EE6132: Assignment 4
RNNs

December 15, 2021

Contents

1 Brief Description of the Code

1.1 Libraries used

The libraries used to run the python code are described below.

- numpy
- matplotlib.pyplot
- os
- torch
- sklearn
- torchvision
- tqdm
- time

1.2 Functions defined

Different models were described for this assignment.
The models described are listed below.
All the models have flags to control the hidden layer size, bidirectional training, e.t.c,

- Vanilla RNN
- Vanilla LSTM
- Vanilla GRU
- Binary LSTM for the last part

The different functions described to run these models for the assignment are listed below.

- Cross Entropy : A custom built loss function as pytorch doesn't support non-one hot tensors for the true label.
- Train : Trains the models for the first part of the assignment.
- Test : Tests/Validates the models for the first part of the assignment.
- Random Pred : Checks the models on a random set of test images.
- Sequence Generator: Generates sequences to be fed to the RNN
- Train Sequence: Trains the models for the second part of the assignment.
- Test Sequence : Tests the models for the second part of the assignment.

- Binary Sequence Generator: Generates the binary sequences and their sums for the third part of the assignment.
- Train Sum:Trains the models for the third part of the assignment.
- Test Sum:Tests the models for the third part of the assignment.
- Check Sequence: Checks the models on a set of varying length sequences.
- Check Binary Sequence: Checks the models on a set of varying length binary sequences.
- Run RNN: a function which encompasses all the above functions and runs them appropriately based on a series of flags.
- Run Assignment: a wrapper function which runs the entire assignment based on a series of user inputs.

2 MNIST Digit Classification using RNNs

For this section of the assignment, the hyperparameters considered where,

- Learning Rate = 0.001
- Batch Size = 64
- Number of Epochs = 5

These hyperparameters were considered based on prior experience and reviewing literature.
The loss function chosen was the Negative Log Likelihood loss function (which coupled with log softmax behaves like Cross Entropy) as it is more efficient.
The optimizer chosen was the ADAM optimizer.
A sample architecture of the RNN used is given below.

```
    RNN(
  (rnn): RNN(28, hidden layer, batch_first=True)
  (HL): Linear(in_features=hidden layer, out_features=10, bias=True)
  (logsoftmax): LogSoftmax(dim=1)
)
```

2.1 Test Images

For conveniences sake, a set of random images were chosen beforehand on which the models were tested.
This lends some equality towards testing the different models.
The test images chosen for testing are shown below.

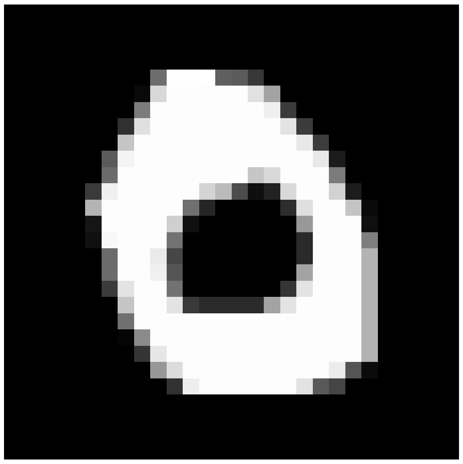


Figure 1: The digit 0 from the MNIST dataset

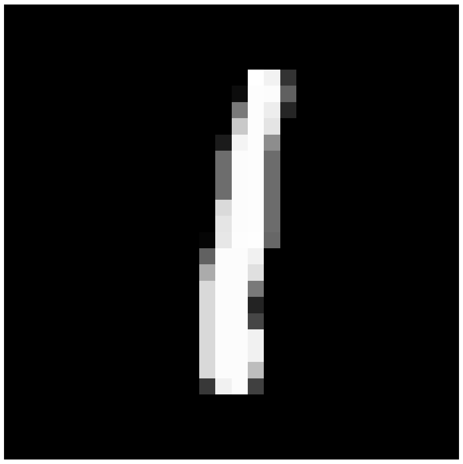


Figure 2: The digit 1 from the MNIST dataset

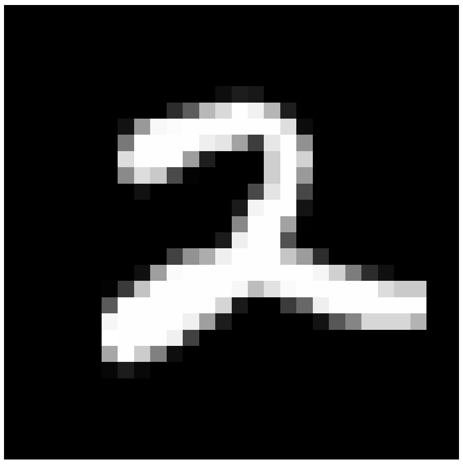


Figure 3: The digit 2 from the MNIST dataset

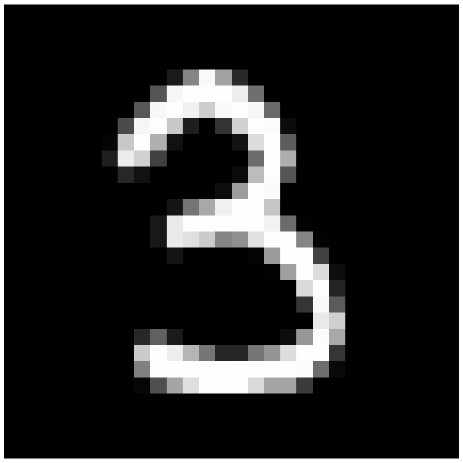


Figure 4: The digit 3 from the MNIST dataset

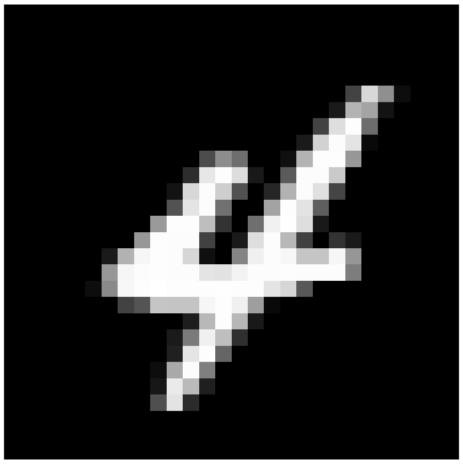


Figure 5: The digit 4 from the MNIST dataset

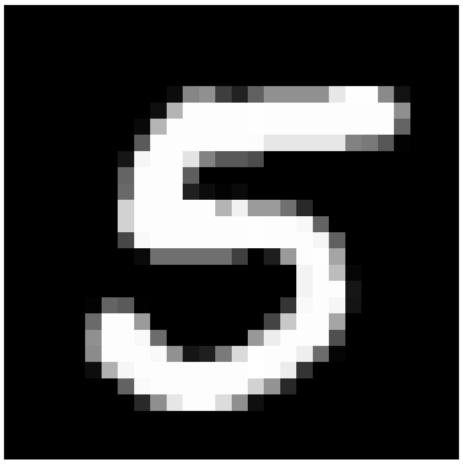


Figure 6: The digit 5 from the MNIST dataset

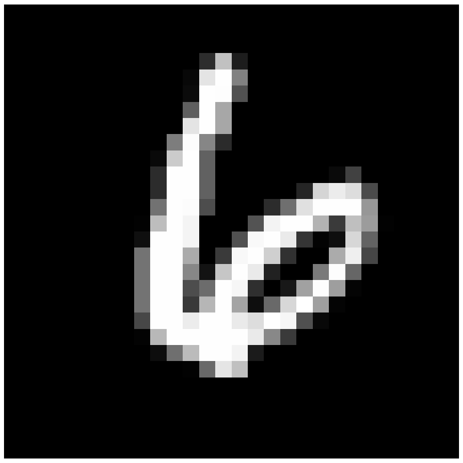


Figure 7: The digit 6 from the MNIST dataset

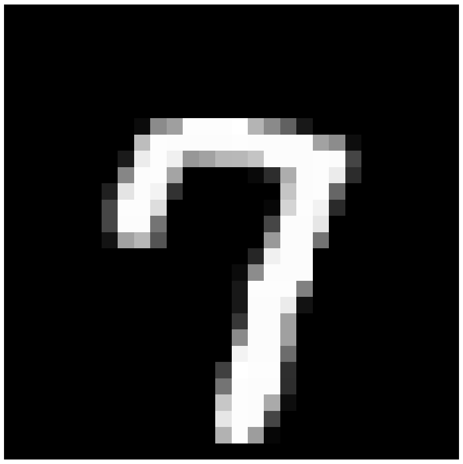


Figure 8: The digit 7 from the MNIST dataset

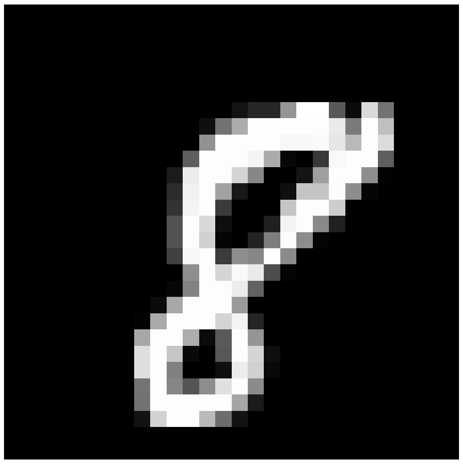


Figure 9: The digit 8 from the MNIST dataset

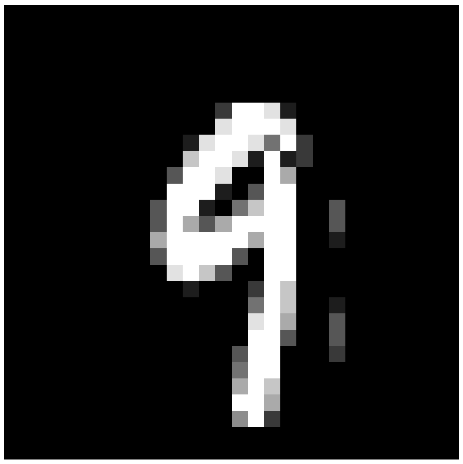


Figure 10: The digit 9 from the MNIST dataset

2.2 Vanilla RNN

For the Vanilla RNN a hidden layer size of 128 was chosen upon consulting various resources. This hidden layer size of 128 was set as the benchmark for all models.

2.2.1 Training Plots:

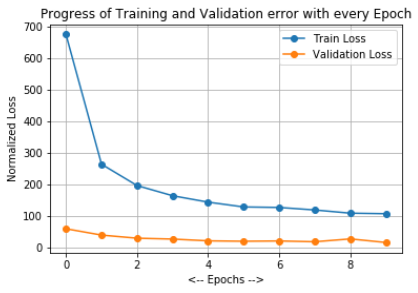


Figure 11: Training Loss plots

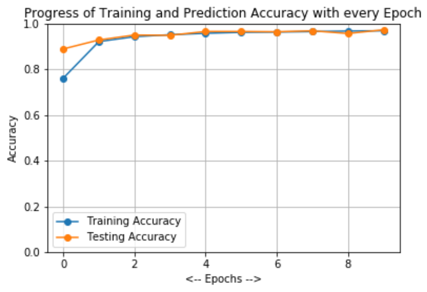


Figure 12: Training Accuracy plots

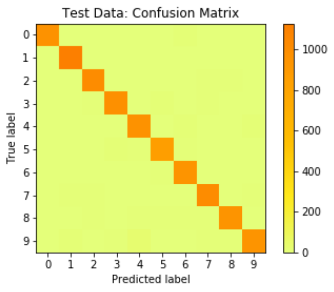
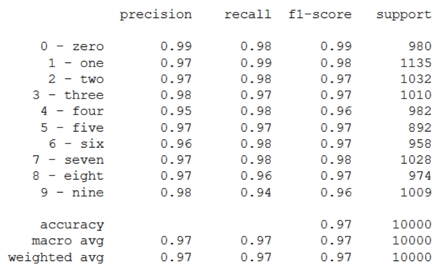


Figure 13: Confusion Matrix of predictions on test set


```
              precision    recall  f1-score   support

    0 - zero       0.99      0.98      0.99       980
     1 - one       0.97      0.99      0.98      1135
     2 - two       0.97      0.98      0.97      1032
   3 - three       0.98      0.97      0.97      1010
    4 - four       0.95      0.98      0.96       982
    5 - five       0.97      0.97      0.97       892
     6 - six       0.96      0.98      0.97       958
   7 - seven       0.97      0.98      0.98      1028
   8 - eight       0.97      0.96      0.97       974
    9 - nine       0.98      0.94      0.96      1009

    accuracy                           0.97     10000
   macro avg       0.97      0.97      0.97     10000
weighted avg       0.97      0.97      0.97     10000
```

Figure 14: Classification Report on the test set

2.2.2 Other Stats

```
Test accuracy for Epoch  10 (Final Epoch) :  0.9719
```

```
Time taken to train and test model:  630.8013446331024
Average prediction accuracy across epochs:  0.94991
Average training accuracy across epochs:  0.9350333333333334
```

2.2.3 Predictions

```
True label:0 predicted as 0
True label:1 predicted as 1
True label:2 predicted as 2
True label:3 predicted as 3
True label:4 predicted as 4
True label:5 predicted as 5
True label:6 predicted as 6
True label:7 predicted as 7
True label:8 predicted as 8
True label:9 predicted as 9
```

2.3 LSTM

Two different hidden layer sizes were tested for the LSTMs.
One with 64 hidden neurons and the other with 128 of them.

2.3.1 Training Plots for h= 128:

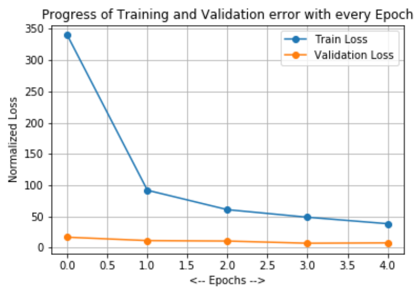


Figure 15: Training Loss plots

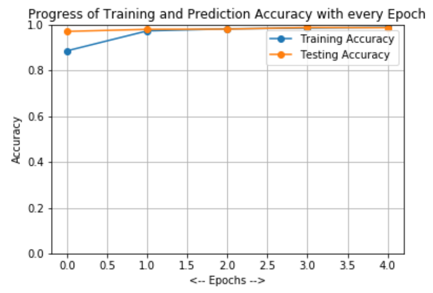


Figure 16: Training Accuracy plots

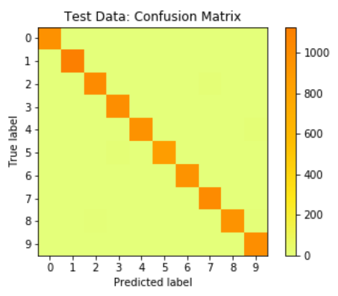
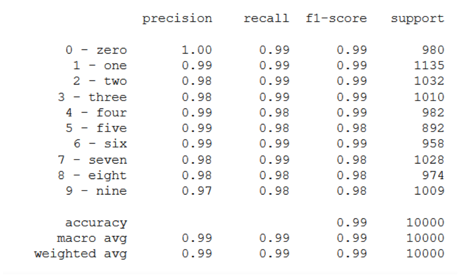


Figure 17: Confusion Matrix of predictions on test set



```
              precision    recall  f1-score   support

    0 - zero       1.00      0.99      0.99       980
     1 - one       0.99      0.99      0.99      1135
     2 - two       0.98      0.99      0.99      1032
   3 - three       0.98      0.99      0.99      1010
    4 - four       0.99      0.98      0.99       982
    5 - five       0.99      0.98      0.98       892
     6 - six       0.99      0.99      0.99       958
   7 - seven       0.98      0.99      0.98      1028
   8 - eight       0.98      0.98      0.98       974
    9 - nine       0.97      0.98      0.98      1009

    accuracy                           0.99     10000
   macro avg       0.99      0.99      0.99     10000
weighted avg       0.99      0.99      0.99     10000
```

Figure 18: Classification Report on the test set

2.3.2 Other Stats for h= 128

```
Test accuracy for Epoch  5 :  0.9858
Time taken to train and test model:  418.2720060348511
Average prediction accuracy across epochs:  0.9800000000000001
Average training accuracy across epochs:  0.9619733333333335
```

2.3.3 Predictions for h= 128

```
True label:0 predicted as 0
True label:1 predicted as 1
True label:2 predicted as 2
True label:3 predicted as 3
True label:4 predicted as 4
True label:5 predicted as 5
True label:6 predicted as 6
True label:7 predicted as 7
True label:8 predicted as 8
True label:9 predicted as 9
```

2.3.4 Training Plots for h= 64:

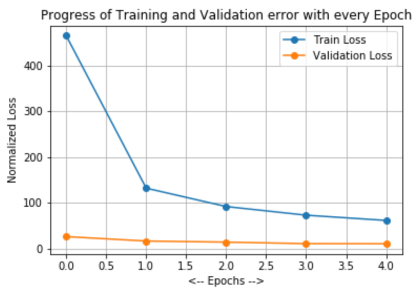


Figure 19: Training Loss plots

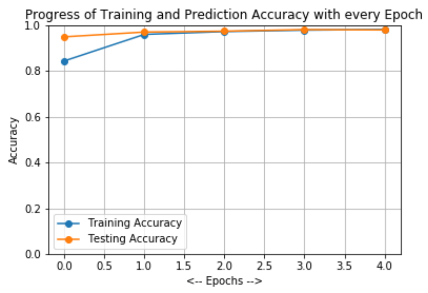


Figure 20: Training Accuracy plots

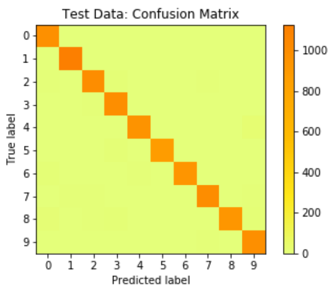
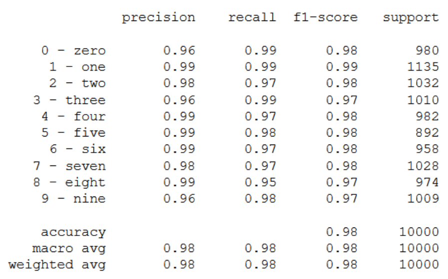


Figure 21: Confusion Matrix of predictions on test set



```
              precision    recall  f1-score   support

    0 - zero       0.96      0.99      0.98       980
     1 - one       0.99      0.99      0.99      1135
     2 - two       0.98      0.97      0.98      1032
   3 - three       0.96      0.99      0.97      1010
    4 - four       0.99      0.97      0.98       982
    5 - five       0.99      0.98      0.98       892
     6 - six       0.99      0.97      0.98       958
   7 - seven       0.98      0.97      0.98      1028
   8 - eight       0.99      0.95      0.97       974
    9 - nine       0.96      0.98      0.97      1009

    accuracy                           0.98     10000
   macro avg       0.98      0.98      0.98     10000
weighted avg       0.98      0.98      0.98     10000
```

Figure 22: Classification Report on the test set

2.3.5 Other Stats for h= 64

```
Test accuracy for Epoch  5 :  0.978
```

```
Time taken to train and test model:  408.54425024986267
Average prediction accuracy across epochs:  0.9693999999999999
Average training accuracy across epochs:  0.9459566666666666
```

2.3.6 Predictions for h= 64

```
True label:0 predicted as 0
True label:1 predicted as 1
True label:2 predicted as 2
True label:3 predicted as 3
True label:4 predicted as 4
True label:5 predicted as 5
True label:6 predicted as 6
True label:7 predicted as 7
True label:8 predicted as 8
True label:9 predicted as 9
```

2.4 GRU

2.4.1 Training Plots:

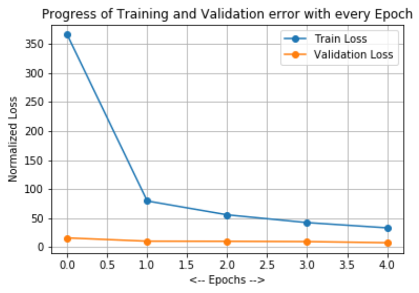


Figure 23: Training Loss plots

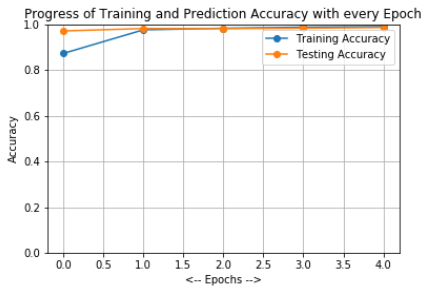


Figure 24: Training Accuracy plots

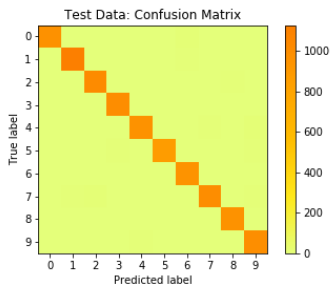
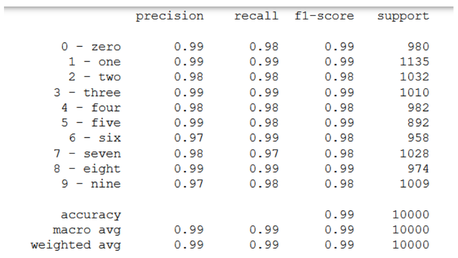


Figure 25: Confusion Matrix of predictions on test set



```
              precision    recall  f1-score   support

    0 - zero       0.99      0.98      0.99       980
     1 - one       0.99      0.99      0.99      1135
     2 - two       0.98      0.98      0.98      1032
   3 - three       0.99      0.99      0.99      1010
    4 - four       0.98      0.98      0.98       982
    5 - five       0.99      0.98      0.99       892
     6 - six       0.97      0.99      0.98       958
   7 - seven       0.98      0.97      0.98      1028
   8 - eight       0.99      0.99      0.99       974
    9 - nine       0.97      0.98      0.98      1009

    accuracy                           0.99     10000
   macro avg       0.99      0.99      0.99     10000
weighted avg       0.99      0.99      0.99     10000
```

Figure 26: Classification Report on the test set

2.4.2 Other Stats

```
Test accuracy for Epoch  5 :  0.9852
Time taken to train and test model:  564.6115252971649
Average prediction accuracy across epochs:  0.9796999999999999
Average training accuracy across epochs:  0.9609366666666668
```

2.4.3 Predictions

```
True label:0 predicted as 0
True label:1 predicted as 1
True label:2 predicted as 2
True label:3 predicted as 3
True label:4 predicted as 4
True label:5 predicted as 5
True label:6 predicted as 6
True label:7 predicted as 7
True label:8 predicted as 8
True label:9 predicted as 9
```

2.5 Bi-Directional LSTM

2.5.1 Training Plots:

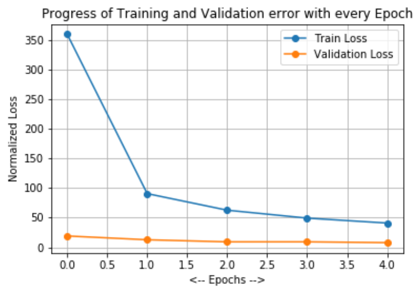


Figure 27: Training Loss plots

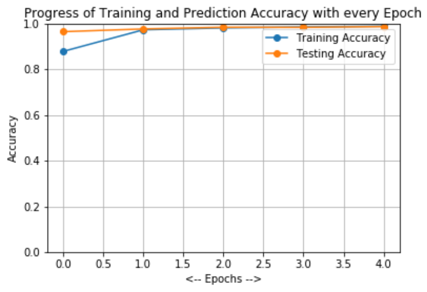


Figure 28: Training Accuracy plots

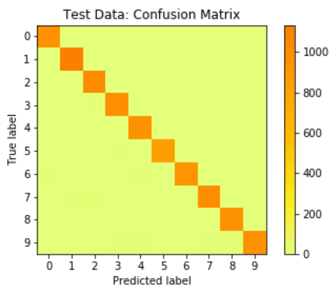
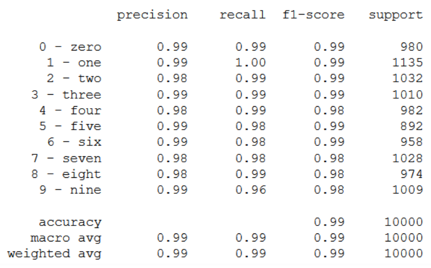


Figure 29: Confusion Matrix of predictions on test set



```
              precision    recall  f1-score   support

    0 - zero       0.99      0.99      0.99       980
     1 - one       0.99      1.00      0.99      1135
     2 - two       0.98      0.99      0.99      1032
   3 - three       0.99      0.99      0.99      1010
    4 - four       0.98      0.99      0.98       982
    5 - five       0.99      0.98      0.99       892
     6 - six       0.99      0.98      0.99       958
   7 - seven       0.98      0.98      0.98      1028
   8 - eight       0.98      0.99      0.98       974
    9 - nine       0.99      0.96      0.98      1009

    accuracy                           0.99     10000
   macro avg       0.99      0.99      0.99     10000
weighted avg       0.99      0.99      0.99     10000
```

Figure 30: Classification Report on the test set

2.5.2 Other Stats

```
Test accuracy for Epoch  5 :  0.986
```

```
Time taken to train and test model:  764.6242773532867
Average prediction accuracy across epochs:  0.97818
Average training accuracy across epochs:  0.9602666666666666
```

2.5.3 Predictions

```
True label:0 predicted as 0
True label:1 predicted as 1
True label:2 predicted as 2
True label:3 predicted as 3
True label:4 predicted as 4
True label:5 predicted as 5
True label:6 predicted as 6
True label:7 predicted as 7
True label:8 predicted as 8
True label:9 predicted as 9
```

2.6 With L2 Regularization

An L_2 regularization was implemented just on the neural network weights with a scaling coefficient of $\lambda = 0.001$ on an LSTM network with 128 hidden units.

2.6.1 Training Plots:

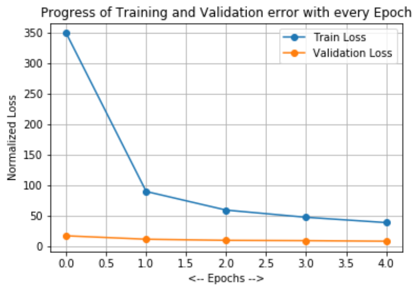


Figure 31: Training Loss plots

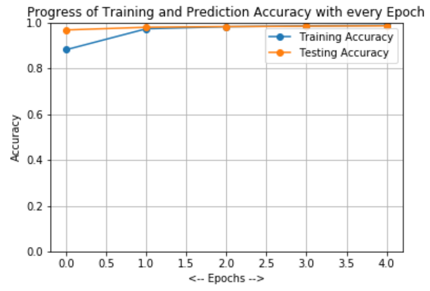


Figure 32: Training Accuracy plots

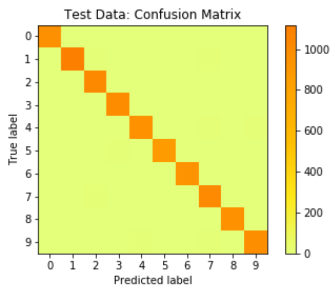
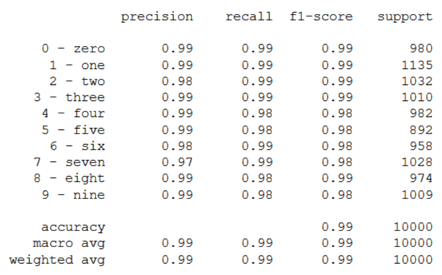


Figure 33: Confusion Matrix of predictions on test set



```
              precision    recall  f1-score   support

    0 - zero       0.99      0.99      0.99       980
     1 - one       0.99      0.99      0.99      1135
     2 - two       0.98      0.99      0.99      1032
   3 - three       0.99      0.99      0.99      1010
    4 - four       0.99      0.98      0.98       982
    5 - five       0.99      0.98      0.98       892
     6 - six       0.98      0.99      0.98       958
   7 - seven       0.97      0.99      0.98      1028
   8 - eight       0.99      0.98      0.99       974
    9 - nine       0.99      0.98      0.98      1009

    accuracy                           0.99     10000
   macro avg       0.99      0.99      0.99     10000
weighted avg       0.99      0.99      0.99     10000
```

Figure 34: Classification Report on the test set

2.6.2 Other Stats

```
Test accuracy for Epoch  5 :  0.9852
Time taken to train and test model:  260.74275064468384
Average prediction accuracy across epochs:  0.97934
Average training accuracy across epochs:  0.9614033333333334
```

2.6.3 Predictions

```
True label:0 predicted as 0
True label:1 predicted as 1
True label:2 predicted as 2
True label:3 predicted as 3
True label:4 predicted as 4
True label:5 predicted as 5
True label:6 predicted as 6
True label:7 predicted as 7
True label:8 predicted as 8
True label:9 predicted as 9
```

2.7 Observations

- First, comparing the RNNs, LSTMs and GRU, we see that for this particular experiment, LSTMs and GRUs outperform the RNN while they train for just half the number of Epochs. Between GRUs and LSTMs there is not much different in the accuracies.
- It is interesting to see that the GRU took longer to train. However, this trend did not carry forward on repeating the experiments.
- As we compare the size of the hidden layers, we see that the one with h = 128 performs better than h = 64 which is expected as h is no where near the size of the Manifold as seen from the previous experiment. Thus, for a larger h, more information is retained and therefore carry forward to the later time instants.
- We see that the one with a smaller hidden size takes slightly lesser to train.
- We see that the bidirectional LSTM didn't improve the accuracy by a lot while taking almost double the time to train.

- The regularized LSTM trains much much faster compared to the other models and has a comparable accuracy too, due to the low value of the regularization coefficient.This brings to light the tradeoff between model complexity and computational complexity.

3 Remembering Index in a sequence

For this experiment, the models were first trained using a fixed index for remembering but varying length sequences (between 3-10 digits).
This experiment was performed by training the models for 30 epochs to squeeze the maximum benefit. All the other hyperparameters were kept the same.
The training and test data were generated on the fly.
The experiment was performed over LSTM, RNN and GRU to ascertain the best model.
A sample architecture is shown below:

```
Vanilla_GRU(
  (gru): GRU(10, hidden layer, batch_first=True)
  (HL): Linear(in_features=hidden layer, out_features=10, bias=True)
  (logsoftmax): LogSoftmax(dim=1)
)
```

Note that the input size is now 10.

3.1 RNN

3.1.1 h = 2

Training Plots

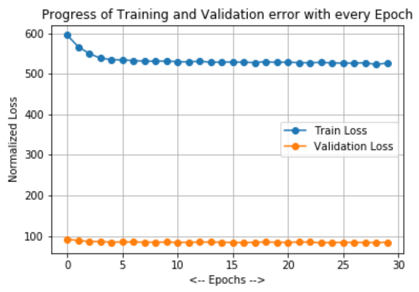


Figure 35: Training Loss plots

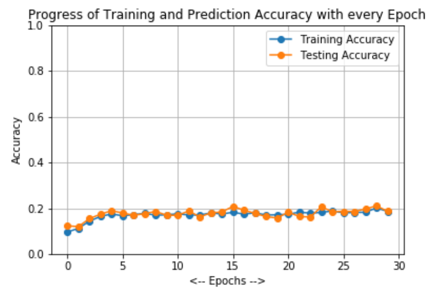


Figure 36: Training Accuracy plots

3.1.2 h =5

Training Plots

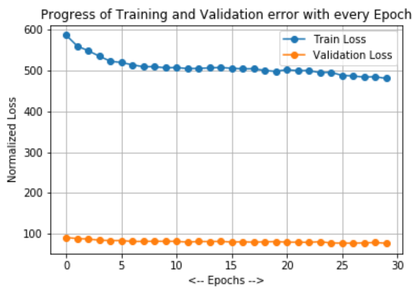


Figure 37: Training Loss plots

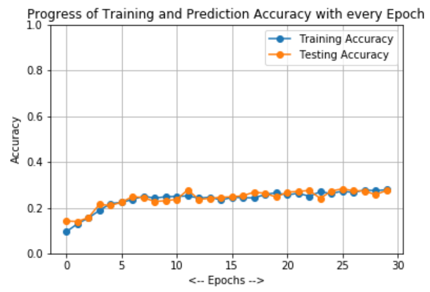


Figure 38: Training Accuracy plots

3.1.3 h = 10

Training Plots

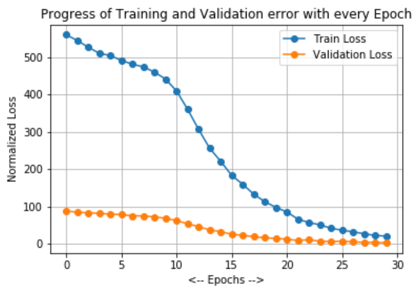


Figure 39: Training Loss plots

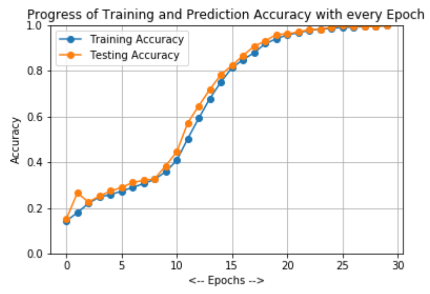


Figure 40: Training Accuracy plots

3.2 LSTM

3.2.1 h = 2

Training Plots

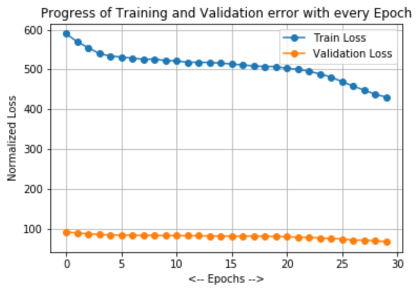


Figure 41: Training Loss plots

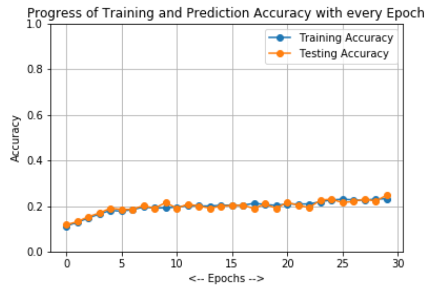


Figure 42: Training Accuracy plots

3.2.2 h =5

Training Plots

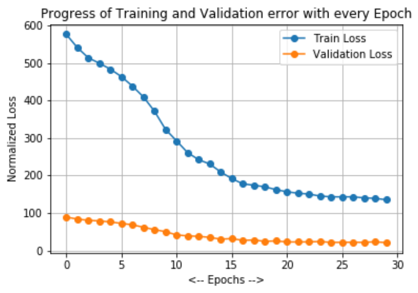


Figure 43: Training Loss plots

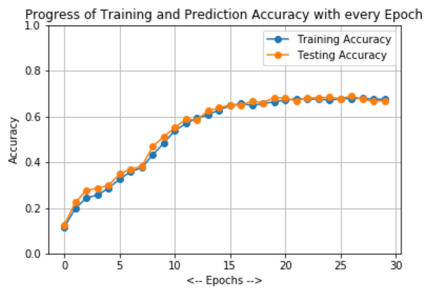


Figure 44: Training Accuracy plots

3.2.3 h = 10

Training Plots

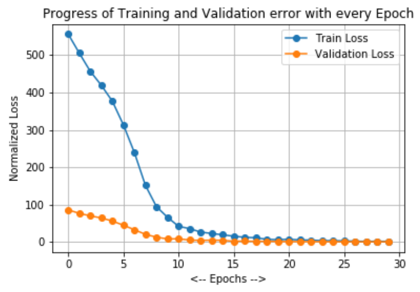


Figure 45: Training Loss plots

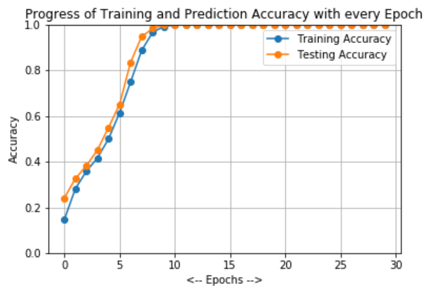


Figure 46: Training Accuracy plots

3.3 GRU

3.3.1 h = 2

Training Plots

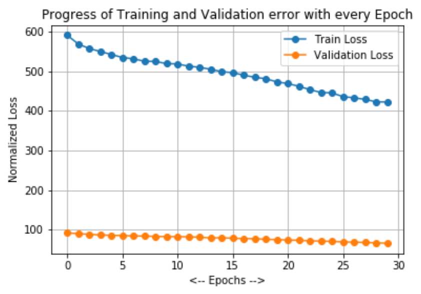


Figure 47: Training Loss plots

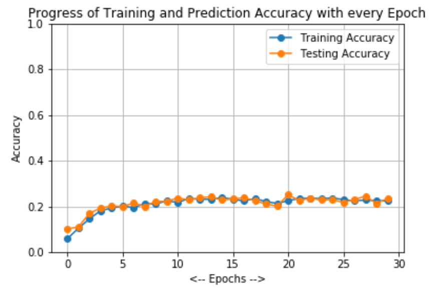


Figure 48: Training Accuracy plots

3.3.2 h =5

Training Plots

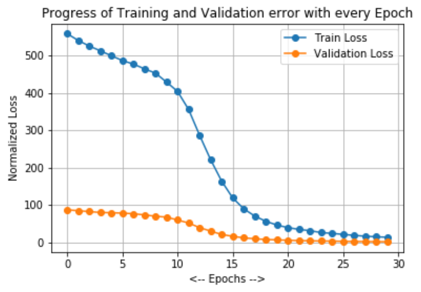


Figure 49: Training Loss plots

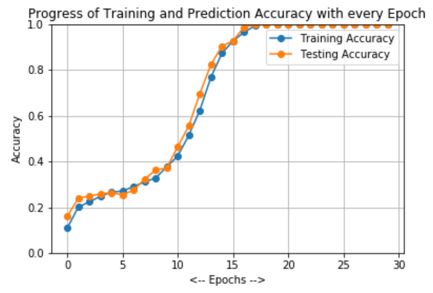


Figure 50: Training Accuracy plots

3.3.3 h = 10

Training Plots

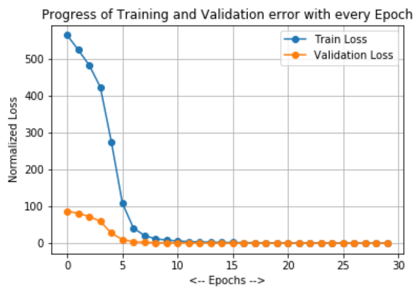


Figure 51: Training Loss plots

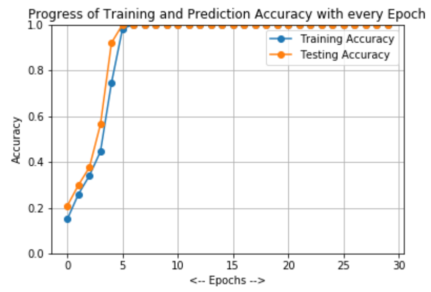


Figure 52: Training Accuracy plots

3.4 Prediction Outputs

Shown below are prediction outputs for a GRU trained with 10 hidden units.
Note: K is the index to be remembered.

```
For K = 3
Generated Sequence:tensor([[6, 8, 0, 2, 8, 7]], dtype=torch.int32)
Predicted Output:tensor([0])
Generated Sequence:tensor([[1, 6, 6, 2, 8]], dtype=torch.int32)
Predicted Output:tensor([6])
Generated Sequence:tensor([[1, 5, 1, 0, 1]], dtype=torch.int32)
Predicted Output:tensor([1])
Generated Sequence:tensor([[5, 5, 1, 3, 5]], dtype=torch.int32)
Predicted Output:tensor([1])
Generated Sequence:tensor([[2, 3, 5, 8, 2, 5, 2]], dtype=torch.int32)
Predicted Output:tensor([5])

For K = 2
Generated Sequence:tensor([[4, 5, 8]], dtype=torch.int32)
Predicted Output:tensor([5])
Generated Sequence:tensor([[6, 8, 7, 8]], dtype=torch.int32)
Predicted Output:tensor([8])
Generated Sequence:tensor([[7, 1, 1, 0, 7, 8]], dtype=torch.int32)
Predicted Output:tensor([1])
Generated Sequence:tensor([[4, 2, 3, 6, 5, 1, 3]], dtype=torch.int32)
Predicted Output:tensor([2])
Generated Sequence:tensor([[1, 8, 7, 7, 3, 7]], dtype=torch.int32)
Predicted Output:tensor([8])
```

3.5 Observations

- We see that as the size of the hidden length increases, the capability to remember increases and therefore the accuracy also improves.
- Amongst RNNs, LSTMs and GRUs, the latter performs the best as the inputs are all small length sequences. LSTMs and GRUs hit a 100 percent accuracy for their last and last two models respectively.
- GRUs converge much faster than LSTMs. The last two observations thus lend to the credibility of GRUs and confirm what we saw in class.
- Observing the predicted outputs, we can see that the trained models indeed remember the index very well.

4 Binary Addition

As in the previous case, all the training data was generated on the fly but with a fixed size of the inputs.
7 Epochs was used to train the models and a learning rate of 0.01 was used.
The sample architecture of the LSTM used is given below:

```
Binary_LSTM(
  (lstm): LSTM(2, hidden size, batch_first=True)
  (HL): Linear(in_features=hidden size, out_features=1, bias=True)
  (sigmoid): Sigmoid()
)
```

Note that the input size is now 2 and the output is just 1 neuron with a sigmoid activation.
First an LSTM with hidden layer size of 3 was trained with the input sequence being of size 3.
Next the hidden layer of the above LSTM was updated to 5 neurons however the size of the input sequence was kept the same.
For the last model, we update the size of the input sequence to 7 while keeping the hidden layer the same.

4.1 h = 3

4.1.1 CE Loss

Training Plots

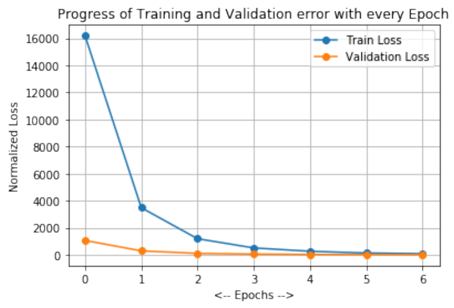


Figure 53: Training Loss plots

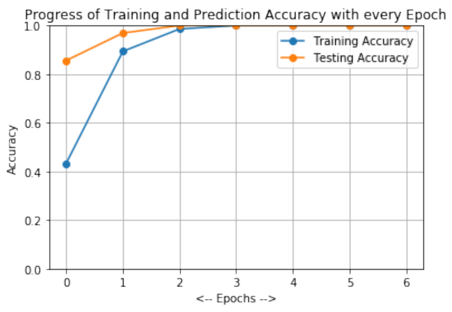


Figure 54: Training Accuracy plots

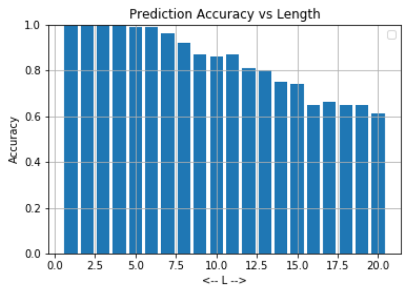


Figure 55: Accuracy plot across different input length

Given below is the test accuracy over 100 samples on input length ranging from 1 to 20.

```
Prediction Accuracies : [1.0, 1.0, 1.0, 1.0, 0.99, 0.99, 0.96, 0.92, 0.87,
0.86, 0.87, 0.81, 0.8, 0.75, 0.74, 0.65, 0.66, 0.65, 0.65, 0.61]
```

4.1.2 MSE Loss

Training Plots

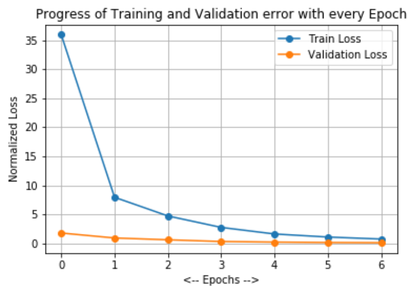


Figure 56: Training Loss plots

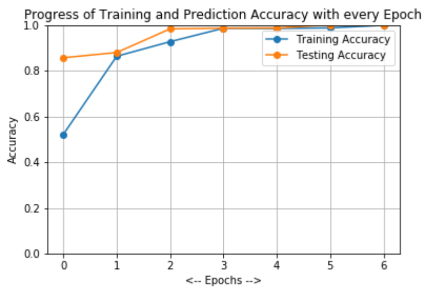


Figure 57: Training Accuracy plots

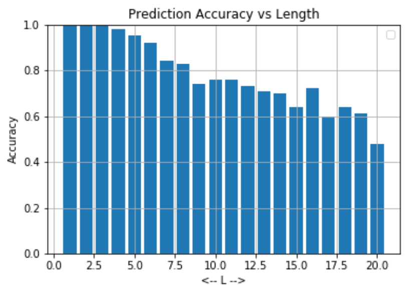


Figure 58: Accuracy plot across different input length

Given below is the test accuracy over 100 samples on input length ranging from 1 to 20.

```
Prediction Accuracies : [1.0, 1.0, 1.0, 0.98, 0.95, 0.92, 0.84, 0.83, 0.74,
0.76, 0.76, 0.73, 0.71, 0.7, 0.64, 0.72, 0.6, 0.64, 0.61, 0.48]
```

4.2 h = 5

4.2.1 CE Loss

Training Plots

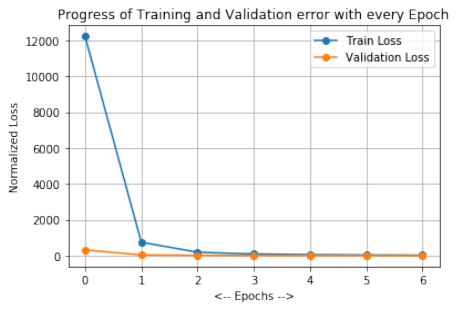


Figure 59: Training Loss plots

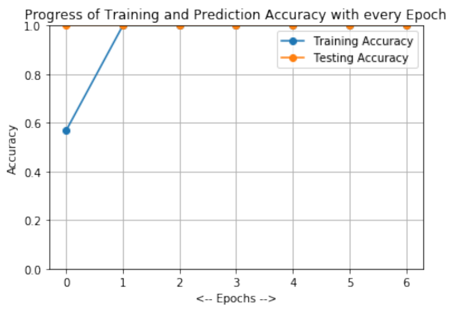


Figure 60: Training Accuracy plots

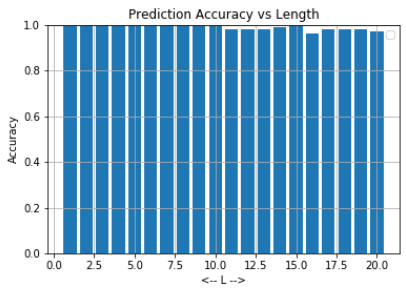


Figure 61: Accuracy plot across different input length

Given below is the test accuracy over 100 samples on input length ranging from 1 to 20.

```
Prediction Accuracies : [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
0.98, 0.98, 0.98, 0.99, 1.0, 0.96, 0.98, 0.98, 0.98, 0.97]
```

4.2.2 MSE Loss

Training Plots

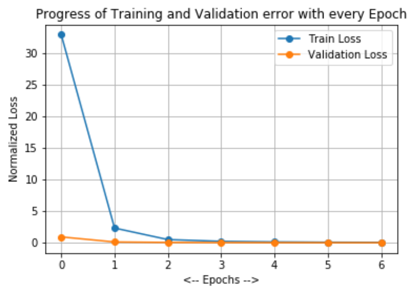


Figure 62: Training Loss plots

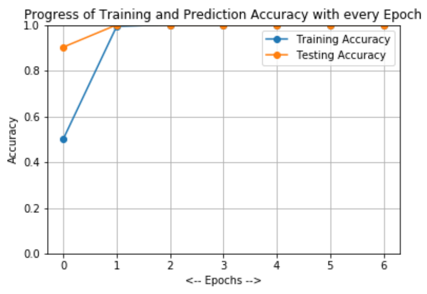


Figure 63: Training Accuracy plots

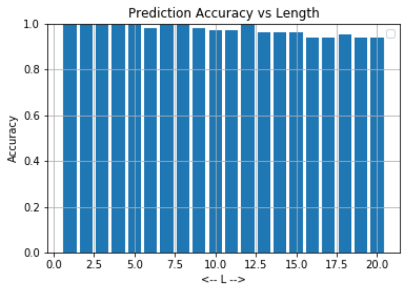


Figure 64: Accuracy plot across different input length

Given below is the test accuracy over 100 samples on input length ranging from 1 to 20.

```
Prediction Accuracies : [1.0, 1.0, 1.0, 1.0, 1.0, 0.98, 1.0, 1.0, 0.98, 0.97,
0.97, 1.0, 0.96, 0.96, 0.96, 0.94, 0.94, 0.95, 0.94, 0.94]
```

4.3 L = 7

4.3.1 CE Loss

Training Plots

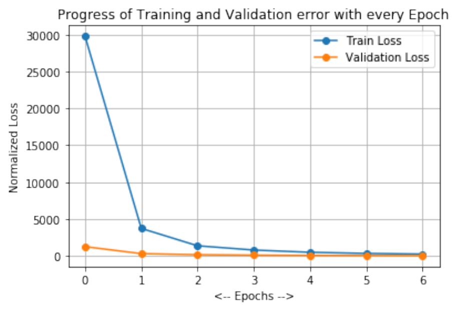


Figure 65: Training Loss plots

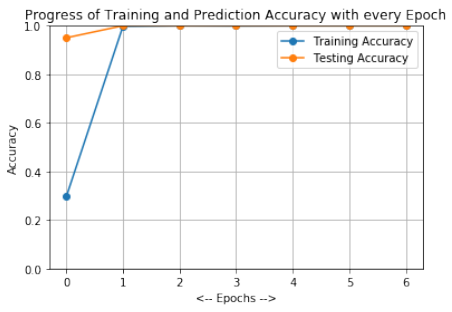


Figure 66: Training Accuracy plots

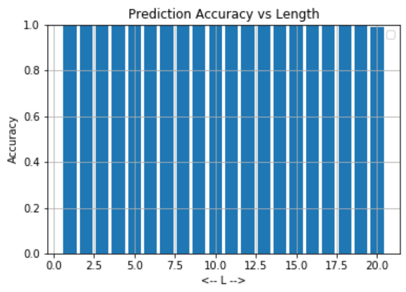


Figure 67: Accuracy plot across different input length

Given below is the test accuracy over 100 samples on input length ranging from 1 to 20.

```
Prediction Accuracies : [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.99]
```

4.3.2 MSE Loss

Training Plots

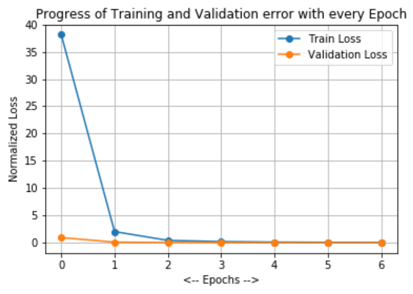


Figure 68: Training Loss plots

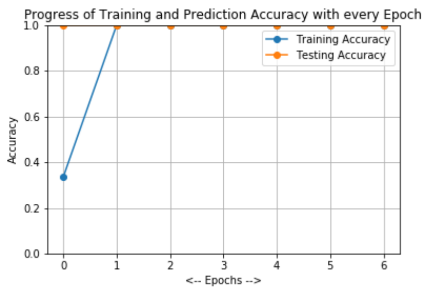


Figure 69: Training Accuracy plots

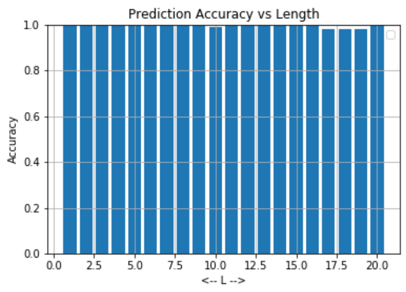


Figure 70: Accuracy plot across different input length

Given below is the test accuracy over 100 samples on input length ranging from 1 to 20.

```
Prediction Accuracies : [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.99,
1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.98, 0.98, 0.98, 1.0]
```

4.4 Observations

- Comparing the losses, we see that the cross entropy loss performs marginally better than the MSE loss, but only for a small hidden layer size.
- The cross entropy loss takes approximately 1-2 seconds more to train than the MSE loss.
- Comparing the size of the hidden layers, we see that as we increase the hidden layer to 5, we get an accuracy in the higher 90s everywhere for both the loss functions which is what we'd expect.
- Additionally, as we increase the training input size, we see that the accuracy now becomes 100 almost everywhere.
- Thus, we see that the RNN is able to crack the code with very few hidden layer neurons and even for a small input size. The training time is also not a lot and the accuracy is perhaps better than humans as they're prone to silly mistakes !!