CS 559 FINAL REPORT

Agamdeep S. Chopra

TASK 1)

LOGISTIC REGRESSION:

I utilized sklearn provided Linear Regression model with One vs Rest strategy in a custom class to fit and predict the MNIST dataset. The reason behind choosing Logistic Regression was to establish a baseline for the neural network models. Since logistic regression is a specific subset of NNs, the idea was to have decently reliable statistics to evaluate the performance of the other complex models. Generally, the training of this method is quite fast and efficient. But, due to the inherent linear dependance assumption/linear decision surface, the model performs very poorly on the dataset, which explains why the model failed to converge.

SVM:

My SVM algorithm utilized rbf kernel and One vs One strategy to classifying the data into 10 categories. This technique splits classification into binary classification per unique label pairs. SVM is a strong classification algorithm and often used as a reliable classification algorithm in many cases. This was one of the reasons why I was interested to test the algorithm on the MNIST dataset. SVM performed very good with clear label separation. But, due to the nature of the algorithm, training time was very high.

SINGLE NODE NEURAL NETWORK:

My algorithm comprises of a single node with softmax activation to predict probabilities of the 10 labels. The algorithm was trained using ADAM optimization and Cross-Entropy Loss. Neural networks work extremely well in generalizing complex non-linear datasets. My reasoning to use this algorithm was to see how well a perceptron may perform on the dataset and to set a reference baseline for the complex NN algorithms. I was able to train this model on a GPU with batches, which made training much faster and efficient. Although, optimizing the hyperparameters such as Learning Rate, epochs, etc. was quite tricky and time consuming.

Logistic Regression	SVM	1 Node NN
Fast training	**Extremely slow training**	**Slightly slow training**
Never converged	**Converges**	**Converges**
Poor accuracy	**Very high accuracy**	**High accuracy**
Linearity assumption	**rbf Kernel**	**Non-Linear Softmax**
One-vs-rest	**One-vs-one**	**-**
Cross-entropy	**Cross-entropy**	**Cross-entropy**

Table 1.1. *Comparison Table for the 3 models used in Task 1.*

TASK 2)

For Task 2, I developed 2 separate deep neural networks. The first was a 10-layer dense Neural Network with 500 nodes per hidden layer and ReLU activation. The final layer had a SoftMax activation to output probability of the 10 class labels. This was trained with batch normalization, ADAM optimizer, cross entropy loss, L2 regularization, and dropout. The model performed extremely well on the datasets. Similarly, I developed a 10 layer Fully Convolutional Neural Network with ReLU activation and SoftMax for the last flattened layer. Again, this was trained with batch normalization, ADAM optimizer, cross entropy loss, L2 regularization, and dropout. The FCNN model performed the best out of all models. Please refer to the provided code for detailed implementation of all the models.

TASK 3)

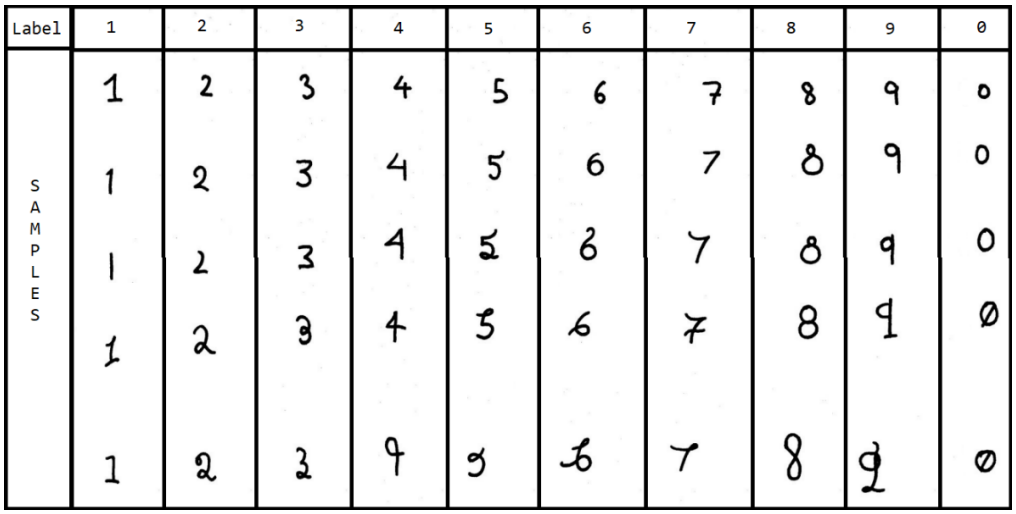


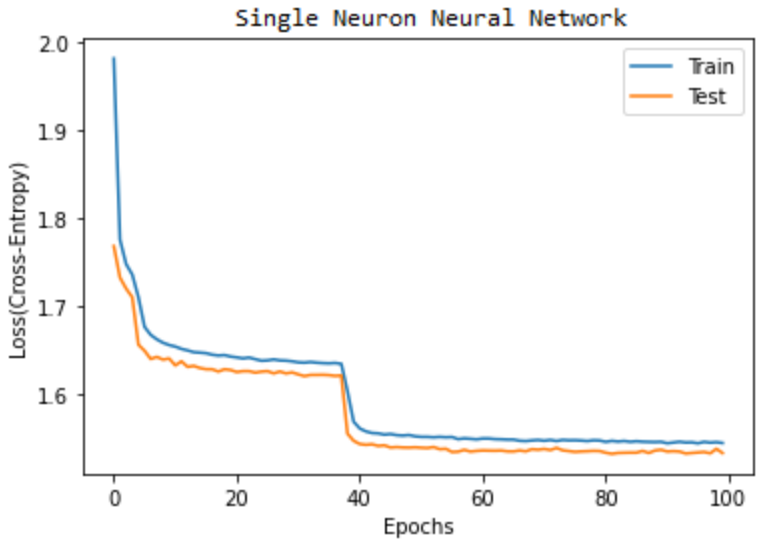
Fig 1. *Custom hand-drawn test dataset.*

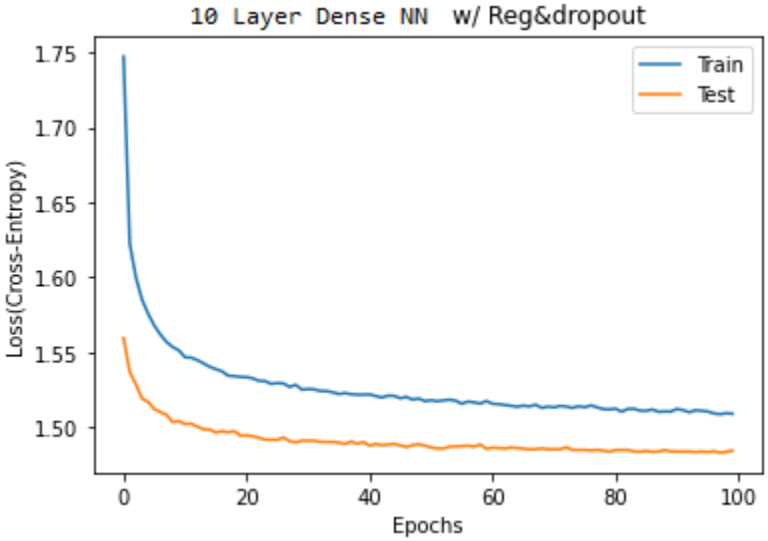
STATISTICS:

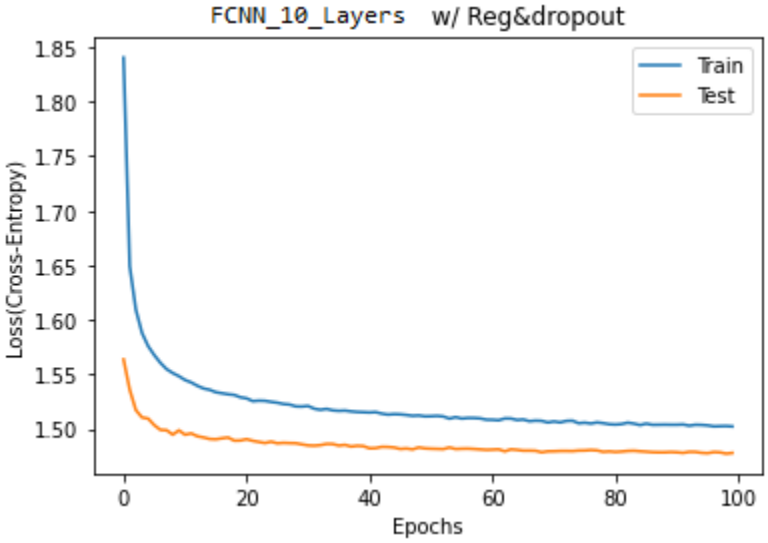
ACCURACY	Log-Reg	SVM	1 Node NN	10 Layer DNN	10 Layer FCNN*
Train	**93.39%**	**98.99%**	**92.85%**	**98.21%**	**98.55%**
Validation	**92.55%**	**97.92%**	**92.72%**	**97.68%**	**98.33%**
Test (my dataset)	**36.00%**	**60.00%**	**46.00%**	**66.00%**	**86.00%**

Table 2. *Accuracy comparison table. FCNN model has the best generalization with the best performance on validation and custom dataset.*

MISCELLANEOUS:


Single Neuron Neural Network
Train
Test
Loss(Cross-Entropy)
Epochs
2.0
1.9
1.8
1.7
1.6
0
20
40
60
80
100


10 Layer Dense NN w/ Reg&dropout
Train
Test
Loss(Cross-Entropy)
Epochs
1.75
1.70
1.65
1.60
1.55
1.50
0
20
40
60
80
100


FCNN_10_Layers w/ Reg&dropout
Train
Test
Loss(Cross-Entropy)
Epochs
1.85
1.80
1.75
1.70
1.65
1.60
1.55
1.50
0
20
40
60
80
100