DL ASSIGNMENT 2

Harsha Vattem | 15CS10015

TASK A

Two structurally different neural networks have been trained and tested on a 70:30 split of the given data.

For training, a batch size of 64 is used, and done over 10 epochs. I observed that the validation accuracy did not change much after 10 epochs, so I stopped there. A common learning rate of 10^{-3} is used.

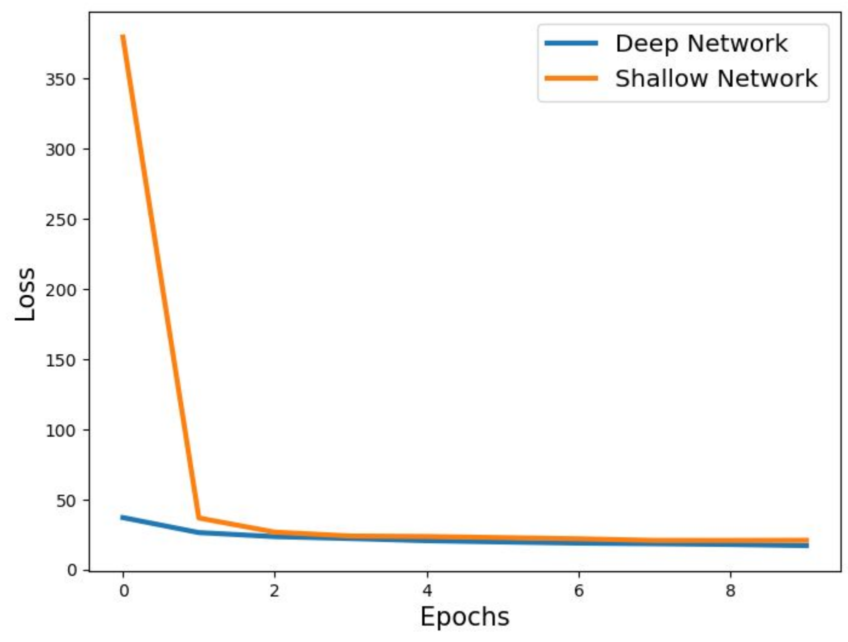
The accuracies for both networks on the test data, as well as the comparative training loss and validation over epochs for the two networks, are shown below.

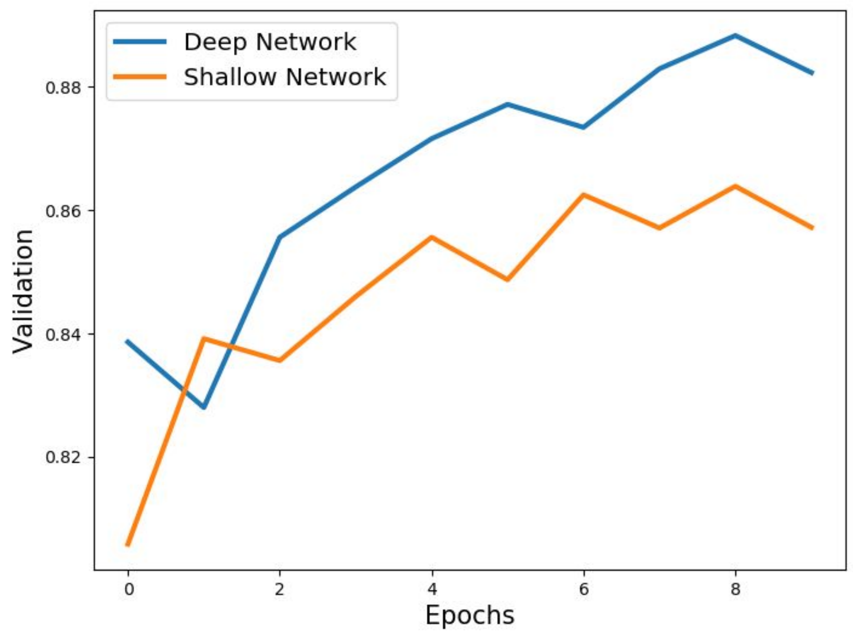
Network 1 (Deep and Narrow):

Accuracy(Deep Network) = 87.66%

Network 2 (Shallow and Wide):

Accuracy(Shallow Network) = 86.58%


Deep Network
Shallow Network
Loss
Epochs
350
300
250
200
150
100
50
0
0
2
4
6
8


Deep Network
Shallow Network
Validation
Epochs
0.88
0.86
0.84
0.82
0
2
4
6
8

TASK **B**

We experiment on the shallow network used earlier, by individually applying various methods of initialization, normalization, regularization (by dropout), and optimizers. For each type of variation, the results are compared with the vanilla network, which uses no explicit normalization, regularization or optimization.

The results and observations are given below, along with graphs comparing each of the variations to the vanilla network.

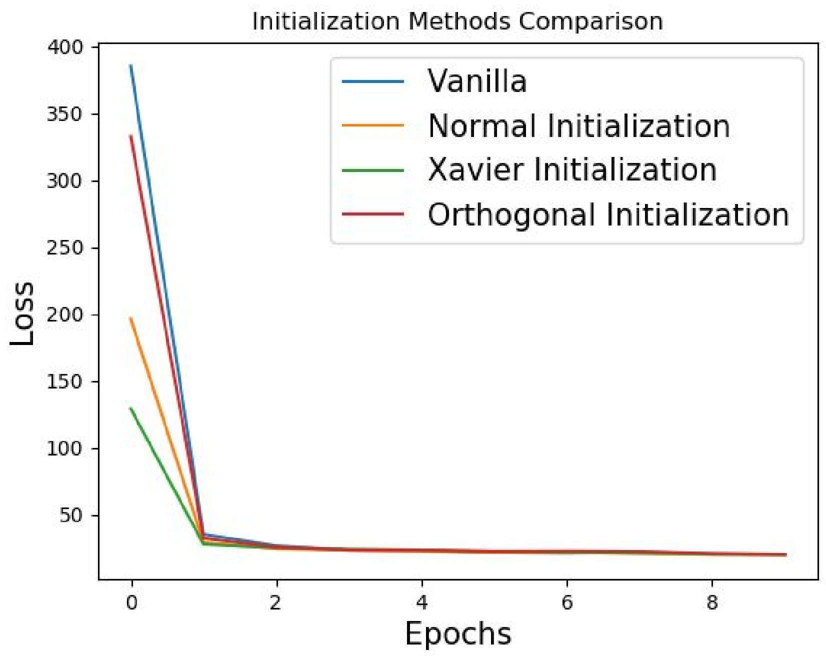
Initialization Techniques:

Accuracy(Vanilla) = 85.1%

Accuracy(Normal Initialization) = 85.27%

Accuracy(Xavier Initialization) = 86.0%

Accuracy(Orthogonal Initialization) = 85.57%

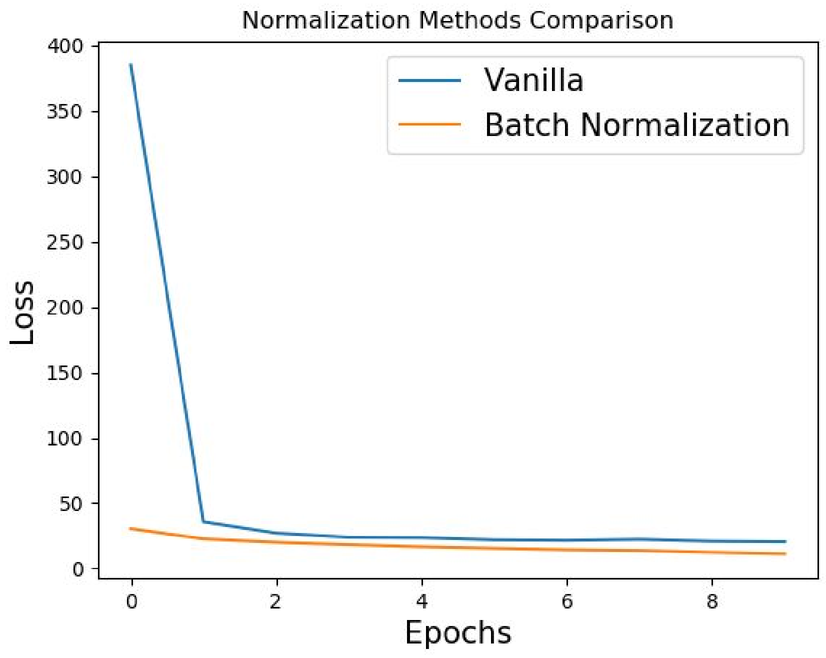


It is observed that all the initializers give a marginal improvement in performance, at not much extra cost. The learning is accelerated due to the improved distribution of initializations.

Normalization Techniques:

Accuracy(Vanilla) = 85.1%

Accuracy(Batch Normalization) = 88.12%



It is observed that batch normalization leads to a faster convergence, accelerating the learning process because after normalization, the outliers no longer create a distracting effect during training.

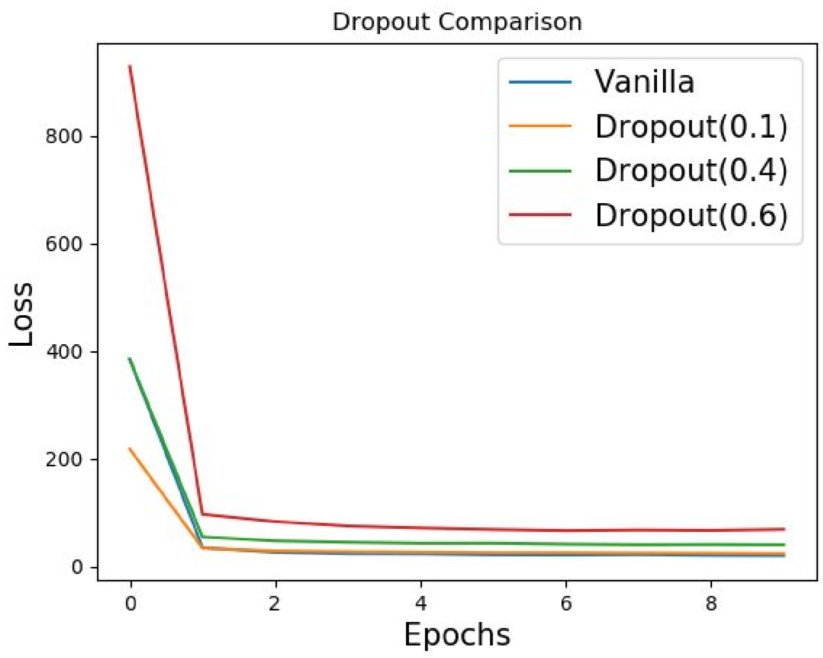
Regularization Techniques:

Accuracy(Vanilla) = 85.1%

Accuracy(Dropout(0.1)) = 83.61%

Accuracy(Dropout(0.4)) = 83.09%

Accuracy(Dropout(0.6)) = 61.02%



It is observed that for the dropout probabilities we used here, the network did not learn as well as the vanilla network. While dropout is generally used to prevent overfitting, the overfitting issue has not seemed to arise here. It will probably become a concern if a more complex network is trained for more epochs, which would give scope for overfitting (which could then be solved by dropout).

Optimization Techniques:

Accuracy(Vanilla) = 85.1%

Accuracy(SGD) = 83.92%

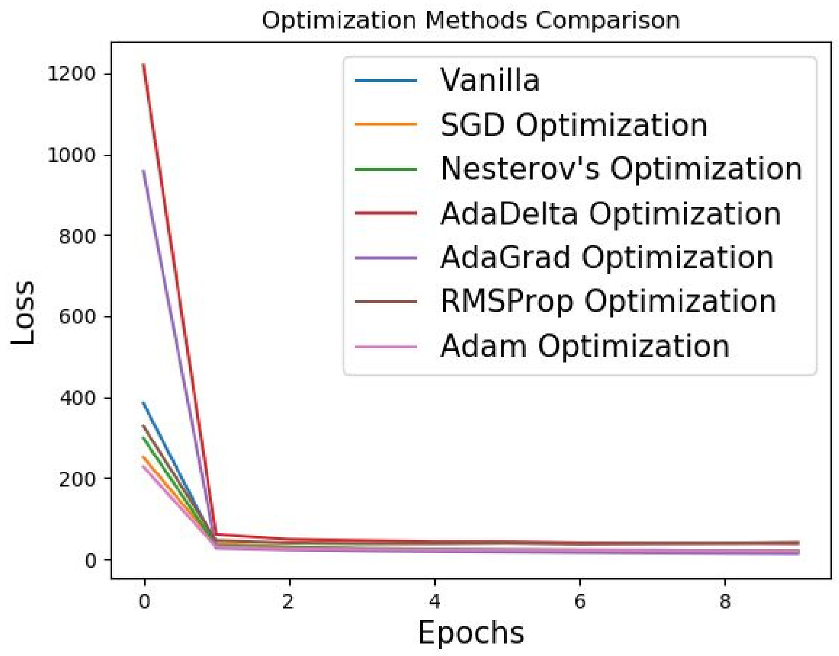
Accuracy(Nesterov's Accelerated Momentum) = 84.53%

Accuracy(AdaDelta) = 74.73%

Accuracy(AdaGrad) = 85.15%

Accuracy(RMSProp) = 78.25%

Accuracy(Adam) = 86.14%



It is observed that of the 6 optimizers that were used, each had its own speed and accuracy in producing results. While SGD is one of the most common, some other optimizers performed either better or worse, and I think the performance of each of the optimizers that were experimented with depends strongly on the type of data being used. For this specific case, Adam and AdaGrad performed well, while AdaDelta and RMSProp were considerably behind.

TASK **C**

Using the trained parameters of Network 2, a logistic regression classifier was trained on the outputs of each of the three layers.

The results of testing the logistic regression classifier are given below:

Accuracy (Layer 0): 87.7%

Accuracy (Layer 1): 87.73%

Accuracy (Layer 2): 87.27%

Although I expected the accuracies to be in increasing order, because the information increases in progressive layers, the test accuracy did not reflect this. The outputs claim that there is not a significant difference in the prediction accuracy from each layer.

CONCLUSION

All the accuracy values that were obtained above are specific to this dataset. I cannot conclude for sure that any one optimizer is better than the others, and similarly for regularization, initialization, etc. It depends a lot on the type of dataset and the type of values that are involved. Each of the methods does definitely have its merits, and a better understanding of each method will help us understand which to use for a given application.