Programming Assignment 3

R KAUSHIK — EE15B105

Octoder 28, 2018

1 MNIST digits classification using RNN

The MNIST data set consisting of 60000 training images is divided into training set of size 51000 and validation set of size 9000. A basic RNN model and two of its variants - LSTM and bidirectional RNN, are used in the experiments. In all the experiments, hyperparameters like number of hidden layers and sizes of hidden layers were tuned using the results from thorough cross validation. For all class of models, the following variation of hyperparameters participated in the evaluation of different model architectures using cross-validation.

$$[64], [128], [256], [64, 64], [128, 128], [256, 256], [256, 128], [128, 256]$$

Where $[N, M]$ represents the RNN cells with 2 stacked hidden layers with N units in first layer and M units in second layer. $[N]$ represents an architecture with single hidden layer having N units

1.1 Cross-validation

RNN

Architecture	$[64]$	$[128]$	$[256]$	$[64, 64]$	$[128, 128]$	$[256, 256]$	$[256, 128]$	$[128, 256]$
Validation accuracy (in %)	92.95	94.87	**94.93**	94.06	**94.96**	93.42	93.49	93.81

Table 1: Cross-validation results for different architectures of basic RNN model

The model with architecture $[256]$ and $[128, 128]$ perform the best with a very minimal difference in the cross-validation accuracy. Hence following the Minimum Description Length rule, the model with architecture $[256]$ is chosen, since it has lesser number of parameters than the other model, and is used for evaluation on test dataset.

LSTM

Architecture	$[64]$	$[128]$	$[256]$	$[64, 64]$	$[128, 128]$	$[256, 256]$	$[256, 128]$	$[128, 256]$
Validation accuracy (in %)	96.98	97.30	97.10	95.96	97.89	**98.14**	**98.10**	97.85

Table 2: Cross-validation results for different architectures of LSTM model

The model with architecture $[256, 256]$ and $[256, 128]$ perform the best with a very minimal difference in the cross-validation accuracy and considerable improvement in performance over the next best model. Hence following the Minimum Description Length rule, the model with architecture $[256, 128]$ is chosen, since it has lesser number of parameters than the other model, and is used for evaluation on test dataset.

Bidirectional RNN

Architecture	$[64]$	$[128]$	$[256]$	$[64, 64]$	$[128, 128]$	$[256, 256]$	$[256, 128]$	$[128, 256]$
Validation accuracy (in %)	97.27	**97.61**	97.11	96.73	96.89	96.61	96.08	96.06

Table 3: Cross-validation results for different architectures of Bidirectional RNN model

The model with architecture $[128]$ performs considerably better than model with other architectures. The best architecture also corresponds to one of those with less number of trainable parameters. Hence model with architecture $[128]$ can be chosen unanimously. This model will be used for evaluation on test set.

1.2 Test performance

- **RNN**
  Average prediction accuracy on test set = **97.29 %**

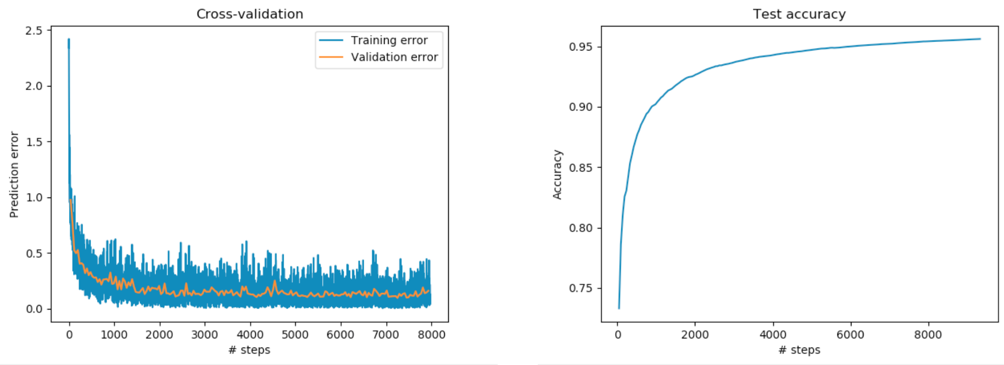


Figure 1: Plots showing the convergence of training and validation errors (right) and the accuracy on test set (left) as training progresses. The plots correspond to training of a basic RNN model with single hidden layer containing 256 units

- **LSTM**
  Average prediction accuracy on test set = **98.55 %**

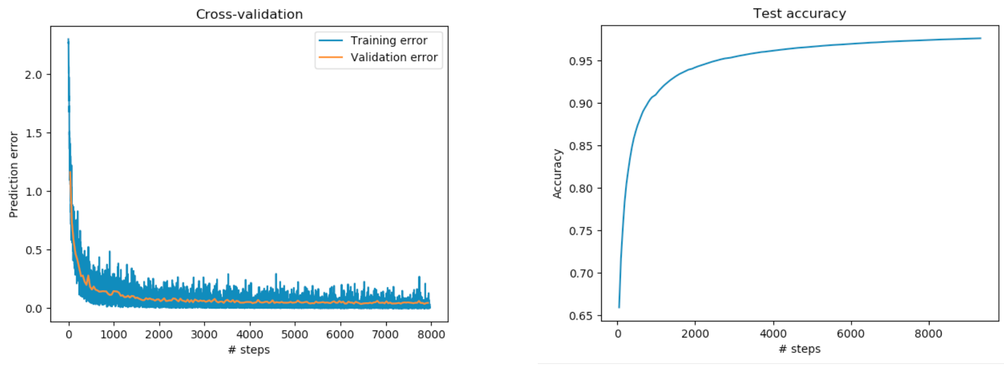


Figure 2: Plots showing the convergence of training and validation errors (right) and the accuracy on test set (left) as training progresses. The plots correspond to training of a basic LSTM model with single hidden layer containing 256 units

- **Bidirectional RNN**
  Average prediction accuracy on test set = **98.36 %**

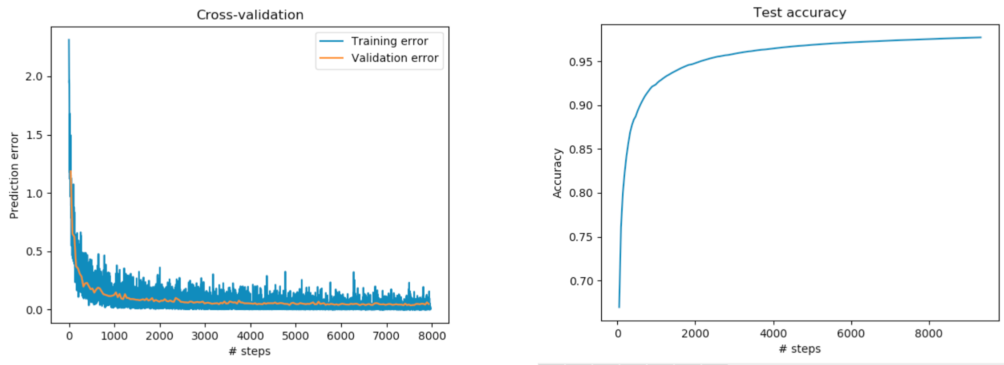


Figure 3: Plots showing the convergence of training and validation errors (right) and the accuracy on test set (left) as training progresses. The plots correspond to training of a basic LSTM model with single hidden layer containing 256 units

1.3 Predictions on sample images

- **LSTM**

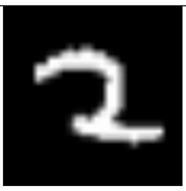
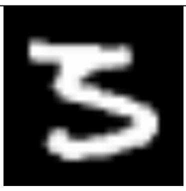
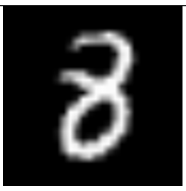
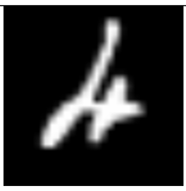
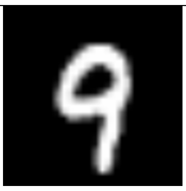
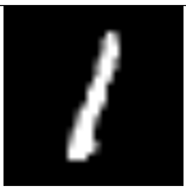
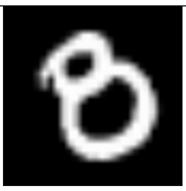
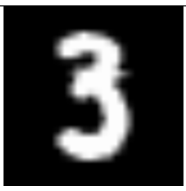
	**True label: 2** **Predicted label: 2**		**True label: 3** **Predicted label: 5**
	**True label: 8** **Predicted label: 3**		**True label: 4** **Predicted label: 2**
	**True label: 9** **Predicted label: 9**		**True label: 1** **Predicted label: 1**
	**True label: 8** **Predicted label: 8**		**True label: 3** **Predicted label: 3**

Table 4: Table showing the true labels and labels predicted by the LSTM model. The images are randomly chosen from test dataset

- **RNN**

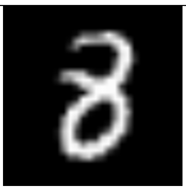
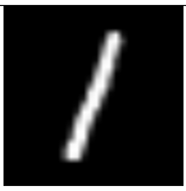
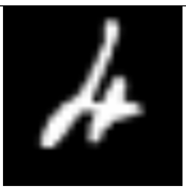
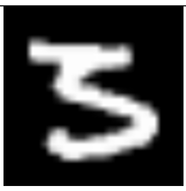
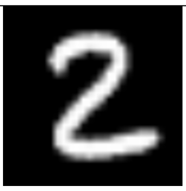
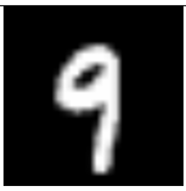
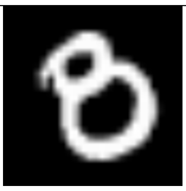
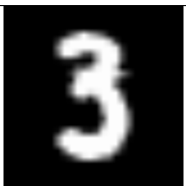
	**True label: 8** **Predicted label: 8**		**True label: 1** **Predicted label: 1**
	**True label: 4** **Predicted label: 6**		**True label: 3** **Predicted label: 5**
	**True label: 5** **Predicted label: 5**		**True label: 9** **Predicted label: 9**
	**True label: 8** **Predicted label: 8**		**True label: 3** **Predicted label: 3**

Table 5: Table showing the true labels and labels predicted by the RNN model. The images are randomly chosen from test dataset

- **Bidirectional RNN**

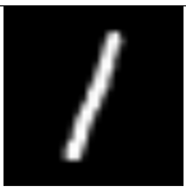
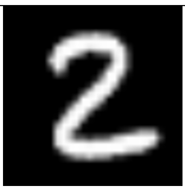
	True label: 8 **Predicted label: 2**		**True label: 1** **Predicted label: 1**
	True label: 4 **Predicted label: 6**		**True label: 3** **Predicted label: 5**
	**True label: 5** **Predicted label: 5**		**True label: 9** **Predicted label: 9**
	True label: 8 **Predicted label: 0**		**True label: 3** **Predicted label: 3**

Table 6: Table showing the true labels and labels predicted by the RNN model. The images are randomly chosen from test dataset

2 Sequence modelling

LSTM models are more robust to remembering the number at a particular index than Vanilla RNN models. This can be expected because LSTM's are variants of RNN designed specially to selectively remember the past. The Vanilla RNN gives average prediction accuracy in test images in the range **35% - 40%**, for 10 units in hidden layer and after 10k iterations. Whereas LSTM's give test prediction accuracy of **97% - 99%** in the same settings. Hence LSTM were used for all the experiments carried out in this section

2.1 Training

For each of the experiments, the models were trained on the data generated on the fly. Each model variant was trained for 10k iterations with batch size 64 and learning rate 0.01. The test data was a set of 10k images generated beforehand.

- **10 unit hidden layer**
  Average prediction accuracy is **98.40%**

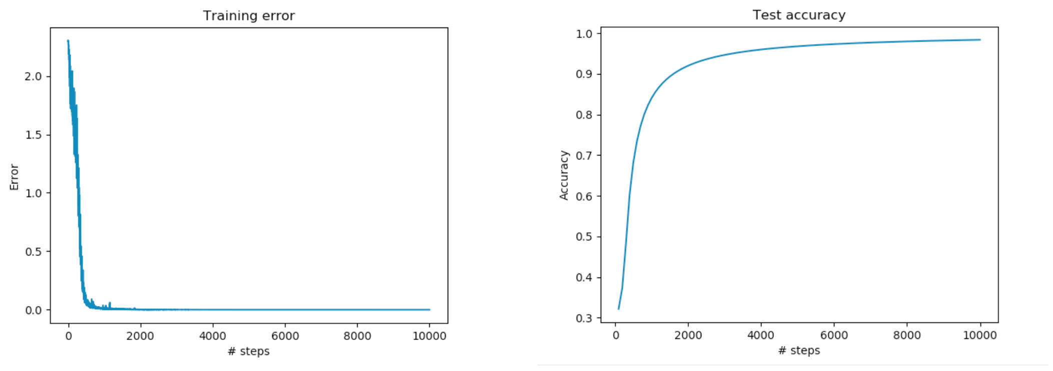


Figure 4: The plots show the variation of training error (left plot) and prediction accuracy on generated test data (right) as training progresses. These plots correspond to LSTM models with 10 unit hidden layer

- **5 unit hidden layer**
  Average prediction accuracy is **89.42%**

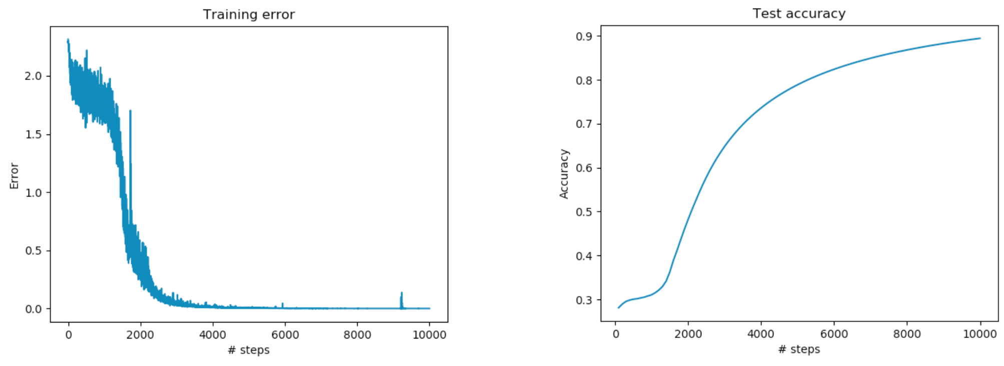


Figure 5: The plots show the variation of training error (left plot) and prediction accuracy on generated test data (right) as training progresses. These plots correspond to LSTM models with 5 unit hidden layer

- **2 unit hidden layer**
  Average prediction accuracy is **25.65%**

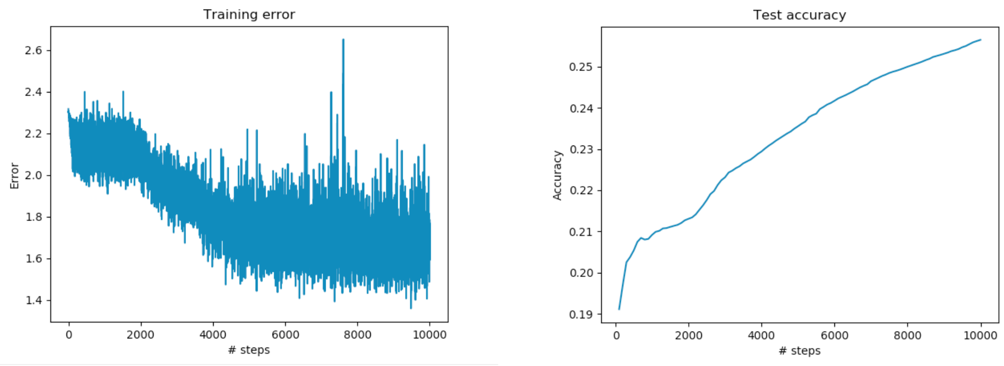


Figure 6: The plots show the variation of training error (left plot) and prediction accuracy on generated test data (right) as training progresses. These plots correspond to LSTM models with 2 unit hidden layer

The performance of LSTM models, after training for a fixed number of iterations, degrades if the size of hidden layer is reduced. However we cannot conclude that the model can never learn to remember the k^{th} number in the sequence. We can see from the test accuracy plots that the learning has not reached its asymptotic limit. So we can say with fair confidence that the models with lesser number of neurons in hidden layer can learn to decently represent the given data if it could be trained for more iterations.

2.2 Results of experiments by varying k

The LSTM model with 10 unit hidden layer is chosen as the best model. The performance of the model after training it for recollecting different positions (k) of input sequence is shown below.

- **k = 3**
  Average prediction accuracy of test sequences: **98.14%**

Sequence	[1,4,**1**]	[2,9,**5**,0,9,6,9,3]	[9,1,**6**,6,2,1,6]	[5,7,**2**,3,9,5]	[9,6,**7**,4,5,2,2,7,1]
Predicted output	1	5	6	2	7

Table 7: Prediction on test sequence samples

- **k = 4**
  Average prediction accuracy of test sequences: **98.21%**

Sequence	[3,6,0,**9**,4,6,1,8,2]	[3,6,5,**9**,1,6,8,8,8]	[6,2,4,**9**,0,1]	[9,8,6,**8**,5,1,8,9]	[9,5,7,**9**,5,6,8]
Predicted output	9	9	9	8	9

Table 8: Prediction on test sequence samples

- $k = \mathbf{5}$
  Average prediction accuracy of test sequences: **98.74%**

Sequence	[4,9,0,7,**2**,7]	[5,9,3,5,**7**,9,7,4]	[5,8,5,8,**0**,0,2,1]	[7,7,0,0,**4**,1,3,8,5]	[4,8,4,4,**2**,8,7,3,3]
Predicted output	2	7	0	4	2

Table 9: Prediction on test sequence samples

- $k = \mathbf{6}$
  Average prediction accuracy of test sequences: **98.62%**

Sequence	[2,3,8,6,4,**7**,0,4]	[5,4,6,5,5,**0**,6,5]	[2,5,5,2,8,**5**,1]	[8,9,4,4,4,**3**]	[7,6,2,8,9,**2**]
Predicted output	7	0	5	3	2

Table 10: Prediction on test sequence samples

- $k = \mathbf{7}$
  Average prediction accuracy of test sequences: **98.38%**

Sequence	[1,3,3,0,6,8,**7**,1,2]	[8,7,9,3,2,9,**6**,4]	[0,6,6,0,7,3,**7**,4]	[0,4,9,6,4,8,**8**,9]	[4,2,6,2,4,4,**1**]
Predicted output	7	6	7	8	1

Table 11: Prediction on test sequence samples

- $k = \mathbf{8}$
  Average prediction accuracy of test sequences: **96.73%**

Sequence	[6,0,8,2,9,8,8,**2**,6]	[4,6,3,5,6,4,0,**1**]	[5,9,4,1,5,8,9,**6**]	[9,2,8,4,4,6,2,**2**]	[6,1,1,1,2,7,7,**7**,4]
Predicted output	2	1	6	2	7

Table 12: Prediction on test sequence samples

- $k = \mathbf{9}$
  Average prediction accuracy of test sequences: **100.00%**

Sequence	[0,5,1,0,8,5,5,3, **6**]	[2,0,0,6,1,5,4,0,**1**]	[4,3,9,4,8,3,7,0,**9**]	[4,9,8,6,3,5,2,3,**7**]	[5,6,4,1,8,4,6,2,**4**]
Predicted output	6	1	9	7	4

Table 13: Prediction on test sequence samples

3 Adding two binary numbers

The training data was generated on the fly with a fixed input size L. The size of each data fed into the network and the output generated had size $L + 1$ to account for the carry bit.

3.1 Experiments

By default, the cross-entropy cost function was used in all the experiments unless mentioned otherwise.

1. **Hidden layer sizes**
   Models with different sizes of hidden state vectors were trained and the results of training are summarized below. Each model is trained for a fixed number ($10k$) iterations.

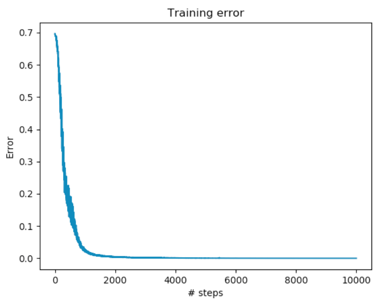
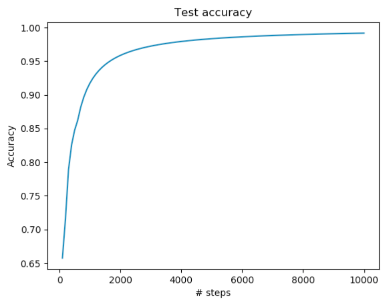
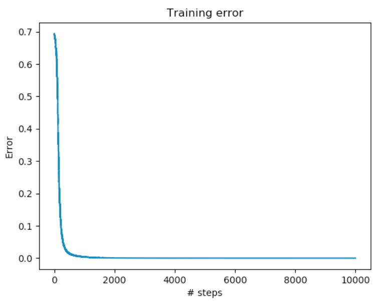
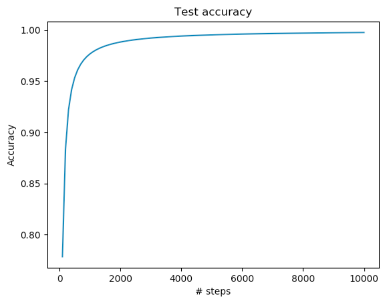
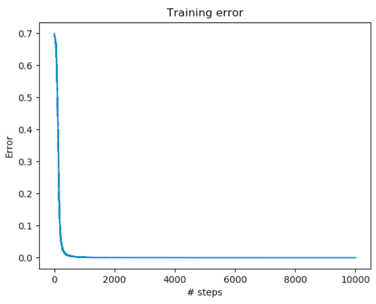
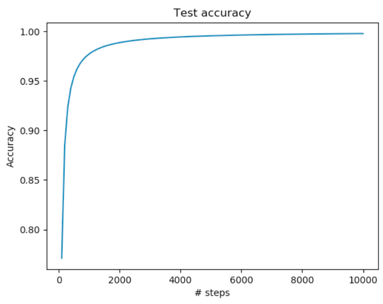
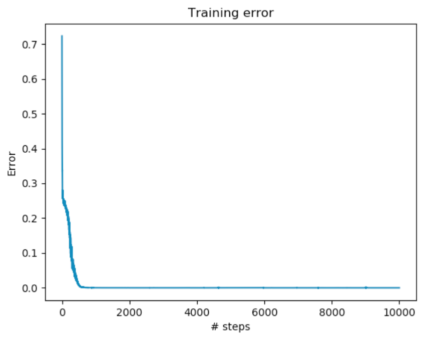
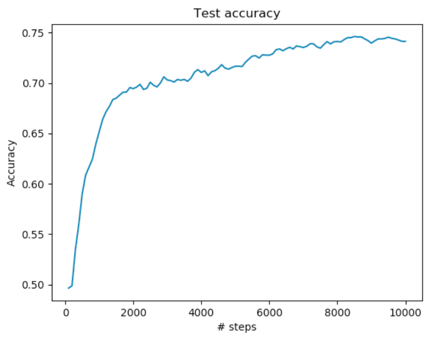
		Performance plots corresponding to models with size of hidden state vector being 3. The prediction accuracy on test data: **99.17%**
		Performance plots corresponding to models with size of hidden state vector being 5. The prediction accuracy on test data: **99.77%**
		Performance plots corresponding to models with size of hidden state vector being 10. The prediction accuracy on test data: **99.77%**

Table 14: Table showing the training error (left) and test accuracy (right) plots as training progresses. The accuracy computed here is bit accuracy - the fraction of bits correctly predicted

We can see that the performance doesn't change much when the hidden state vector size is changed. Hence among the models that give best performances, a model that uses least number of parameters was chosen. Hence the model with 5 unit hidden layer is chosen for further experiments.

2. **Cost function**
The performance of models trained with cross-entropy loss and models trained with MSE loss are shown below





The plots show the variation of training error (left plot) and prediction accuracy on generated test data (right) as training progresses. These plots correspond to LSTM models with 5 units hidden layer, trained with MSE loss

Models trained with cross-entropy loss performed much better than the models trained with MSE loss. Prediction bit-accuracy with cross-entropy loss: **99.77%**; Prediction bit-accuracy with MSE loss: **74.14%**

3. **testing model with inputs of different sizes**
The performance of all models, trained with a particular length, on different input sequences in summarized below

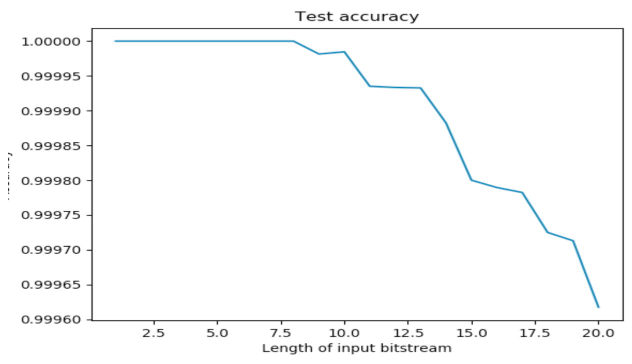
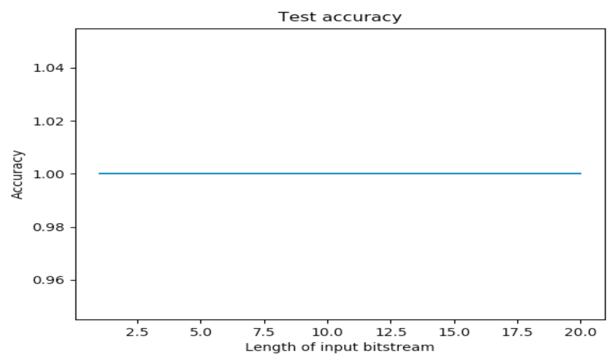
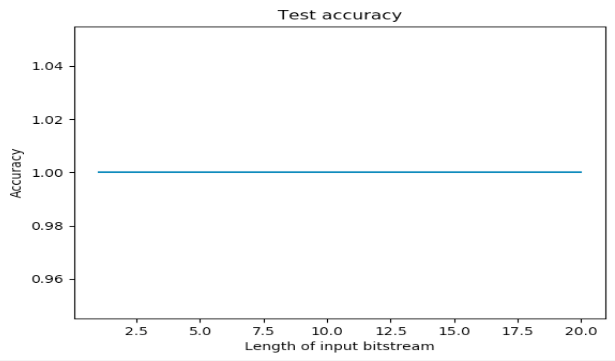
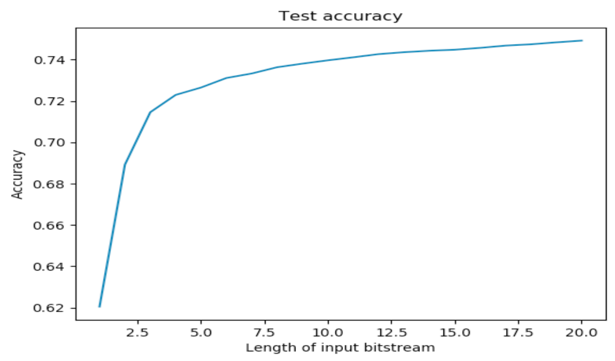
	Plot showing the variation of average bit-accuracy of predictions made by a model trained with **3-length** input sequences and **cross-entropy** loss, with different input sequence lengths during test time
	Plot showing the variation of average bit-accuracy of predictions made by a model trained with **5-length** input sequences and **cross-entropy** loss, with different input sequence lengths during test time
	Plot showing the variation of average bit-accuracy of predictions made by a model trained with **9-length** input sequences and **cross-entropy** loss, with different input sequence lengths during test time
	Plot showing the variation of average bit-accuracy of predictions made by a model trained with **5-length** input sequences and **Mean Squared Error** loss, with different input sequence lengths during test time

Table 15: Plots showing variation of prediction bit-accuracy with length of input sequences

We can observe that the models trained with larger input lengths are more robust than those trained with smaller input lengths.