ELL409 Assignment 2 Report
Neural Networks

Aditi Khandelwal (2018EE10434) , Kamna Meena (2018EE10470)

May 20, 2021

In this assignment, we have implemented N-layered neural network with different activation functions and explored different optimization algorithms and regularization techniques. We also observed the effect of number of layers as well as number of neurons in the hidden layers in the learning of the neural network model.

Contents

1 Assignment Problem

In this section, we discuss about the neural network as given in the problem statement. Following are the results of a neural network with 3 layers (input layer, hidden layer, output layer). The input layer has 784 neurons. The hidden layer has 'sigmoid' activation function with 128 neurons and the output layer has 'softmax' activation function with 10 neurons. The parameters have been initialized using Xavier Initialization.

Learning Rate = 0.1
Accuracy on Train Set = 90.525%
Accuracy on Test Set = 87.9%

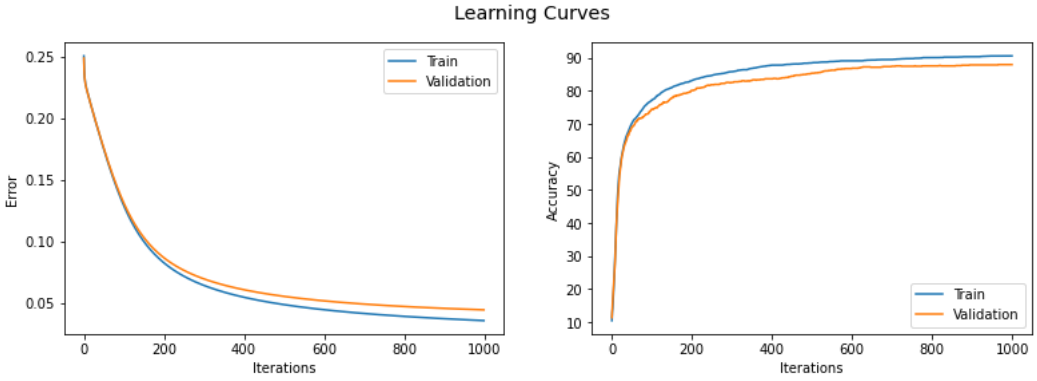
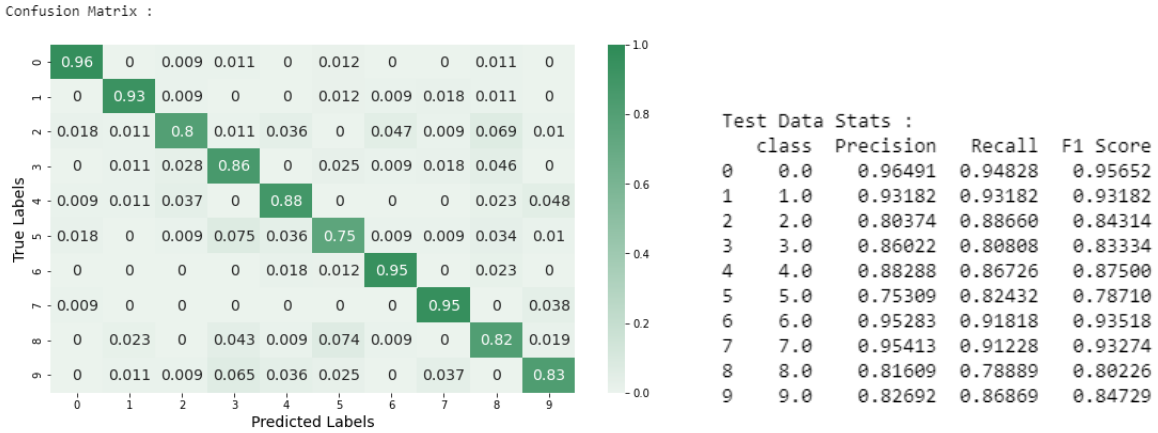


Figure 1: Learning Rate = 0.1



Test Data Stats :

	class	Precision	Recall	F1 Score
0	0.0	0.96491	0.94828	0.95652
1	1.0	0.93182	0.93182	0.93182
2	2.0	0.80374	0.88660	0.84314
3	3.0	0.86022	0.80808	0.83334
4	4.0	0.88288	0.86726	0.87500
5	5.0	0.75309	0.82432	0.78710
6	6.0	0.95283	0.91818	0.93518
7	7.0	0.95413	0.91228	0.93274
8	8.0	0.81609	0.78889	0.80226
9	9.0	0.82692	0.86869	0.84729

Figure 2: (a) Confusion Matrix, (b) Test Data Stats

1.1 Results with Regularization

The following regularization techniques were employed to see the difference keeping the learning rate and number of iterations same.

1.1.1 L1 Regularization

Lambda = 0.0003
Accuracy on Train Set = 90.025%
Accuracy on Test Set = 87.6%

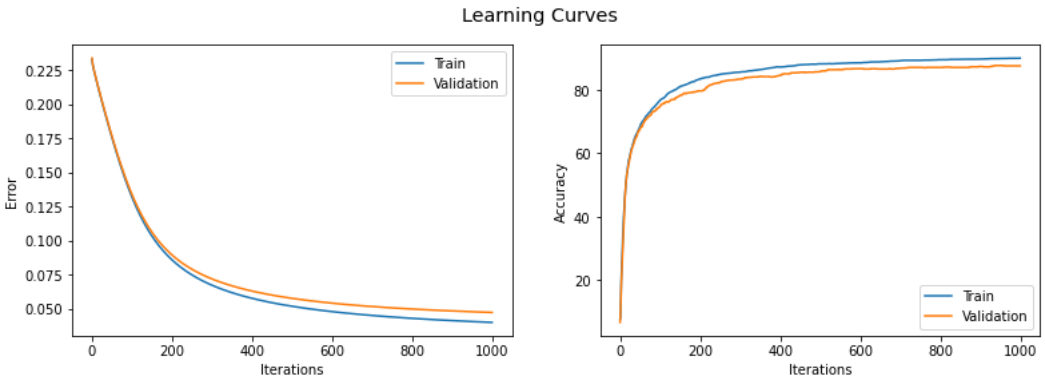
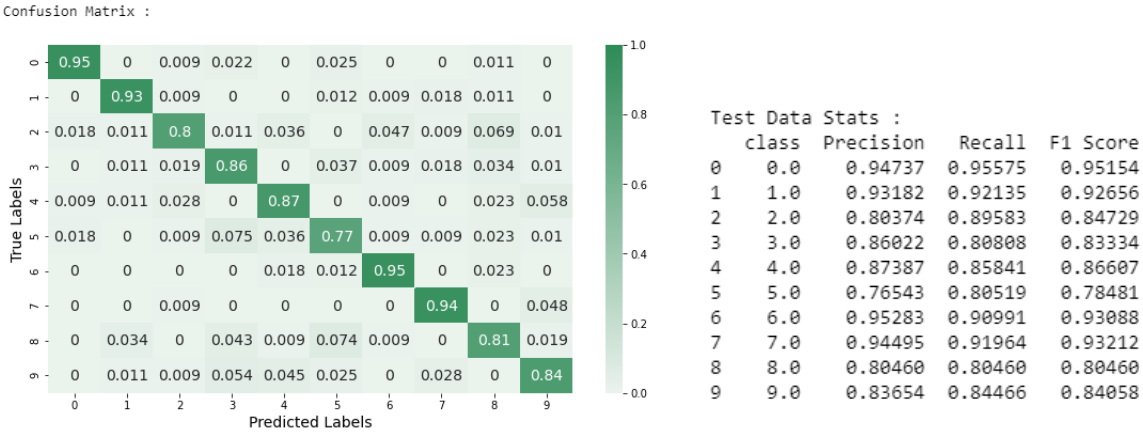


Figure 3: Learning Rate = 0.1



Test Data Stats :

	class	Precision	Recall	F1 Score
0	0.0	0.94737	0.95575	0.95154
1	1.0	0.93182	0.92135	0.92656
2	2.0	0.80374	0.89583	0.84729
3	3.0	0.86022	0.80808	0.83334
4	4.0	0.87387	0.85841	0.86607
5	5.0	0.76543	0.80519	0.78481
6	6.0	0.95283	0.90991	0.93088
7	7.0	0.94495	0.91964	0.93212
8	8.0	0.80460	0.80460	0.80460
9	9.0	0.83654	0.84466	0.84058

Figure 4: (a) Confusion Matrix, (b) Test Data Stats

1.1.2 L2 Regularization

Lambda = 0.01
Accuracy on Train Set = 89.075%
Accuracy on Test Set = 87.0%

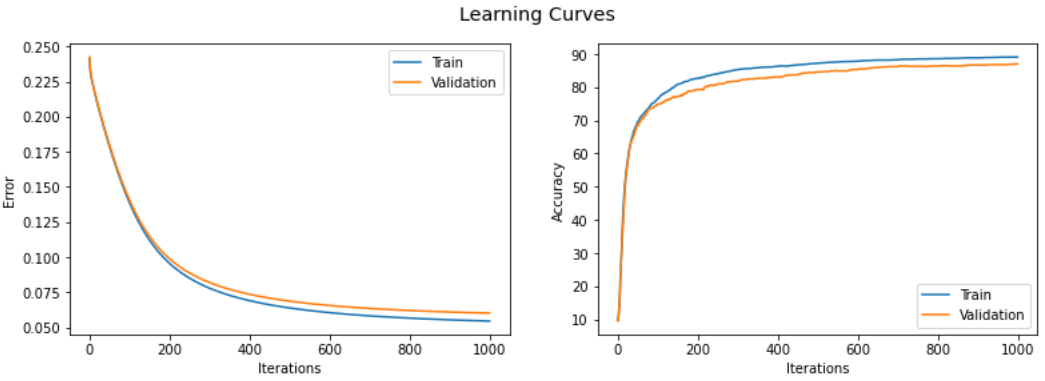


Figure 5: Learning Rate = 0.1

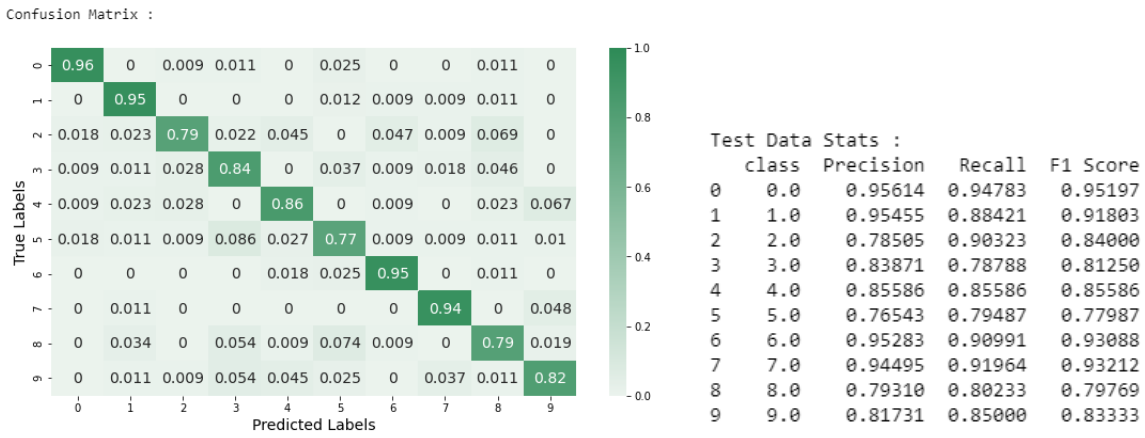


Figure 6: (a) Confusion Matrix, (b) Test Data Stats

1.1.3 Elastic Net Regularization

Lambda1 (L1) = 0.0003
Lambda (L2) = 0.01
Accuracy on Train Set = 88.075%
Accuracy on Test Set = 85.0%

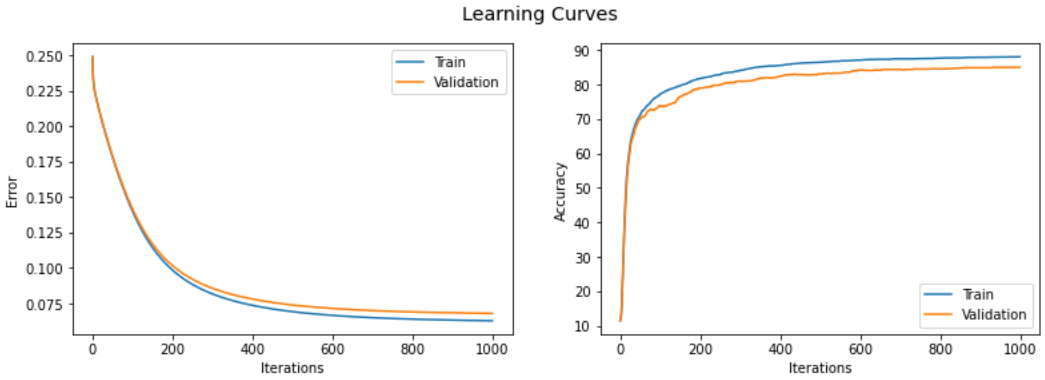


Figure 7: Learning Rate = 0.1

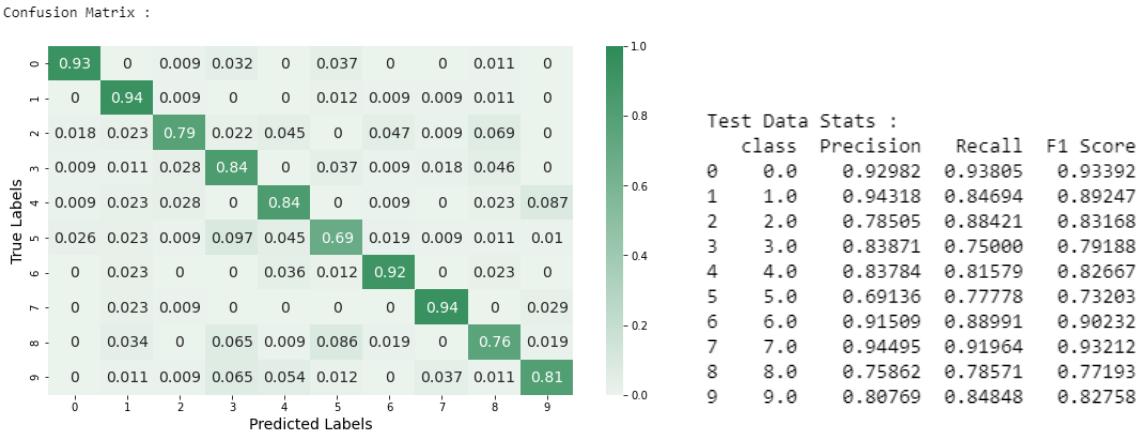


Figure 8: (a) Confusion Matrix, (b) Test Data Stats

1.1.4 Tikhonov Regularization

Accuracy on Train Set = 90.45%
Accuracy on Test Set = 87.8%

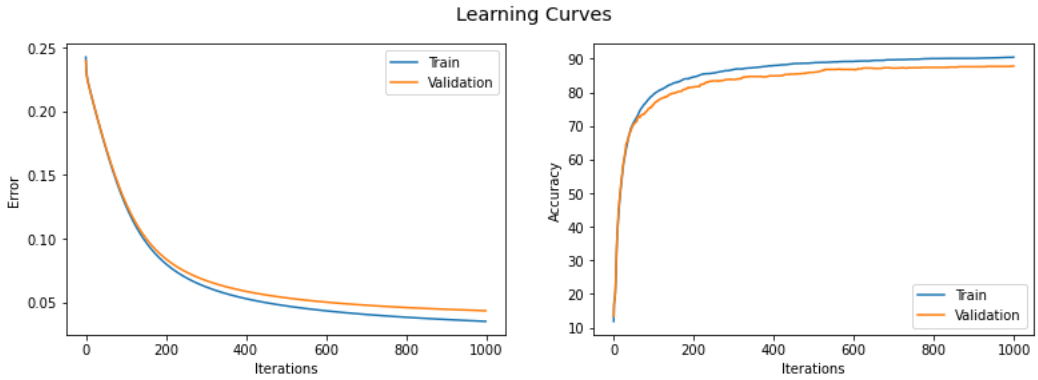


Figure 9: Learning Rate = 0.1

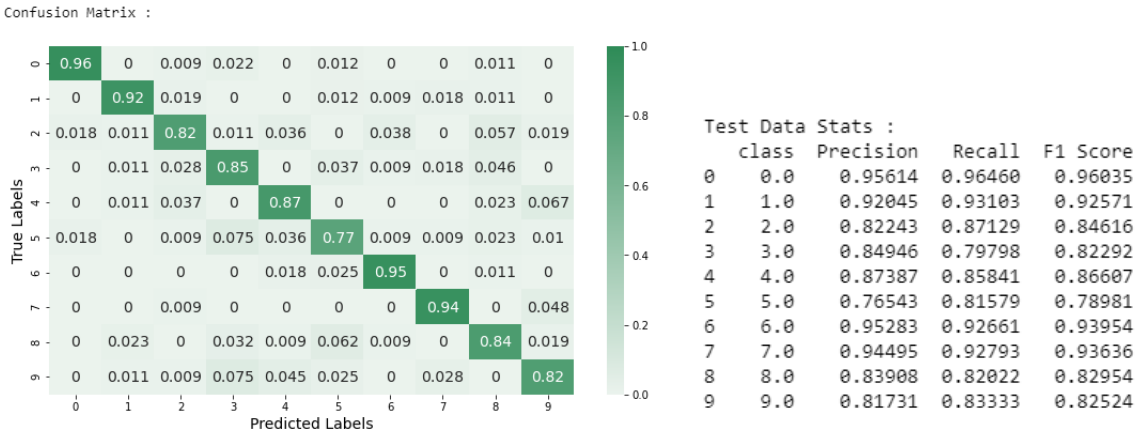


Figure 10: (a) Confusion Matrix, (b) Test Data Stats

Inference

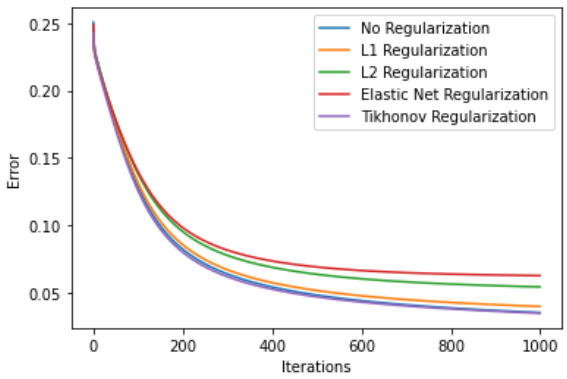


Figure 11: Effect of regularizations in NN Training

It was empirically observed that no regularizer improved the accuracy of the model significantly since there is no sign of overfitting for sigmoid activation function (when the test starts to rise up). Thus we can say that the dataset and the classification problem is not challenging in the sense that the train and test set are not very different and even if the model memorizes the train set, it will still do good on test set as can be seen in the Appendix.

2 Exploration

In this section, we explore the impact of changing the size of neural network, different activation functions and optimization algorithms.

2.1 Effect of using different Activation Functions

In this section, we assess the effect of using different non-linear activation functions on the hidden layers, namely, Sigmoid, Tanh, ReLU and leaky-ReLU. The output layer is activated by softmax activation for all the models.

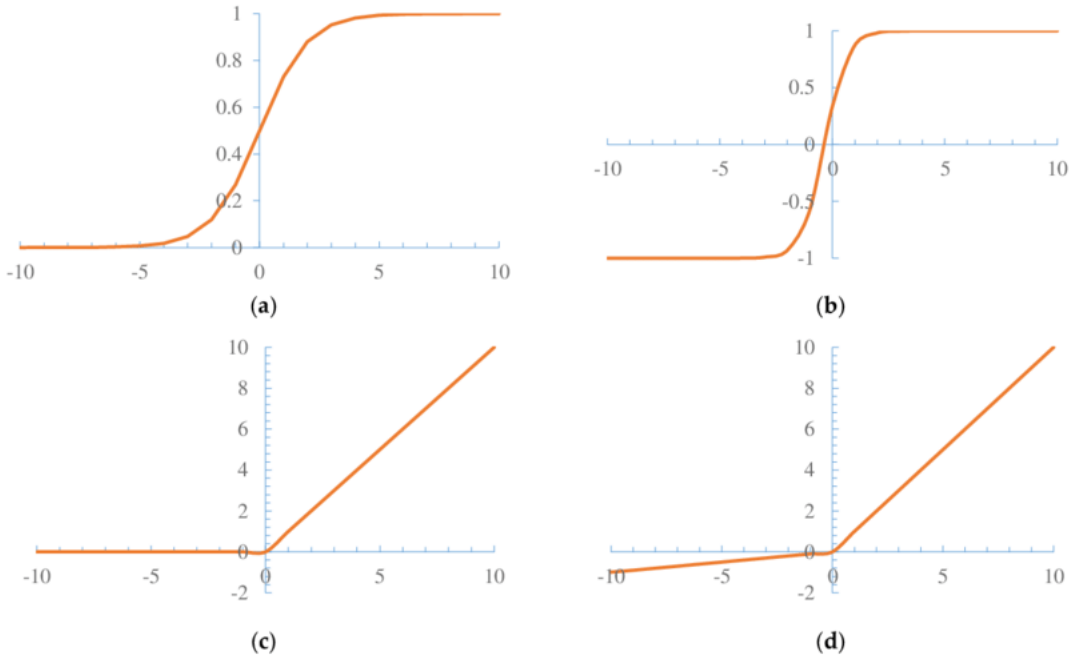


Figure 12: (a) Sigmoid, (b) Tanh, (c) ReLU, (d) Leaky ReLU Activations[1]

2.1.1 Sigmoid or Logistic Activation Function:

Accuracy on Train Set = 90.1%
Accuracy on Test Set = 89.1%
Learning Rate = 0.1

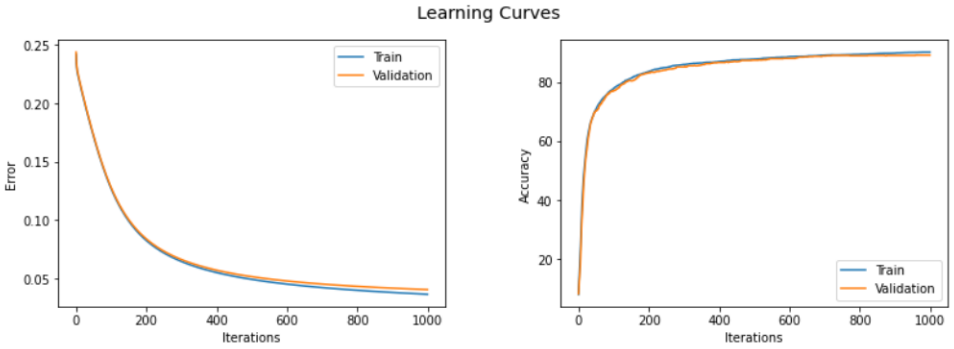


Figure 13: Learning Rate = 0.1

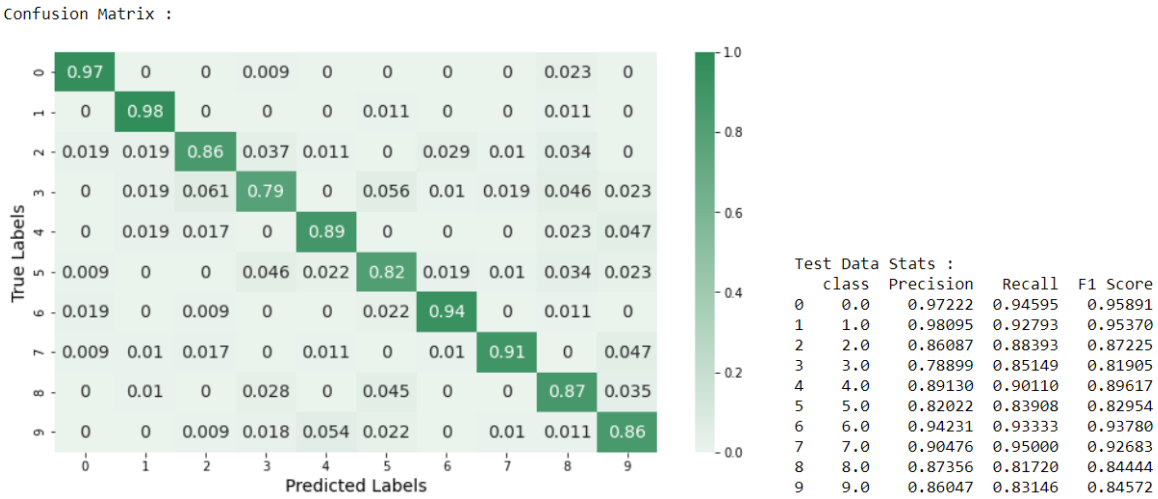


Figure 14: (a) Confusion Matrix, (b) Test Data Stats

2.1.2 Tanh or hyperbolic tangent Activation Function:

Accuracy on Train Set = 93.4%
Accuracy on Test Set = 89.5%
Learning Rate = 0.1

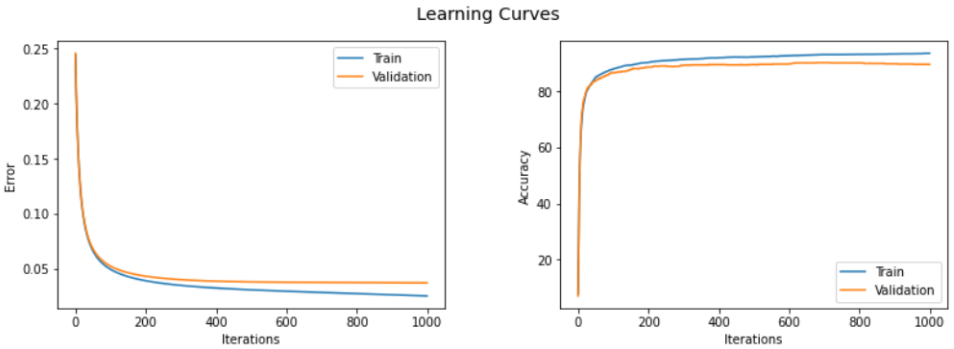


Figure 15: Learning Rate = 0.1

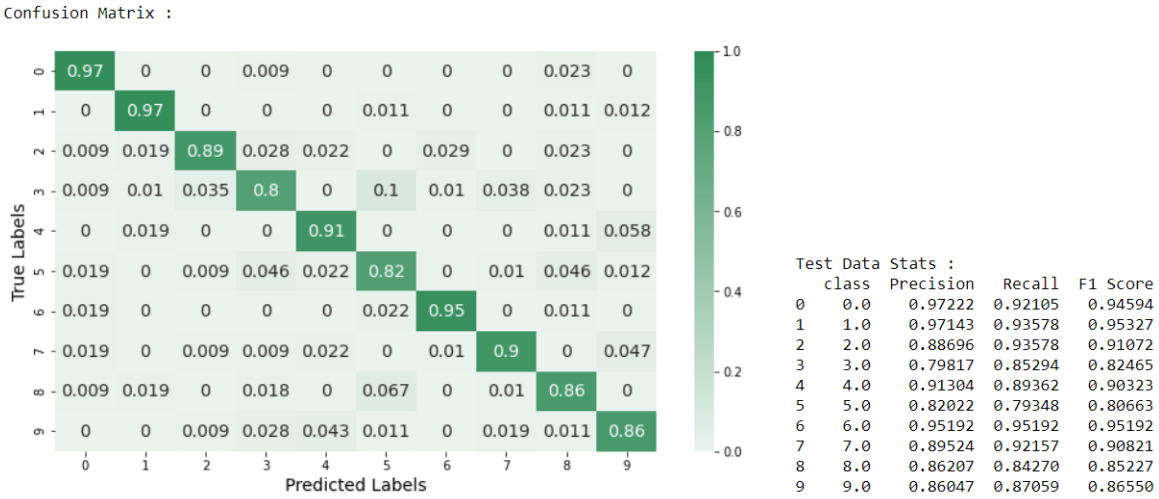


Figure 16: (a) Confusion Matrix, (b) Test Data Stats

2.1.3 ReLU (Rectified Linear Unit) Activation Function:

Accuracy on Train Set = 97.05%
Accuracy on Test Set = 91.3%
Learning Rate = 0.1

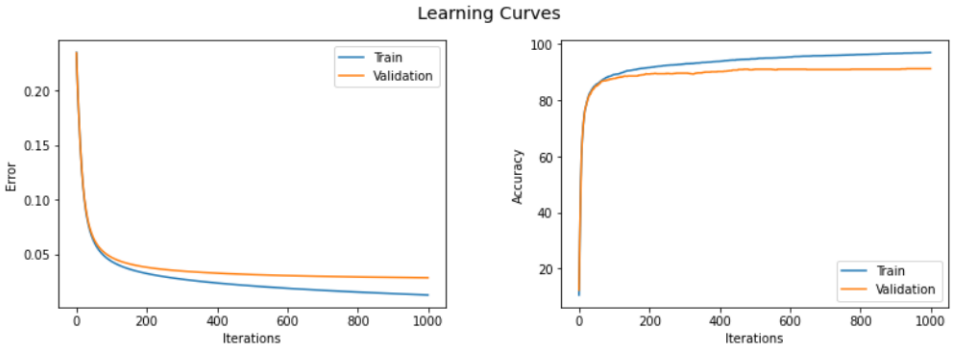
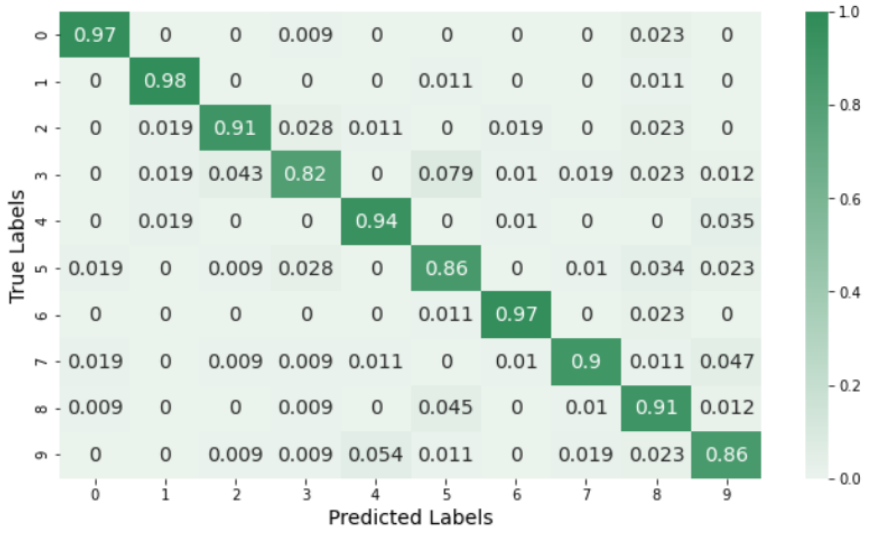


Figure 17: Learning Rate = 0.1

Confusion Matrix :



Test Data Stats :

	class	Precision	Recall	F1 Score
0	0.0	0.97222	0.95455	0.96330
1	1.0	0.98095	0.94495	0.96261
2	2.0	0.91304	0.92920	0.92105
3	3.0	0.81651	0.89899	0.85577
4	4.0	0.93478	0.92473	0.92973
5	5.0	0.86517	0.84615	0.85555
6	6.0	0.97115	0.95283	0.96190
7	7.0	0.89524	0.94000	0.91707
8	8.0	0.90805	0.84043	0.87293
9	9.0	0.86047	0.87059	0.86550

Figure 18: (a) Confusion Matrix, (b) Test Data Stats

2.1.4 Leaky ReLU:

Accuracy on Train Set = 96.925 %
Accuracy on Test Set = 91.4%
Learning Rate = 0.1

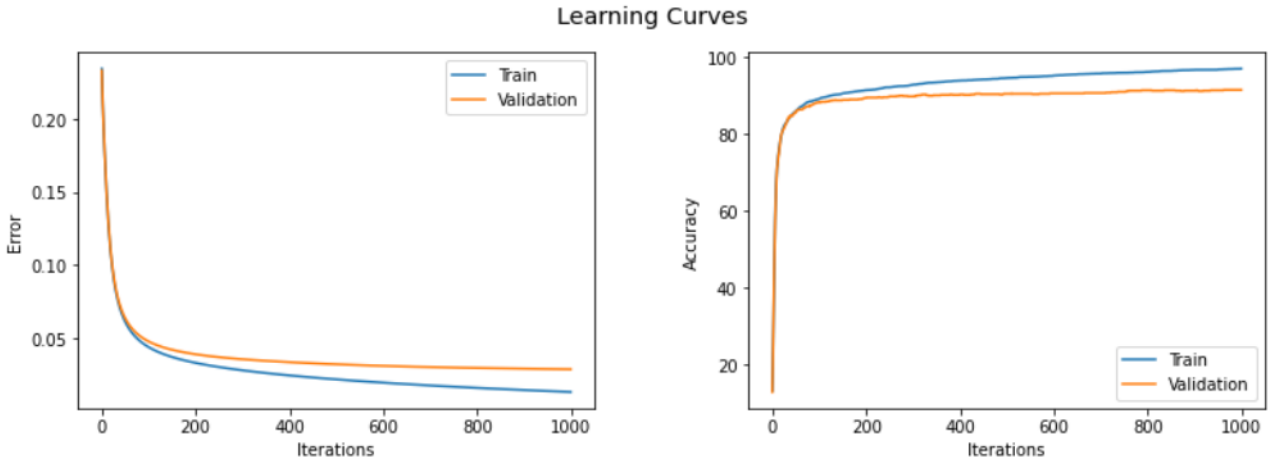


Figure 19: Learning Rate = 0.1

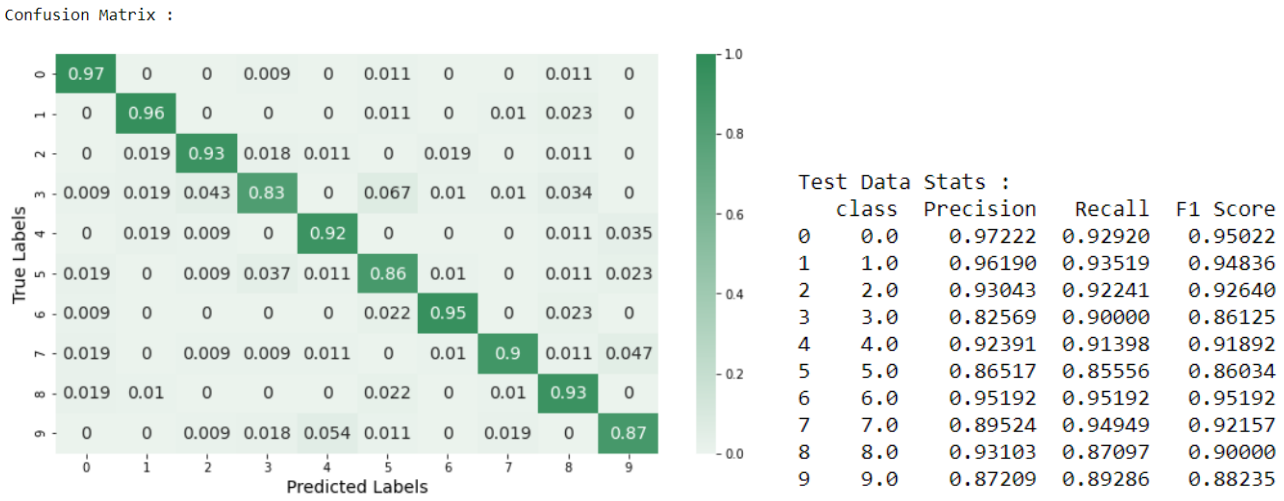


Figure 20: (a) Confusion Matrix, (b) Test Data Stats

Inference

Upon comparing the learning curves of models with different activation functions, we can see that the model with ReLU activation function has learned the fastest as can be witnessed from the plots that the loss and accuracy plateaus the fastest for this case. The speed of the learning is in the order as: ReLU > Leaky ReLU > Tanh > Sigmoid. See Appendix for the accuracy metrics.

2.2 Effect of Changing Width and Depth of Neural Network

The impact of changing the size of the neural network is assessed on NN with ReLU activation function and gradient descent optimization algorithm with no regularization.

2.2.1 Results with different number of layers in the Neural Network:

Learning Rate = 0.1

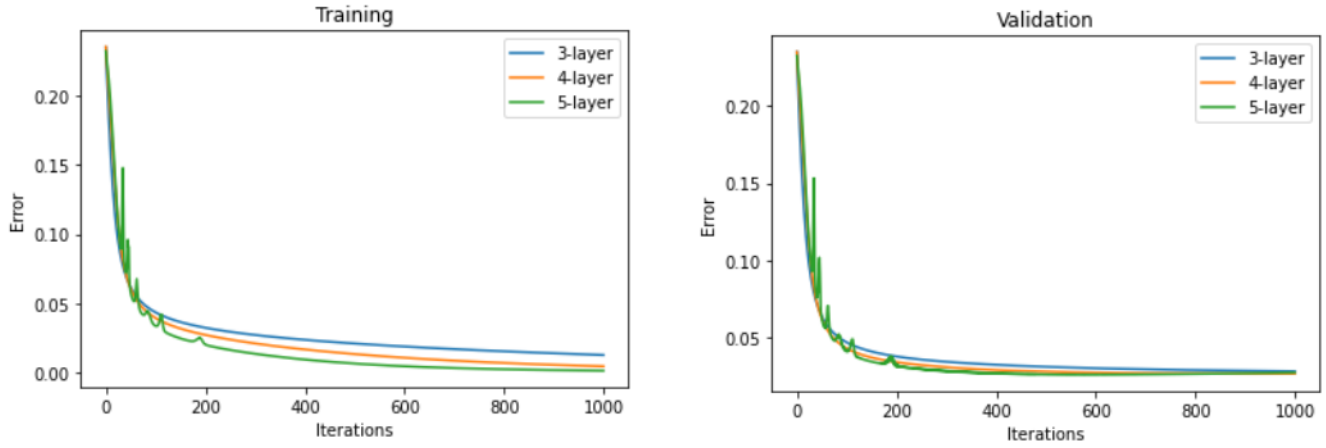


Figure 21: Convergence of different layered NN on Training and Validation Set

Number of Hidden Layers	Training Accuracy	Testing Accuracy
3-layer [128]	97.05%	91.3%
4-layer [128,64]	99.525%	92.1%
5-layer [256,128,64]	99.95%	92.4%

2.2.2 Results with different number of neurons in hidden layer of the 3-layer Neural Network:

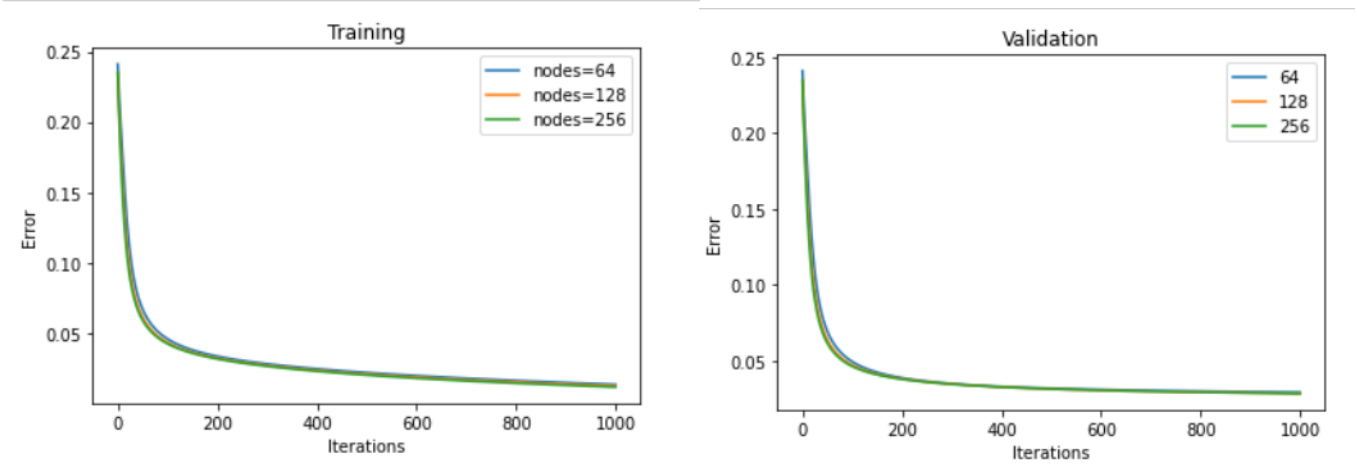


Figure 22: Convergence of 3-layer NN with different number of nodes in the hidden layer on Training and Validation Set

Neurons	Training Accuracy	Testing Accuracy
64	96.575%	91.4%
128	97.05%	91.3%
256	97.525%	91.2%

Inference

We can make inference that increasing the width of the network does not increase the learning capacity of the model significantly but this is not the case upon change the depth. By increasing the depth of the network, there is significant improvement in the performance. We can see that there are few peaks for the model with 5 layers. This is most likely due to exploding gradients when the depth is high.

2.3 Exploring Different Optimization Algorithms

In this section, we explore different optimization algorithms, namely, gradient descent, Momentum, RMS Prop, Adam[2]. We use 'Tanh' activation function with one hidden layer of 128 neurons and and keeping learning rate constant for all algorithms equal to 0.01.

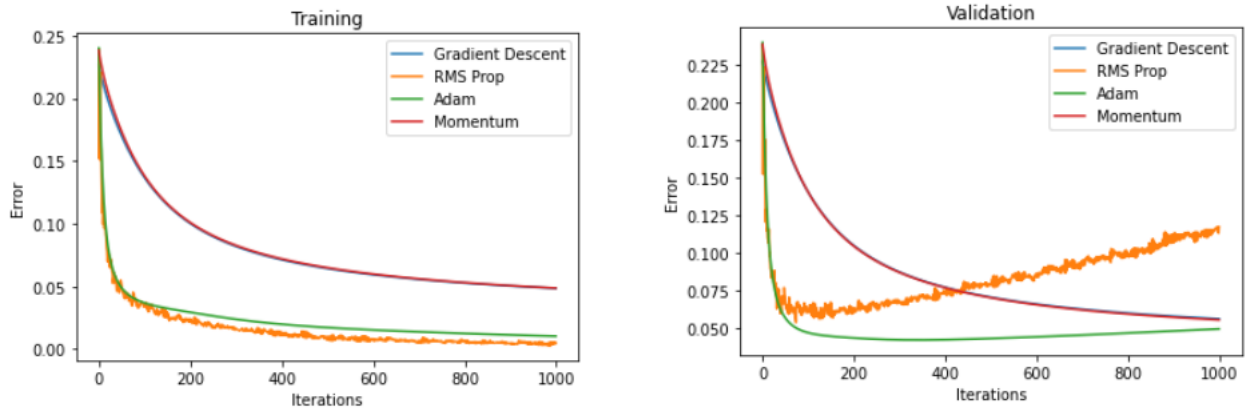


Figure 23: Convergence of optimizers on Training and Validation Set

Optimizer	Training Accuracy	Testing Accuracy
Gradient Descent	88.35%	85.8%
Momentum	88.1%	85.9%
RMS Prop	98.2%	86.2%
Adam	97.75%	88.5%

Inference:

1. We know that gradient descent can be highly sensitive to some directions and insensitive to others in the parameter space, the momentum tries to fix this issue. We can see that both the momentum and gradient descent plots overlap. This might be indicating that our loss function is not skewed i.e. our loss function is like bowl shaped i.e., convex.

2. We can see that RMSProp and Adam are converging fast. This stems from the fact that both methods use adaptive learning rate.

3. Since Adam is a combination of RMSProp and Momentum, we can see that we got a boost in the validation accuracy. The boost is explainable because Adam also fixes the bias correction in the RMSProp. See Appendix for the accuracy metrics.

3 Conclusion

In this assignment, we implemented a fully connected neural network from scratch. From our experiments, we can say that these models have high learning capacity, but require little bit of hyperparameter tuning. We have reported experimental results obtained by using different optimizers. We also investigated the effect of changing the depth and width of the neural network i.e. how and when the model learns better. It was found out that increasing the depth improved the performance of the model, but there was not much difference upon changing width. We can further explore the effect of different regularizers, activations and optimizers using the code implemented. There are many possible combinations that can be tried but we have limited ourselves to most reasonable combinations.

4 Appendix

The following results were obtained by tuning and finding the best hyperparameters of the combinations. These hyperparameters can be further tuned using random search or grid search [3] to improve performance.

Algorithm	Train Accuracy	Test Accuracy
ReLU Activation		
Gradient Descent	97.25%	90.9%
Gradient Descent with L1 Regularization	92.575%	89.1%
Gradient Descent with L2 Regularization	94.65%	90.8%
Gradient Descent with Elastic Net Regularization	90.8%	87.9%
Gradient Descent with Tikhonov Regularization	96.525%	91.0%
Momentum	97.25%	91.0%
RMS Prop	99.95%	91.7%
Adam	100.0%	92.3%
Tanh Activation		
Gradient Descent	93.2%	88.9%
Gradient Descent with L1 Regularization	91.1%	88.6%
Gradient Descent with L2 Regularization	93.25%	88.6%
Gradient Descent with Elastic Net Regularization	89.5%	86.9%
Gradient Descent with Tikhonov Regularization	93.15%	88.7%
Momentum	93.2%	88.5%
RMS Prop	97.4%	87.6%
Adam	98.125%	89.6%
Sigmoid Activation		
Gradient Descent	90.525%	87.9%
Gradient Descent with L1 Regularization	90.025%	87.6%
Gradient Descent with L2 Regularization	89.075%	87.0%
Gradient Descent with Elastic Net Regularization	88.075%	85.0%
Gradient Descent with Tikhonov Regularization	90.45%	87.8%
Momentum	91.67%	88.7%
RMS Prop	95.975%	86.4%
Adam	95.5%	87.3%
Leaky Relu		
Gradient Descent	97.425%	91.3%
Gradient Descent with L1 Regularization	97.3%	90.5%
Gradient Descent with L2 Regularization	97.275%	90.8%
Gradient Descent with Elastic Net Regularization	92.0%	89.3%
Gradient Descent with Tikhonov Regularization	96.225%	90.4%
Momentum	97.05%	90.6%
RMS Prop	100.0%	92.9%
Adam	100.0%	93.2%

References

[1] Jing, Yang & Guanci, Yang. (2018). Modified Convolutional Neural Network Based on Dropout and the Stochastic Gradient Descent Optimizer. Algorithms. 11. 28. 10.3390/a11030028.

[2] https://medium.com/analytics-vidhya/a-complete-guide-to-adam-and-rmsprop-optimizer-75f4502d83be

[3] https://machinelearningmastery.com/hyperparameter-optimization-with-random-search-and-grid-search/