EE6132 Assignment No. 2: Report

-Akash Reddy A, EE17B001

September 26, 2019

Introductory Notes

PyTorch has been leveraged to construct all the models in this assignment and answer the sub-questions as well. `torchvision.datasets` and `torchvision.transforms` have been used to extract the MNIST datasets directly from the website and convert them into Tensors. The code has been written into a Google Colab notebook, linked here.

1 MNIST Classification Using RNN

Submissions

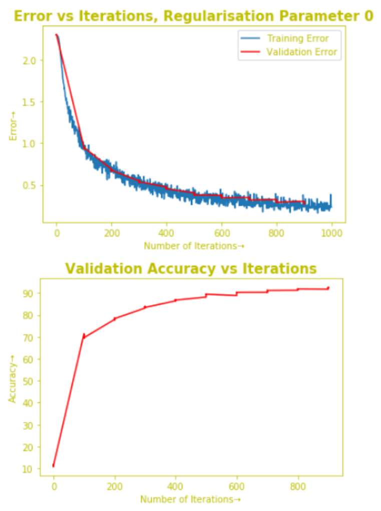
1

Hyperparameters: regularisation parameters λ for each case and hidden layer size $h = 100$ were selected based on experimentation, and also a paper on RNN digit classification linked here. $h = 100$ was selected as it gave nearly the same performance as $h = 128$ (which is suggested in the paper) due to the large training set size, but with a smaller number of parameters and subsequently higher computational efficiency.

2

Vanilla RNN

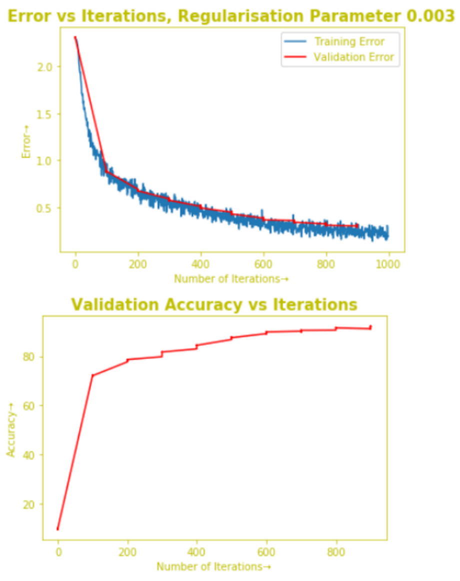
We observe an increase in the test accuracy for $\lambda = 0.003$. Regularisation, therefore, helps the vanilla RNN model.



```
Tester(RNN)
RandomTrueVsPredicted(RNN, num_examples = 7)
```

```
Test Accuracy of the Network =  93.23 %
```

```
True labels =  [0, 1, 4, 5, 1, 1, 0] , Predicted labels =  [0, 1, 9, 5, 1, 1, 0]
```



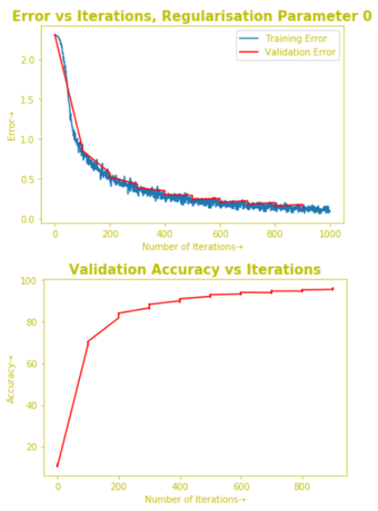
```
Tester(rRNN)
```

```
Test Accuracy of the Network =  93.62 %
```

Figure 1: Vanilla RNN: Error and Accuracy Plots, Final Test Accuracy Without and With Regularisation Respectively, Test Samples Without Regularisation

GRU

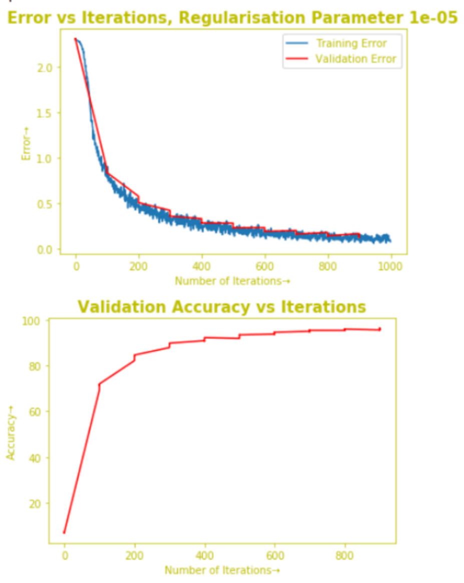
We observe that the unregularised test accuracy is very high for the GRU. Even a small regularisation rate of 0.00001 reduces the accuracy. We can infer from this that the GRU model does not overfit, and any regularisation seems to worsen the accuracy.



```
Tester(GRU)
RandomTrueVsPredicted(GRU, num_examples = 7)
```

```
Test Accuracy of the Network =  96.89999999999999 %
```

```
True labels =  [2, 5, 3, 3, 7, 1, 1] , Predicted labels =  [2, 5, 3, 3, 7, 1, 1]
```



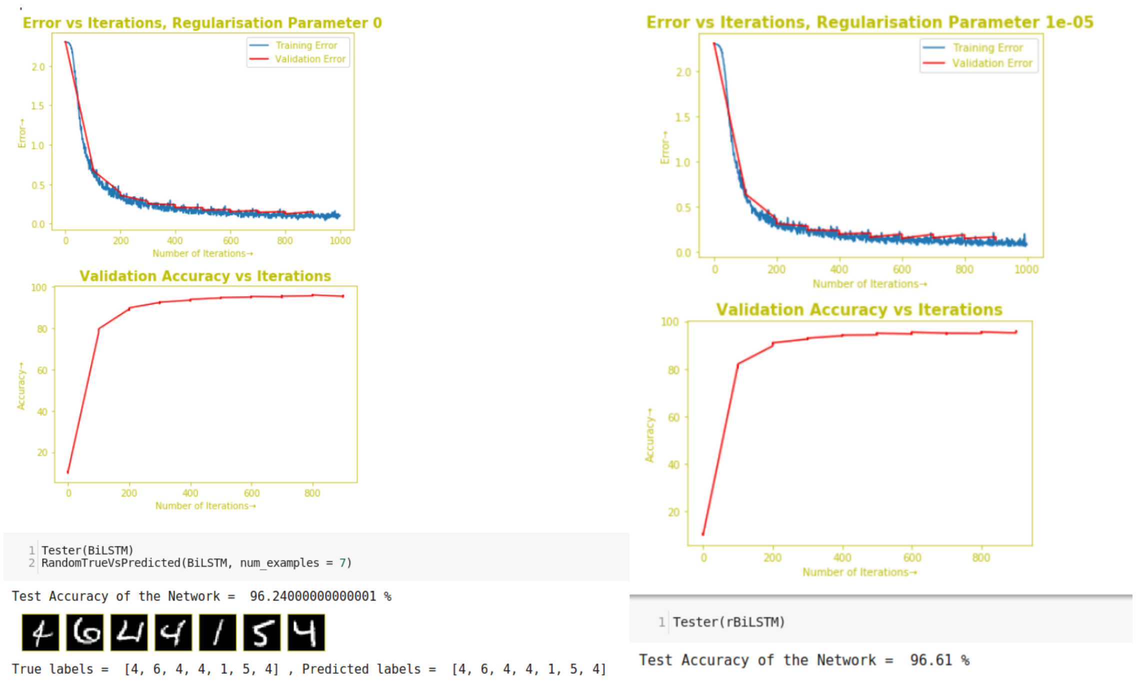
```
Tester(rGRU)
```

```
Test Accuracy of the Network =  96.87 %
```

Figure 2: GRU: Error and Accuracy Plots, Final Test Accuracy Without and With Regularisation Respectively, Test Samples Without Regularisation

Bidirectional LSTM

This model seems to be affected by regularisation, however the parameter $\lambda = 0.00001$ that causes a positive change is really small.



```
Tester(BiLSTM)
RandomTrueVsPredicted(BiLSTM, num_examples = 7)
```

```
Test Accuracy of the Network =  96.24000000000001 %
```

```
True labels =  [4, 6, 4, 4, 1, 5, 4] , Predicted labels =  [4, 6, 4, 4, 1, 5, 4]
```

```
Tester(rBiLSTM)
```

```
Test Accuracy of the Network =  96.61 %
```

Figure 3: Bidirectional LSTM: Error and Accuracy Plots, Final Test Accuracy Without and With Regularisation Respectively, Test Samples Without Regularisation

2 Remembering a Number at a Particular Index in a Given Sequence

The RNN (LSTM) has been trained to remember the number at the **second** index. A validation set has been used to keep overfitting in check. Training set is made with random-length sequences with lengths in the range of 3-10 as required by the question.

LSTM Models can learn to remember the second number for even longer sequences. Upon trying a vanilla RNN model for this problem, I discovered that it functions well for input sequences of length 3, average for length 5, and abysmal for length 10. This is due to the fact that vanilla RNNs have no long-term memory. Hence, I picked an LSTM Model to solve this problem.

Submissions

1

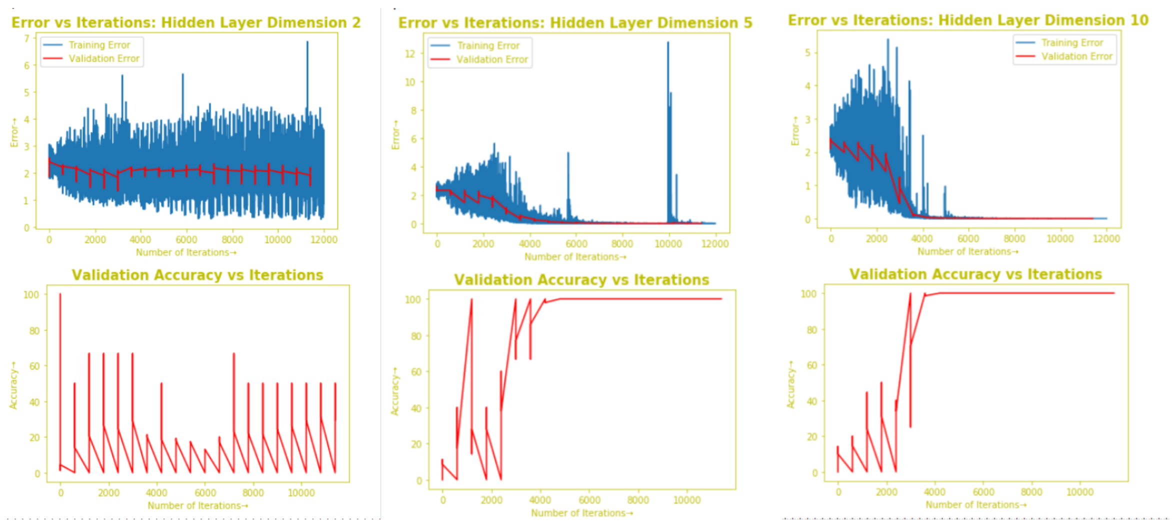
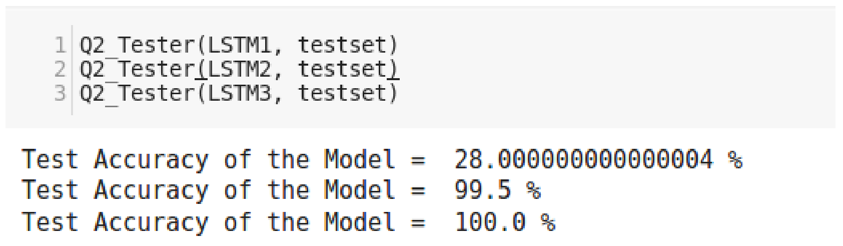


Figure 4: Error and Accuracy Plots for Hidden Layer Dimensions 2, 5, 10

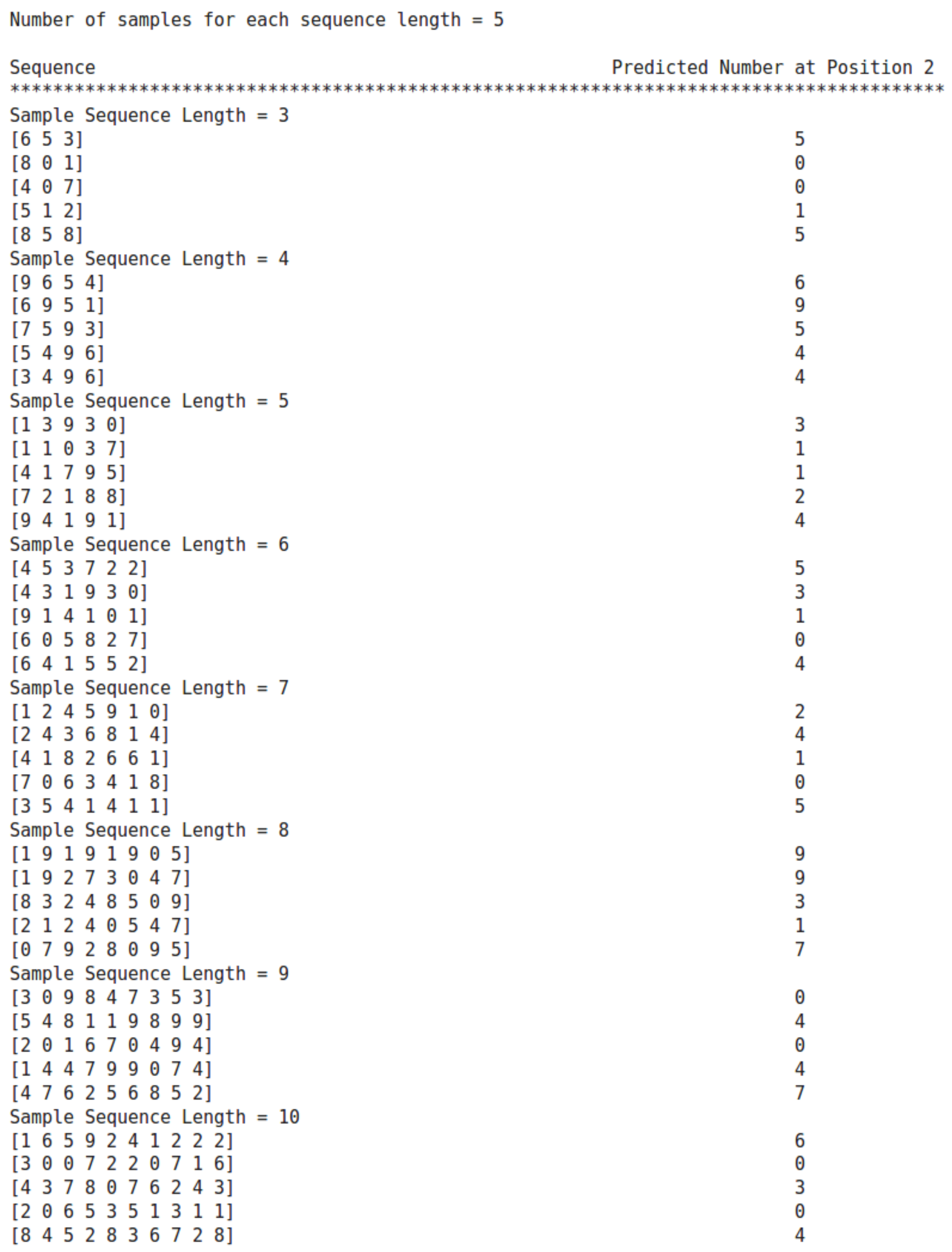


```
Q2_Tester(LSTM1, testset)
Q2_Tester(LSTM2, testset)
Q2_Tester(LSTM3, testset)
```

```
Test Accuracy of the Model =  28.000000000000004 %
Test Accuracy of the Model =  99.5 %
Test Accuracy of the Model =  100.0 %
```

Figure 5: Respective Test Set Accuracies

2



```
Number of samples for each sequence length = 5

Sequence                                                    Predicted Number at Position 2
*************************************************************************************
Sample Sequence Length = 3
[6 5 3]                                                                    5
[8 0 1]                                                                    0
[4 0 7]                                                                    0
[5 1 2]                                                                    1
[8 5 8]                                                                    5
Sample Sequence Length = 4
[9 6 5 4]                                                                  6
[6 9 5 1]                                                                  9
[7 5 9 3]                                                                  5
[5 4 9 6]                                                                  4
[3 4 9 6]                                                                  4
Sample Sequence Length = 5
[1 3 9 3 0]                                                                3
[1 1 0 3 7]                                                                1
[4 1 7 9 5]                                                                1
[7 2 1 8 8]                                                                2
[9 4 1 9 1]                                                                4
Sample Sequence Length = 6
[4 5 3 7 2 2]                                                              5
[4 3 1 9 3 0]                                                              3
[9 1 4 1 0 1]                                                              1
[6 0 5 8 2 7]                                                              0
[6 4 1 5 5 2]                                                              4
Sample Sequence Length = 7
[1 2 4 5 9 1 0]                                                            2
[2 4 3 6 8 1 4]                                                            4
[4 1 8 2 6 6 1]                                                            1
[7 0 6 3 4 1 8]                                                            0
[3 5 4 1 4 1 1]                                                            5
Sample Sequence Length = 8
[1 9 1 9 1 9 0 5]                                                          9
[1 9 2 7 3 0 4 7]                                                          9
[8 3 2 4 8 5 0 9]                                                          3
[2 1 2 4 0 5 4 7]                                                          1
[0 7 9 2 8 0 9 5]                                                          7
Sample Sequence Length = 9
[3 0 9 8 4 7 3 5 3]                                                        0
[5 4 8 1 1 9 8 9 9]                                                        4
[2 0 1 6 7 0 4 9 4]                                                        0
[1 4 4 7 9 9 0 7 4]                                                        4
[4 7 6 2 5 6 8 5 2]                                                        7
Sample Sequence Length = 10
[1 6 5 9 2 4 1 2 2 2]                                                      6
[3 0 0 7 2 2 0 7 1 6]                                                      0
[4 3 7 8 0 7 6 2 4 3]                                                      3
[2 0 6 5 3 5 1 3 1 1]                                                      0
[8 4 5 2 8 3 6 7 2 8]                                                      4
```

Figure 6: Varied-Length Sample Results for Best Model (Hidden Layer Dimension= 10)

3 Adding Two Binary Strings

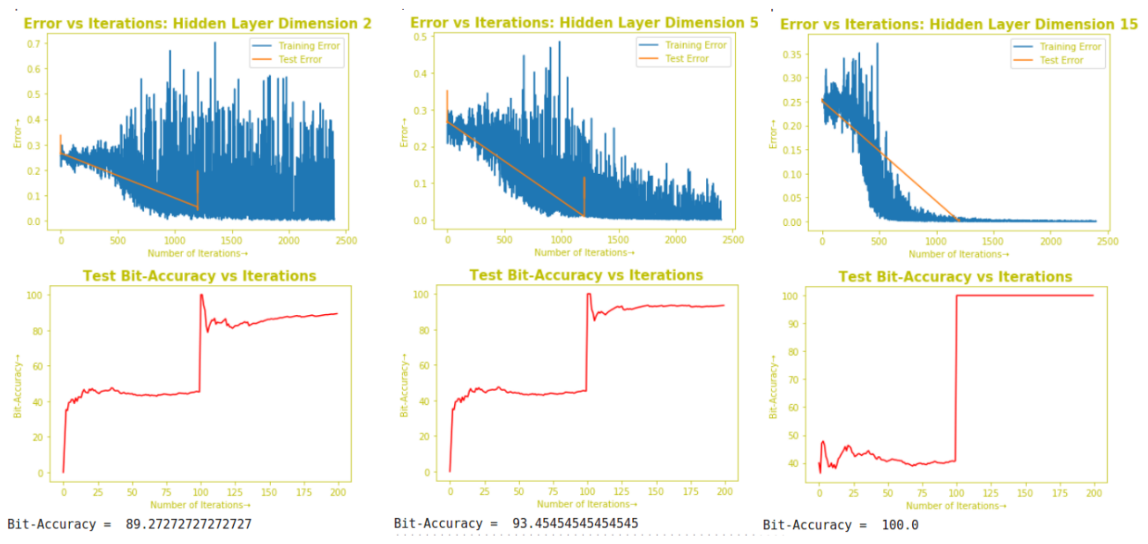
Data Preparation

One extra zero each has been prefixed to each of the addends of length L in case the sum is of length L+1. In this question, the number of epochs and batch size have been chosen such that the difference between the various cases become more apparent.

Submissions

Experiment 1: Changing Hidden Layer Dimension

As we increase the increase the hidden layer dimension, we observe an increase in the bit-accuracy. (This is for input length 5.)



```
Bit-Accuracy =  89.27272727272727
Bit-Accuracy =  93.45454545454545
Bit-Accuracy =  100.0
```

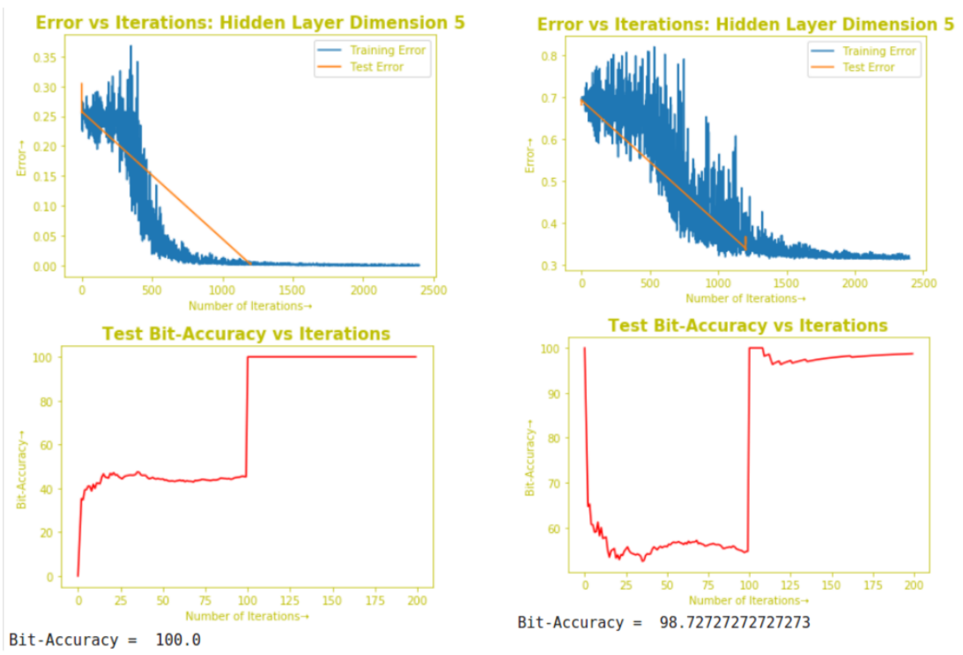
Figure 7: Plots and Accuracy for Hidden Layer Dimension 2, 5, 15 Respectively

Experiment 2: MSE vs Cross-Entropy Loss

Here, my network for the Cross-Entropy Loss has 2 output neurons, due to the output in the Cross-Entropy Loss model being a probability vecctor, with two probabilities of output digit being either 0 or 1.

We find that MSE Loss does better than Cross-Entropy Loss. The training loss converges to 0 faster, and also the test accuracy reaches closer to 100% in MSE Loss.

This could be because the MSE Loss gives a sort of quadratic penalty to the output neurons, hence always pushing the weights to output the correct class, 1 or 0. On the other hand, the Cross Entropy Loss gives penalty based on probabilistic outputs. As soon as the probability for the correct digit (1 or 0) becomes ¿0.5, it is accepted and is penalised lesser, and hence convergence occurs at a slower rate.



```
Bit-Accuracy =  100.0
```

```
Bit-Accuracy =  98.72727272727273
```

Figure 8: Plots and Accuracy for MSE Loss (LHS) vs Cross-Entropy Loss (RHS)

3.1 Experiment 3: Bit-Accuracies for Various Input Lengths

The bit-accuracies for varying length inputs in the range of 1-20 have been plotted for both MSE Loss and Cross-Entropy Loss models, with varying hidden layer dimension.

The bit-accuracies are bad for larger digits hidden layer size 2 for all input lengths, as the small hidden layer fails to capture features well. The effect of increasing input length doesn't seem to improve accuracy much.

The accuracies are also **bad for larger digits** in the top-right most graph, where the hidden layer dimension is large (h = 15) enough to capture features, BUT the training inputs are short (L = 3). This is because the network has not learnt how to accurately calculate the sum for longer binary numbers since the training inputs are all short. There is a general trend of improvement as we increase the size of training inputs (network can learn to add longer binary numbers more accurately) and the hidden layer dimension (network can capture more features).

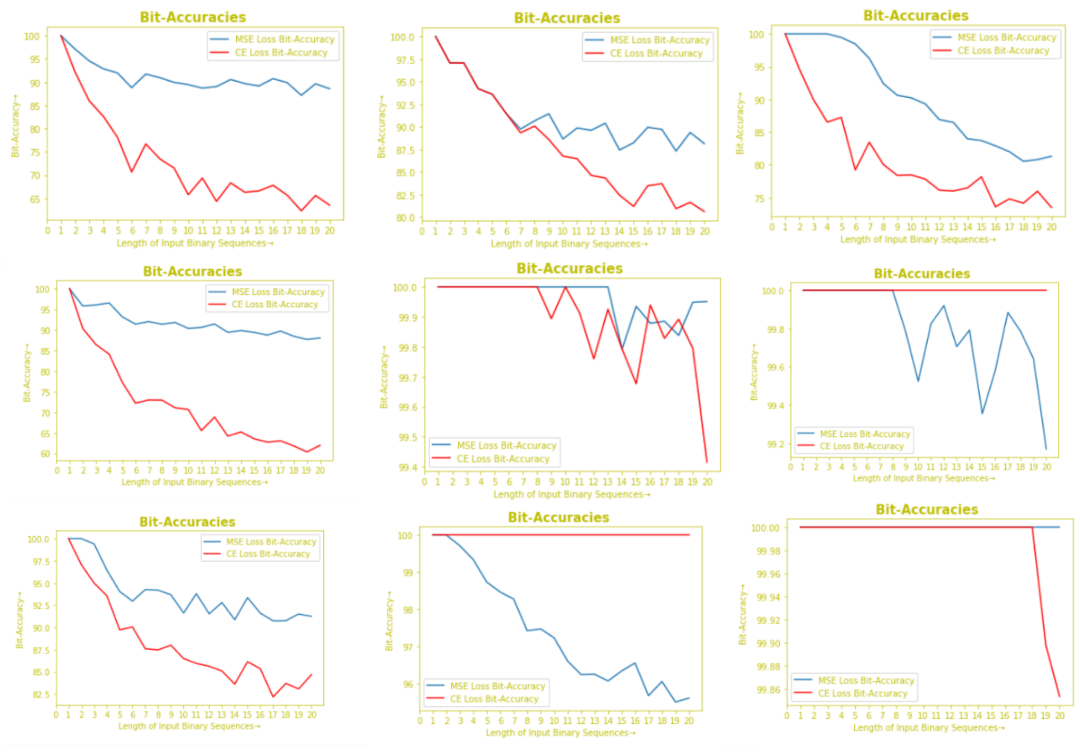


Figure 9: Bit-Accuracies: Rows- Input Lengths L = 3, L = 5, L = 10 Respectively. Columns- Hidden Layer Dimensions h = 2, h = 5, h = 15 Respectively.

* *