# HW3: https://github.com/RHBadhon/ECGR\_5106/tree/main/HW3

Rashadul Hasan Badhon

801341298

**All figures are added as supplementary materials at the end.**

Problem 1: Comparison among RNN, LSTM and GRU

Used sequence: 10, 20, 30.

Figures are added as supplementary materials at the end.

The following table shows a comparison of accuracy among different models for different lengths of sequence of text learning.

|  |  |  |  |
| --- | --- | --- | --- |
| **Sequence length** | **RNN**  (Training loss,  Val. Accuracy,  Execution time,  Model complexity) | **LSTM**  (Training loss,  Val. Accuracy,  Execution time,  Model complexity) | **GRU**  (Training loss,  Val. Accuracy,  Execution time,  Model complexity) |
| 10 | 0.095  0.515  2.49 sec  44332 | 0.11  0.48  4.995 sec  143404 | 0.074  0.53  9.13 sec  110380 |
|  |  |  |  |
| 20 | 0.11  0.502  7.08 sec  44332 | 0.10  0.504  15.07 sec  143404 | 0.057  0.53  21.15 sec  110380 |
|  |  |  |  |
| 30 | 0.092  0.50  22.31 sec  44332 | 0.14  0.51  24.99 sec  143404 | 0.054  0.496  33.01 sec  110380 |

In conclusion, GRU consistently outperforms RNN and LSTM in terms of training loss and validation accuracy across different sequence lengths, highlighting its efficiency in learning long-term dependencies. However, this comes at the cost of increased execution times, suggesting a trade-off between performance and computational efficiency. LSTM, despite its high model complexity, does not consistently yield the best performance, indicating that higher complexity does not necessarily translate to better predictive capability for these sequence lengths. RNN, while the fastest and least complex, tends to underperform in accuracy metrics, emphasizing the importance of advanced recurrent units like GRU for tasks requiring the capture of long-term dependencies in data.

Problem 2: Comparison between LSTM and GRU for the tiny Shakespeare dataset.

1. Comparison for a sequence of 20 and 30

|  |  |  |
| --- | --- | --- |
| **Sequence length** | **LSTM**  (Training loss,  Val. Accuracy,  Execution time,  Model complexity) | **GRU**  (Training loss,  Val. Accuracy,  Execution time,  Model complexity) |
| 20 | 1.27  0.525  48.64 min  148801 | 1.68  0.44  46.11 min  115777 |
|  |  |  |
| 30 | 2.46  0.59  50.80 min  148801 | 2.44  0.48  50.13 min  115777 |

Here, the LSTM model demonstrates superior validation accuracy in both sequence lengths, suggesting its effectiveness in capturing dependencies in longer sequences, despite having a higher training loss and slightly longer execution time for the sequence length of 30. The increased complexity of LSTM, with more parameters than GRU, seems to afford it a slight advantage in performance, although at the cost of computational resources and time. GRU, while more efficient in terms of execution time, particularly for the sequence length of 20, tends to lag behind LSTM in achieving high validation accuracy, especially as sequence length increases. This indicates a trade-off between model complexity, execution time, and predictive performance when choosing between LSTM and GRU for processing sequences of varying lengths.

1. Adjusting various hyperparameters

I tried different combinations of hyperparameter tuning to observe performance for both LSTM and GRU models.

**LSTM**

|  |  |
| --- | --- |
| Hyperparameter change | (Training loss,  Val. Accuracy,  Execution time,  Model complexity) |
| 2 hidden layers, 1 fc layer, sequence length 20, hidden state 128 | 2.89  0.23  2998.3 sec  280897 |
| 2 hidden layers, 2 fc layers, sequence length 20, hidden state 128 | 1.37  0.47  3341.82 sec  346817 |
| 2 hidden layers, 2 fc layers, sequence length 20, hidden state 256 | 1.5  0.48  5202.9 sec  1348929 |
| 2 hidden layers, 2 fc layers, sequence length 30, hidden state 256 | 0.83  0.55  6358.6 sec  1348929 |
| 2 hidden layers, 2 fc layers, sequence length 10, hidden state 256 | 1.3  0.58  4307.09  1348929 |

From this table we get a clear indication that increasing hidden states or hidden layers increases the model complexities as well as execution time. Although the model complexity increases with the increasing execution time, the performance increases as well.

**GRU**

|  |  |
| --- | --- |
| Hyperparameter change | (Training loss,  Val. Accuracy,  Execution time,  Model complexity) |
| 2 hidden layers, 2 fc layers, sequence length 20, hidden state 256 | 1.93  0.52  5098.10 sec  1085761 |
| 2 hidden layers, 2 fc layers, sequence length 30, hidden state 256 | 2.36  0.36  6348.54 sec  1085761 |

GRU shows a similar trend of having higher accuracy for 20 sequence length containing multiple hidden and fc layers.

1. Comparison for a sequence of 50

|  |  |  |
| --- | --- | --- |
| **Sequence length** | **LSTM**  (Training loss,  Val. Accuracy,  Execution time,  Model complexity) | **GRU**  (Training loss,  Val. Accuracy,  Execution time,  Model complexity) |
| 50 | 1.32  0.53  49.87 min  148801 | 1.87  0.42  50.48 min  115777 |

It is noticeable that for higher sequence length, LSTM performs better than GRU following a similar trend for 20 and 30 sequence length. Although comparably they take more time and accuracy drops in the process.

# Figures

A screenshot of a computer

Description automatically generated

(a)

A screenshot of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated(b)

(c)

A screenshot of a computer code

Description automatically generated

A screenshot of a computer code

Description automatically generated(d)

(e)

A screenshot of a computer

Description automatically generated

(f)

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated(g)

(h)

A screenshot of a computer code

Description automatically generated

(i)

Fig 1: (a-c) RNN, (d-f) LSTM, (g-i) GRU (Problem 1)

|  |
| --- |
| (a) |
| (b) |
| (c) |
| (d) |
| (e) |
| (f) |

Fig. 2: (a-c) LSTM, (d-f) GRU (Problem 2)

|  |
| --- |
| (a) |
| (b) |
| (c) |
| (d) |
| (e) |
| (f) |
| (g) |

Fig. 3: Hyperparameter tuning (a-e) LSTM, (f,g) GRU (Problem 2)