Deep Learning – Adding Integers with Recurrent Networks

Qing Zhou, s2501597

1 Introduction

The purpose of this project is to implement different Recurrent Neural Networks (RNNs) on the addition of integers, to understand how the networks differ in performance, and what are the relationships between the input elements and network architecture. Three RNNs will be implemented in this project, Simple RNN, Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU). Additionally, to explore the input-output relation, the following input format are experimented:

1. "plain": the addition of integers "123+345" is read from left to right;
2. "reversed": the addition is read reversely as "543+321";
3. "binary plain": the integers are first converted into binaries, then read from left to right as usual;
4. "binary reversed": the integers are first converted into binaries, then read reversely.

Moreover, for each input format listed above, we have an extra option of pairing the inputs, for example, the input "123+345" is paired as "(1,3),(2,4),(3,5)" and then read regularly. Hence we have altogether 8 ways of different input formats. In the following experiments, I have used 50000 samples for training the network, and 5000 for validation. The training is set to run for 200 iterations by default. All the network is set to have one hidden layer with 128 neurons, and the batch size is set to be 128 as well, a reasonable choose given a training size of 50000. To see the training results, the loss of the validation set is plotted against iteration epoch, as well as the accuracy of the validation set as a function of iteration epoch. The skeleton code for all the experiments can be found here. In order to see the differences in the performance of the networks in each case more clearly, the errors are extended to the whole set (1 million in case of decimal representations, and 2^{20} in case of binary representations). The percentage of correctly identified inputs is calculated, as well as the Mean Absolute Error (MAE) and the Mean Squared Error (MSE). The MAE and MSE are computed as

$$\begin{aligned} MAE &= \frac{1}{N}\sum_{N} |x_{\text{correct}} - x_{\text{predicted}}| \\ MSE &= \frac{1}{N}\sum_{N} (x_{\text{correct}} - x_{\text{predicted}})^2 \end{aligned} \tag{1}$$

2 Task 1

In this task we focus on exploring the relation between the "plain" input (also its reversed form) and the three different network architectures. The results of the training are presented in Fig. 1. It can be seen, for both LSTM and GRU, the accuracy on the validation set is able to reach $> 99\%$, and stay well above it, for both reversed and plain inputs. As for the Simple RNN, the accuracy is not

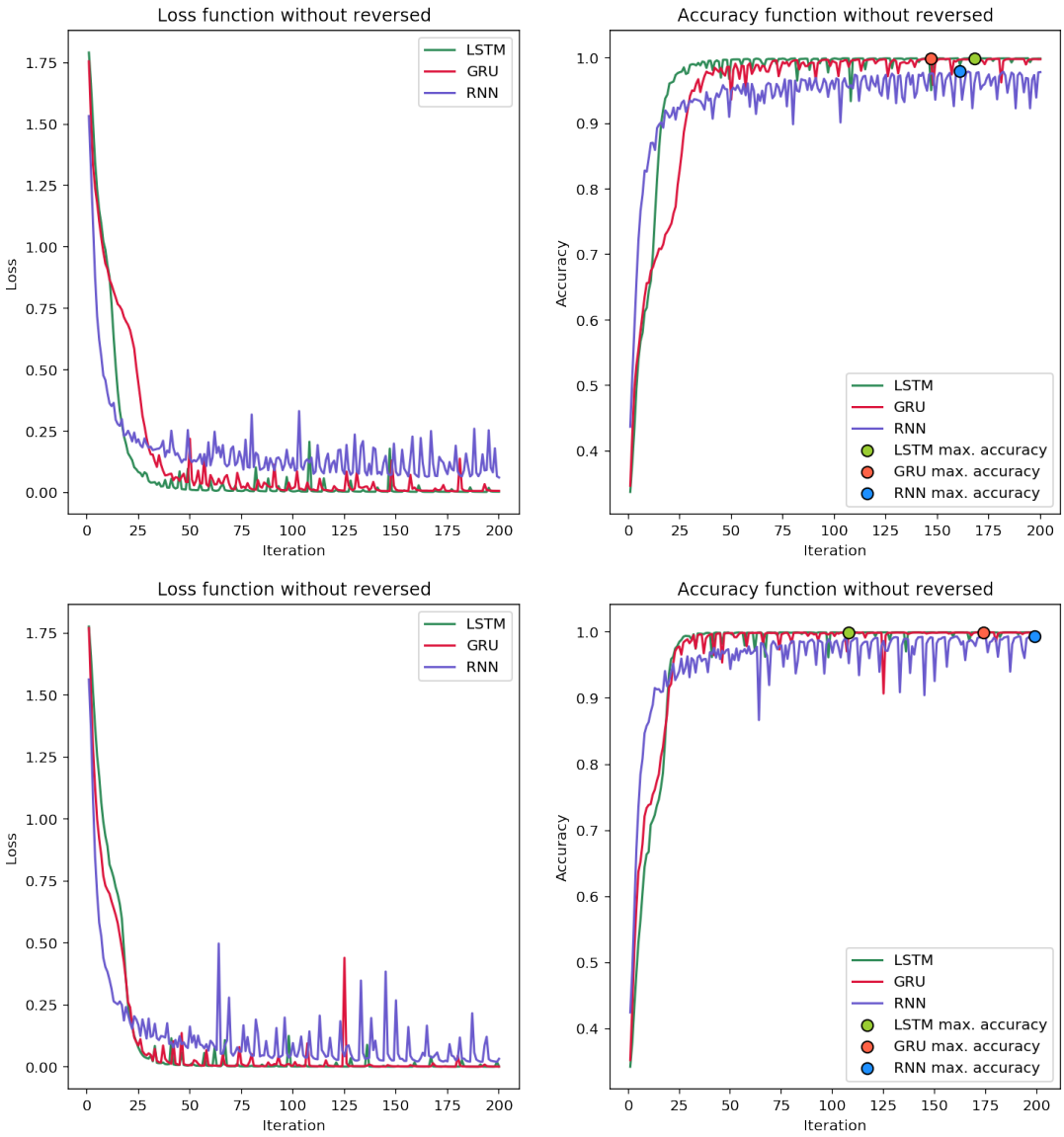

Figure 1: Loss function and accuracy function on the validation set, for all three networks (green: LSTM, red: GRU, blue: RNN), with(bottom panel) and without (top panel) reversed. The marker gives the position at which the maximum accuracy is achieved.

as good as the previous ones. The highest accuracy it can get is 97.9% for the plain input case, and 99.3% for the reversed case. To see their differences more clearly, the extended errors are listed in Table 1.

It can be seen, all the networks have better performance when the input is reversed, amongst which GRU has the highest accuracy on the total set, 99.76%, and with lowest MAE and MSE in the case of reversed input. On the other hand, when the input is not reversed, LSTM is the network with best performance, with an accuracy of 99.0% on the total set.

3 Task 2

In this task, the relation between the network architectures and the binary input is explored. The input integers are first converted into binaries of maximum length 10, if the converted binary is shorter than the maximum length, then spaces are padded at the front of the converted integers.

Table 1: Extended Error on the whole data set of 1 million inputs.

Item	REVERSE	LSTM	GRU	SimpleRNN
Accuracy	False	99.00%	98.58%	87.86%
	True	99.68%	99.76%	95.90%
MAE	False	4.89	3.93	14.24
	True	1.21	0.183	0.959
MSE	False	3.30×10^4	2.52×10^4	7.07×10^4
	True	8.1×10^3	8.97×10^2	2.64×10^4

Then the converted binary strings are added with "+". The training results are presented in Fig. 2. All the networks are able to reach 99.6% with binary input with ease, and quite remarkably, for the

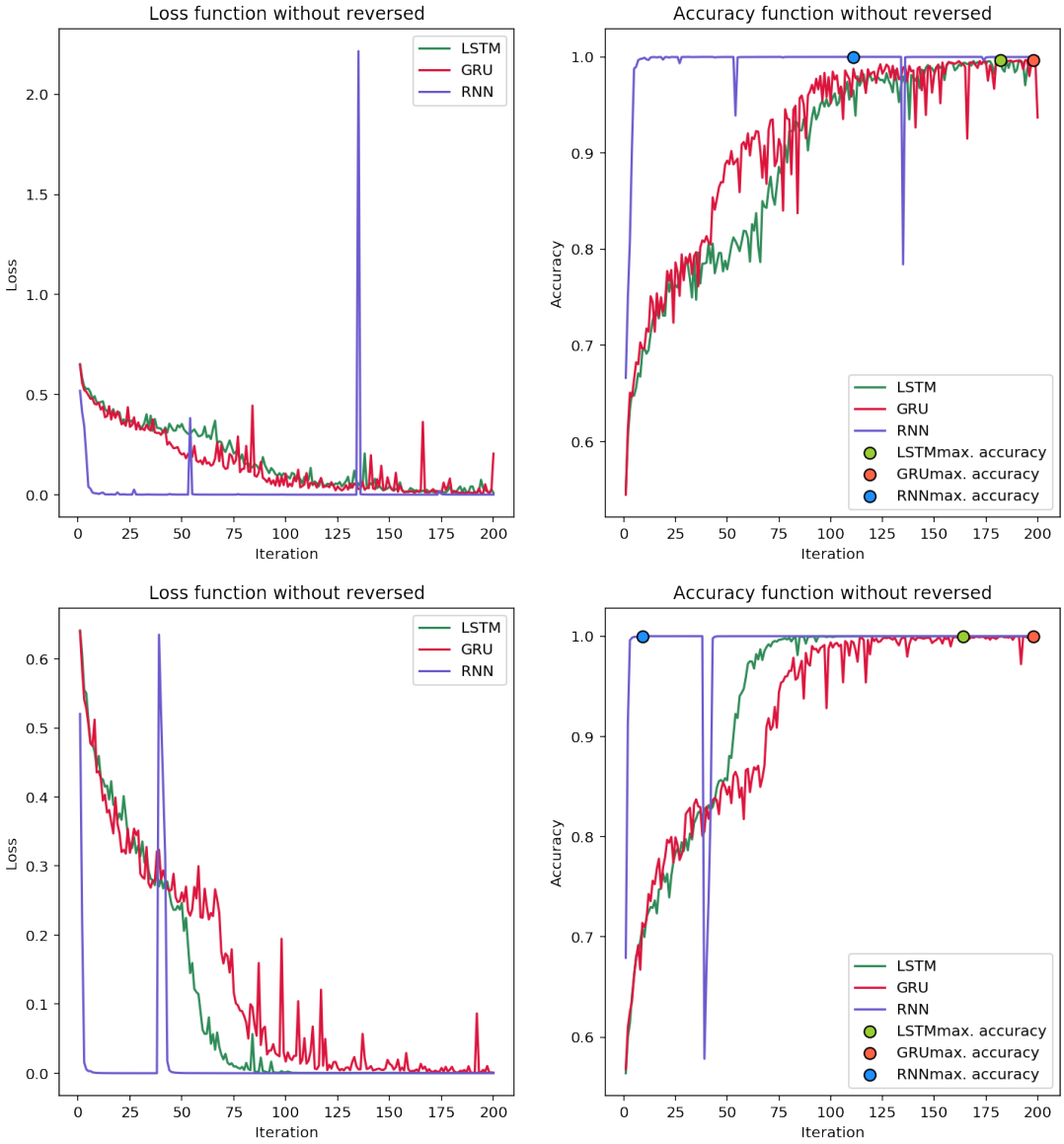

Figure 2: Loss function and accuracy function on the validation set, for all three networks (green: LSTM, red: GRU, blue: RNN), with(bottom panel) and without (top panel) reversed. The marker gives the position at which the maximum accuracy is achieved.

SimpleRNN network, the training was able to reach 100% for binary input. As for the reversed binary input, both LSTM and GRU are able to reach 99.99%, and for the SimpleRNN, it is able to reach 100% and stays there steadily. The extended error measures after 200 iterations of training are listed in Table 2. In overall, when the input is reversed, all network has much higher accuracy. For both

Table 2: Extended Error on the whole data set of 2^{20} inputs.

Item	REVERSE	LSTM	GRU	SimpleRNN
Accuracy	False	73.94%	29.70%	98.78%
	True	98.80%	98.37%	100%
MAE	False	1.265×10^{7}	7.98×10^{7}	2.87×10^{6}
	True	2.8×10^{4}	2.2×10^{5}	0
MSE	False	5.23×10^{16}	5.61×10^{17}	3.13×10^{15}
	True	1.60×10^{13}	9.14×10^{14}	0

plain binary and reversed binary, the SimpleRNN is able to deliver best performance amongst the three network. Apart from the SimpleRNN, LSTM outperforms GRU with both formats of input. Compared to Fig. 2, the loss of SimpleRNN drops most quickly, reaches $\ll 0.1$ in a few iterations. However, there is a very sharp spike at ~ 40, probably because the potential has climbed out of the local minimum. Yet afterwards the accuracy is able to stay steadily at 100%.

4 Task 3

In this task, the relation between the network architectures and the paired plain input is explored. The input integers 123+345 are first paired as $(1,3),(2,4),(3,5)$ ($(3,5),(2,4),(1,3)$ if reversed), then the encoded paired inputs are feeded into the neural network. The training results are presented in 3. When the input is not reversed, LSTM can achieve quite remarkable result. The highest accuracy on the validation set is 99.995%, and it can well maintain this accuracy for $> 99.9\%$. The second best network is the SimpleRNN, for it can reach a maximum accuracy of 99.97%, and maintains it well above 99.9%. Lastly, GRU is able to achieve a maximum accuracy of 99.1%, but its performance has a larger fluctuation, and it can maintain the accuracy for $> 97.9\%$. When the input is reversed, however, the best network is LSTM, for it can reach a maximum accuracy of 100% and well maintain it. The GRU can also reach a maximum accuracy of 100%, and maintain it for $> 99.99\%$. Even with the SimpleRNN, it is able to reach a maximum accuracy of 099.99%, and maintain it $> 99.9\%$. To see the differences between the networks we refer Table 3 It can be seen, in both cases LSTM is able

Table 3: Extended Error on the whole data set of 1 million inputs.

Item	Paired	LSTM	GRU	SimpleRNN
Accuracy	False	99.39%	77.20%	99.59%
	True	99.98%	80.62%	99.09%
MAE	False	0.518	3.585	0.354
	True	0.054	0.015	0.441
MSE	False	2.14×10^{3}	2.99×10^{3}	3.46×10^{2}
	True	3.21×10^{2}	65.6	1.31×10^{2}

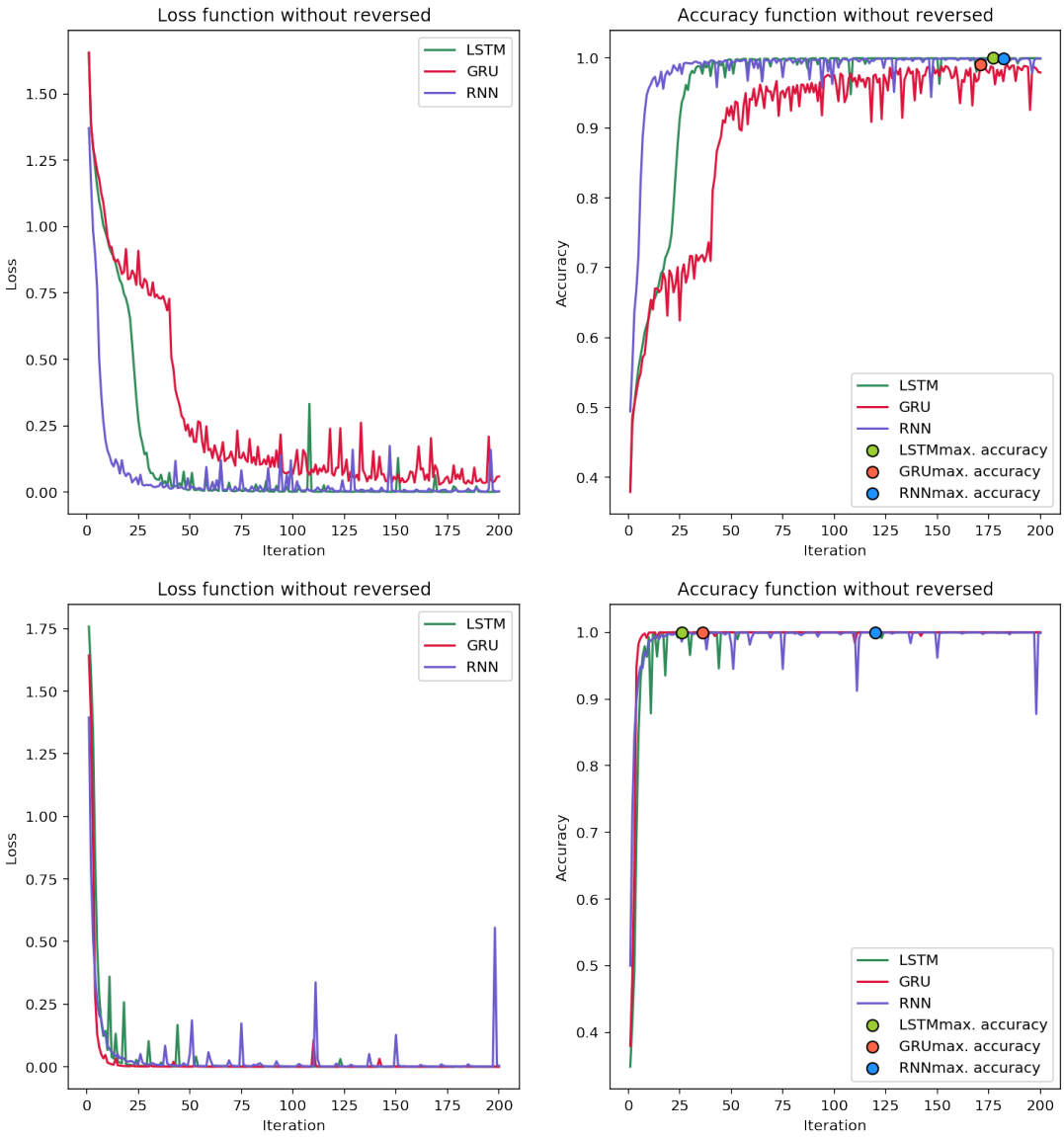

Figure 3: Loss function and accuracy function on the validation set, for all three networks (green: LSTM, red: GRU, blue: RNN), with(bottom panel) and without (top panel) reversed. The marker gives the position at which the maximum accuracy is achieved.

to deliver quite stable result, and when the input is reversed, the accuracy on the whole data set increased from 99.39% to 99.98%. A SimpleRNN also has good performance in both cases. However, for the GRU, in both cases the accuracy is quite low. When the input is reversed, the accuracy only increased from 77.20% to 80.62%.

5 Task 4

In the last task, the relation between the paired binary input and the network architectures is explored. The input integers are of maximum length 10. When the input string has shorter length, empty spaces were added in the front. The two strings of integers for addition are later added with a "+" sign, and the encoded inputs are then fed into the neural network for training. The training results are shown in Fig. 4. It can be seen, when the input is formatted as plain paired binaries, the SimpleRNN has the best performance, reaching the highest accuracy of $\sim$ 99.999% after $\sim$ 20

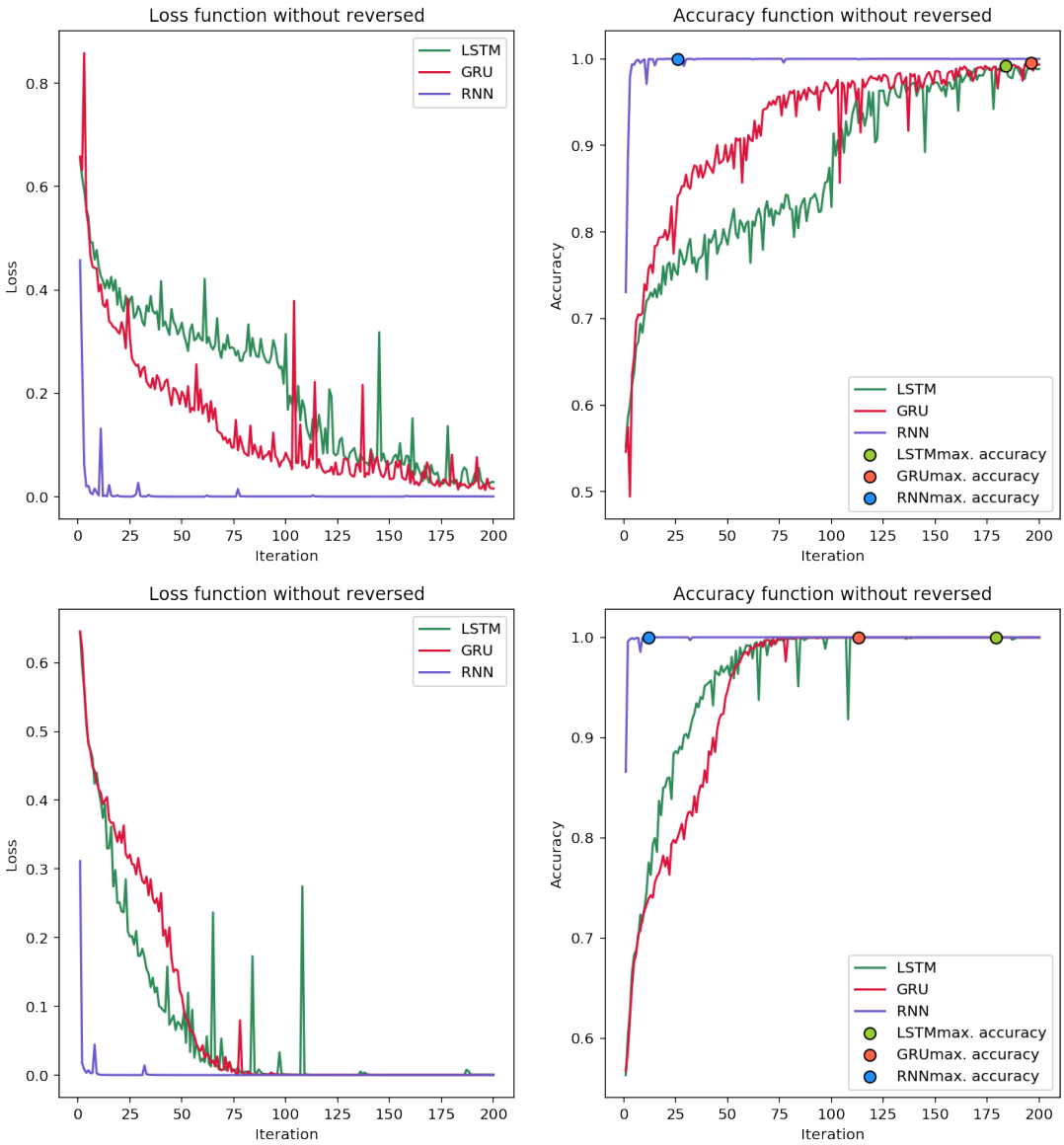

Figure 4: Loss function and accuracy function on the validation set, for all three networks (green: LSTM, red: GRU, blue: RNN), with(bottom panel) and without (top panel) reversed. The marker gives the position at which the maximum accuracy is achieved.

iterations. The accuracy is able to be kept for $> 99.99\%$ thereafter, of minuscule fluctuations. The second best is the GRU, reaching a maximum accuracy of 99.5% at the very end of the training. LSTM however, has a maximum accuracy of 99.1%, with 0.3% fluctuations. On the other hand, when the input is reversed, both GRU and LSTM are able to deliver a maximum accuracy of $> 99.99\%$, after some 75 iterations. The SimpleRNN again, is the best performed network, reaching 100% accuracy and maintains it after ~ 10 iterations. The extended errors of the three networks can be seen in Table 4. In the case where the input is paired binaries, in overall, the SimpleRNN has the best performance, with 99.30% accuracy when the input is not reversed and 99.997% when the input is reversed. In other networks, when the input is reversed both networks have improved significantly, with accuracy almost doubled.

Table 4: Extended Error on the whole data set of 1 million inputs.

Item	Paired	LSTM	GRU	SimpleRNN
Accuracy	False	57.29%	61.84%	99.30%
	True	98.99%	99.40%	99.997%
MAE	False	2.68×10^7	5.11×10^7	1.06×10^6
	True	6.04×10^5	2.03×10^5	0.0007
MSE	False	1.77×10^{17}	3.26×10^{17}	2.10×10^{15}
	True	3.50×10^{15}	1.77×10^{15}	0.05

6 Conclusion

In conclusion, we have seen an advantage in LSTM when the input is relatively shorter (see Task 1 and Task 3). When the input gets more complicated (see Task 2 and Task 4), a SimpleRNN is a better choice to handle the problem with steady and reliable solution. It is also found that, in all cases when the input is reversed, the performance of the network is also improved. The cause for the two phenomena is probably the introduction of many short term dependencies of the dataset. Normally when we concatenate a word with a target, each letter in the word is far from its corresponding letter in the target word. As a result, there is a "minimal time lag". By reversing the letters in the source word, the average distance between the corresponding letters in the source and target is unchanged. However, the first few letters in the source word are now very close to the first few letters in the target word, so the problem of the minimal time lag is greatly suppressed. In this way, backpropagation would have an easier time establishing communication between the source and the target, which in turn results in greatly improved performance of the neural network.

References

[1] Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. "Sequence to sequence learning with neural networks." Advances in neural information processing systems. 2014.