset:-**

Plots for MNIST dataset (from nnScript.py) :-

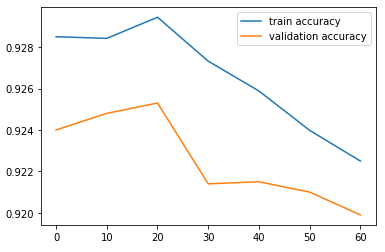
**Hidden units = 4** **Hidden units = 8**

**Hidden units = 12** **Hidden units = 16**

**Hidden units = 20**



Each of the above figures is a plot of training and validation set accuracy for different values of the regularization parameter lambda. The lambda values are shown on the X-axis, and have the values 0,10,20,30,40,50,60.

Also, each graph is plotted for a particular number of hidden units. The values of hidden units used are 4,8,12,16 and 20.

The set of hyperparameters (hidden units and regularization parameter lambda) to be chosen, will be the ones for which the prediction accuracy on the validation set is the highest. Only high training set accuracy is not enough, because then the model might be overfitting. We want the model to be such that it performs as well as possible on unseen data, hence high validation set accuracy is required.

From the graph, we can see that the highest accuracy of predictions for the validation set we get, is when:-

**Hidden units = 20**

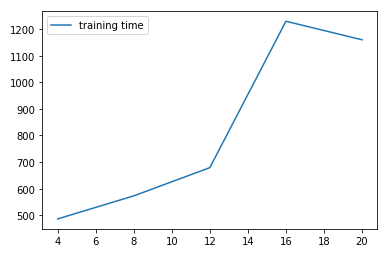
**Lambda value = 20**

The validation set accuracy at those parameters = **0.9253**

**Performance(training time) of network versus hidden units:-**

Shown below is the graph for the average training times of the hidden units.

Every point on the X-axis is a hidden unit value, and the y axis shows the average training time at that hidden unit, for different values of the regularization parameter lambda.



The curve for training times shows an increase as we increase the number of hidden units. However, it shows a drop after hidden units =16.

We can see that the average training time at hidden units = 20, which gave the best validation set accuracy between all combinations of hyperparameters, is high, being equal to 1159 seconds (almost 19 mins).

When hidden units are equal to 8, the training time is very low (573 secs, or 9.55 mins), and the max validation set accuracy is also good, being almost 89%.

|  |  |  |
| --- | --- | --- |
| Hidden units | Average Training Time | Max validation set accuracy |
| 4 | 486.41 secs (8.1 mins) | 38.79% |
| 8 | 573.04 secs (9.5 mins) | 89% |
| 12 | 676.19 secs (11.31 mins) | 87.13% |
| 16 | 1229.43 secs (20 mins) | 91.1% |
| 20 | 1159.74 secs (19.32 mins) | 92.53% |

However, we will choose the hyperparameter combination of hidden units = 20 and lambda = 20, because when the dataset size increases, this difference of 3.5% accuracy might magnify. Of course, the training time will also increase, but accuracy of prediction is of greater importance in machine learning models, especially when the models are trained offline, which is the case here.

Hence, the best set of hyperparameters:-

**Hidden units = 20**

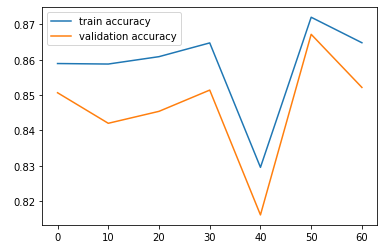
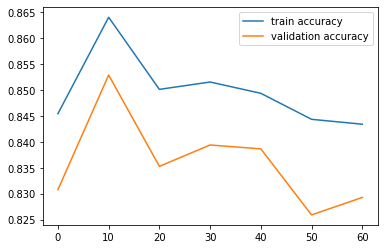
**Lambda value = 20**

The corresponding accuracy of classification method on the handwritten digits test data = **92.73%**

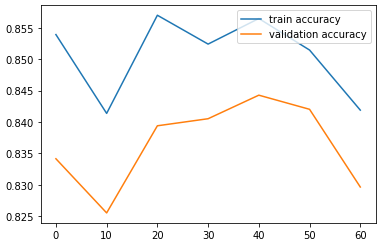
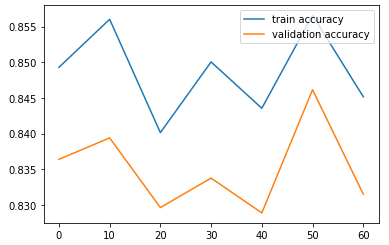
**CelebA Dataset:-**

Plots for CelebA (facennScripts.py):-

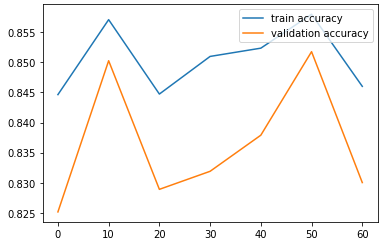
Hidden units = 4 Hidden units = 8

Hidden units = 12 Hidden units = 16

Hidden units = 20



Each of the above figures is a plot of training and validation set accuracy for different values of the regularization parameter lambda. The lambda values are shown on the X-axis, and have the values 0,10,20,30,40,50,60.

Also, each graph is plotted for a particular number of hidden units. The values of hidden units used are 4,8,12,16 and 20.

From the graph, we can see that the highest accuracy of predictions for the validation set we get, is when:-

**Hidden units = 4**

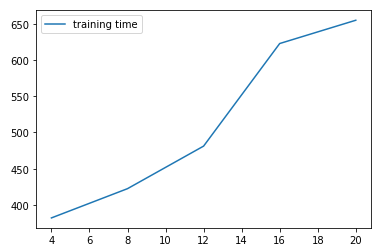
**Lambda value = 50**

The validation set accuracy at those parameters = **0.8671**

**Performance(training time) of network versus hidden units:-**

Shown below is the graph for the average training times of the hidden units.

Every point on the X-axis is a hidden unit value, and the y axis shows the average training time at that hidden unit, for different values of the regularization parameter lambda.



The graph shows a constant increase in training time as we increase the number of hidden units in the hidden layer.

|  |  |  |
| --- | --- | --- |
| Hidden units | Average Training Time | Max validation set accuracy |
| 4 | 382.20 secs (6.37 mins) | 86.71% |
| 8 | 422.54 secs (7.04 mins) | 85.29% |
| 12 | 481.22 secs (8.02 mins) | 84.4% |
| 16 | 622.67 secs (10 mins) | 84.6% |
| 20 | 654.78 secs (10.91 mins) | 85.1% |

For the CelebA dataset, trained on the implementation of a single hidden layer, both lowest average training time(6.37 mins) and the highest validation set accuracy (86.71%) occur on the same no. of hidden units = 4.

The exact training time at hidden units equal to 4, and the lambda value = 50, where the validation set accuracy is the highest is 408 secs (6.8 mins).

Hence, the best set of hyperparameters of CelebA dataset for the single hidden layer implementation in facennScript.py is :-

**Hidden units = 4**

**Lambda value = 50**

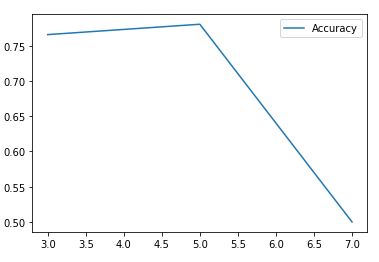
The corresponding accuracy of classification method on the CelebA dataset = **86.94%**

**Deep Neural network with Tensorflow (CelebA dataset):-**

In the file deepnnScript.py, an implementation of deep neural network with tensorflow is given. We will use 3,5 and 7 hidden layers to check the accuracy on test data, and also see the training time.

The number of hidden units we use in each layer, are the optimal number of hidden units we found from the facennScript.py file, which is equal to 4.

Since there are random initializations of weights when the script executes, we will get different results for both training time and accuracy each time the script is ran. Hence, the average value of both the results are shown.



The graph above shows the accuracy for the hidden layers.

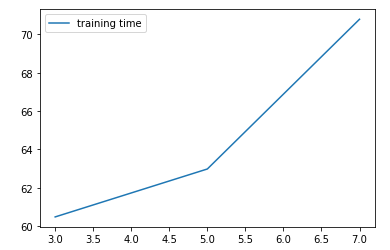
|  |  |
| --- | --- |
| **Hidden units** | **Accuracy** |
| 3 | 0.7657 |
| 5 | 0.7804 |
| 7 | 0.50 |

We can see that the accuracy increases when we use 5 hidden layers instead of only 3, however, if we increase hidden layers to 7, the accuracy decreases to just 50%. This shows the phenomena of overfitting. The network is following the training data much more closely, and is not able to generalize well on unseen data (test data).

Also note, that the accuracy of tensorflow neural network with 3 layers (**76.57%**), is lower than the stock implementation with just one layer in facennScript.py (**86.71%**). The possible reason for it could be that since we are also doing regularization in facennScript.py file, it has lower variance than the tensorflow implementation, hence, having a higher accuracy on the test data set.

**Training time:-**

The graph below shows the training time plotted against the number of hidden units.



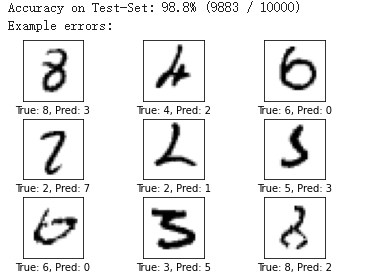
We can see that the training time is increasing as we increase the number of hidden layers in the network, which is as per expectation.

|  |  |
| --- | --- |
| **Hidden layers** | **Training time** |
| 3 | 60.48 |
| 5 | 62.98 |
| 7 | 70.79 |

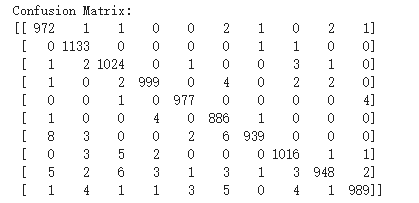
On comparing with our implementation of single hidden layer neural network, we see that the training time is much lower here. Even the network with 7 hidden layers takes only 70.79 secs (1.17 mins). Whereas, the single layer implementation with its optimal parameters of 4 hidden layers and regularization parameter equal to 50, takes 6.37 mins, which is almost 6 times more. Hence, the tensorflow implementation performs much faster.

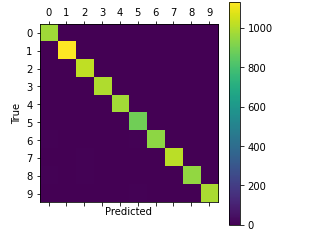
**MNIST Dataset From TensorFlow.Examples.Tutorials.Mnist:**

Example Error Image After 10000 Times Of Iterations(from CnnScript.py):

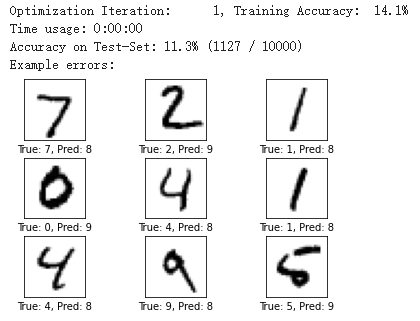


The confusion Matrix:





As the training process, starting from the first iteration to the 10000 iterations, the accuracy rate on Test-set changes from 11.3% to 98.8%, which improves the accuracy significantly. On the other hand, by looking at the example errors, we can actually tell that the model is learning and making good progress:



This is the graph of the first iteration. Here we can see the prediction number and the true number are quite off by their look. Compare to the graph example error above, we can clearly see that the model have learn some feature of the handwriting and can produce close prediction.

Now, compare to other model, CNN have a relatively high accuracy and shorter training time. Compare to other models, some reason why CNN have a high accuracy may due to:

1. The use of convolutional layers greatly reduces the number of parameters in the fully connected layers which makes the learning progress easier.
2. Using more powerful normalization technique like convolution to reduce overfitting.
3. It uses GPU for calculation.
4. It use modified linear units instead of S type neuron, which this will speed up the training based on experience.
5. Use a good initial weight to avoid slow learning process caused by neuron saturation.

And as all these above, they are the reason why cnnScript have a higher accuracy and shorter learning time than other models.