Comparative Analysis of RNN, LSTM, and GRU on Synthetic Sequence Tasks

Sekenova Balym, 23B031433

1 Introduction

Recurrent Neural Networks (RNNs) are widely used for modeling sequential data due to their ability to maintain a hidden state across time steps. However, vanilla RNNs are known to suffer from the vanishing gradient problem, which severely limits their ability to capture long-range temporal dependencies during backpropagation through time (BPTT).

To address this limitation, gated architectures such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) were proposed. These models introduce gating mechanisms that regulate information flow and help preserve gradients over extended time horizons.

The objective of this study is to conduct a controlled empirical comparison of RNN, LSTM, and GRU architectures on a synthetic sequence task, systematically varying sequence length in order to evaluate performance degradation, convergence behavior, and computational cost.

2 Hypotheses

- **H1:** LSTM and GRU outperform vanilla RNN on long sequences ($T > 50$).
- **H2:** GRU trains faster than LSTM while achieving comparable accuracy.
- **H3:** Vanilla RNN performs comparably to LSTM and GRU on short sequences ($T = 10$–20).

3 Experimental Setup

3.1 Dataset

We used the synthetic Adding Problem. Each sequence contains random values and a binary mask indicating two elements whose sum is the target output.

Training samples: 10,000
Test samples: 2,000

Sequence lengths evaluated:

$$T \in \{10, 25, 50, 100, 200, 500\}$$

3.2 Model Configuration

All models were trained under identical conditions:

- Hidden size: 64

- Number of layers: 1
- Optimizer: Adam (learning rate = 0.001)
- Batch size: 64
- Epochs: 50
- Three independent trials per configuration

4 Results

4.1 Parameter Count

Model	Number of Parameters
RNN	4417
LSTM	17473
GRU	13121

Table 1: Number of trainable parameters per model.

4.2 Performance vs Sequence Length

Sequence Length	RNN (MSE)	LSTM (MSE)	GRU (MSE)
10	0.000798 ± 0.000161	0.000072 ± 0.000018	0.000046 ± 0.000025
25	0.008954 ± 0.000297	0.000177 ± 0.000014	0.000206 ± 0.000047
50	0.090557 ± 0.067123	0.000249 ± 0.000136	0.000176 ± 0.000053
100	0.170629 ± 0.001892	0.000484 ± 0.000078	0.000382 ± 0.000155
200	0.163895 ± 0.000550	0.056098 ± 0.076302	0.000318 ± 0.000055
500	0.164240 ± 0.004415	0.164148 ± 0.003944	0.000363 ± 0.000069

Table 2: Test MSE (mean ± std over 3 trials).

4.3 Training Time per Epoch

Sequence Length	RNN (sec)	LSTM (sec)	GRU (sec)
10	0.366 ± 0.002	0.397 ± 0.001	0.383 ± 0.001
25	0.366 ± 0.003	0.397 ± 0.001	0.384 ± 0.000
50	0.368 ± 0.001	0.400 ± 0.001	0.387 ± 0.001
100	0.373 ± 0.001	0.403 ± 0.001	0.395 ± 0.001
200	0.399 ± 0.000	0.412 ± 0.001	0.408 ± 0.002
500	0.437 ± 0.001	0.585 ± 0.002	0.605 ± 0.001

Table 3: Average training time per epoch (mean ± std).

4.4 Loss Curves and Scaling Behavior

Figure 1 shows test MSE as a function of sequence length. Figure 2 illustrates training loss dynamics. Figure 3 presents training time scaling.

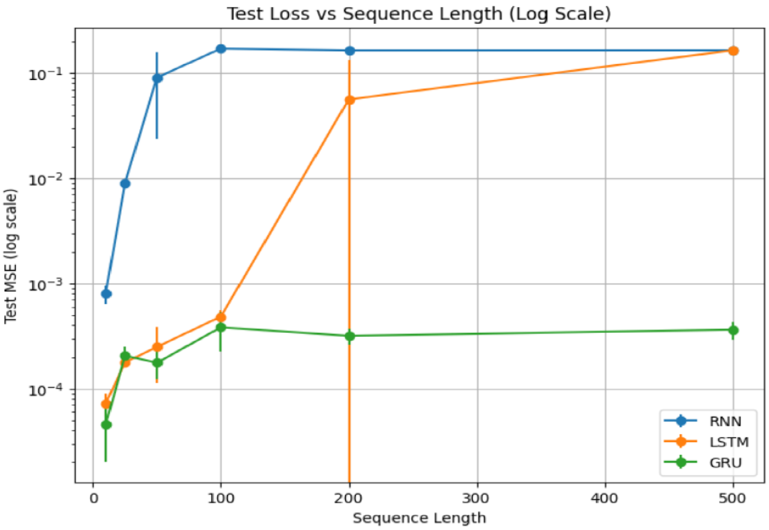

Figure 1: Test loss vs sequence length (log scale).

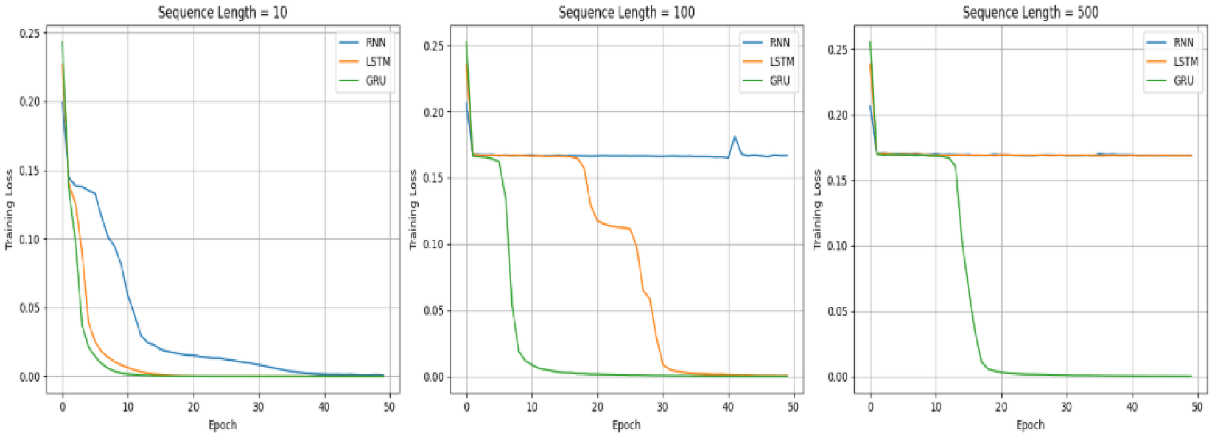

Figure 2: Training loss curves for selected sequence lengths.

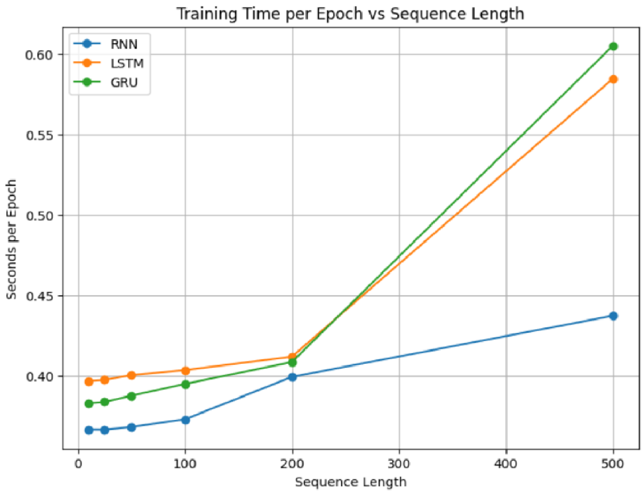

Figure 3: Training time per epoch vs sequence length.

5 Discussion

The experimental results provide strong empirical evidence supporting the theoretical limitations of vanilla RNNs. As sequence length increases beyond $T = 50$, the performance of the RNN deteriorates sharply. For example, at $T = 100$, the RNN achieves a test MSE of approximately 0.1706, whereas LSTM and GRU maintain errors below 5×10^{-4}. This substantial gap confirms the presence of the vanishing gradient problem, where gradients decay exponentially as they are propagated backward through long sequences.

LSTM demonstrates significantly improved stability compared to the vanilla RNN. Its gating mechanisms allow it to preserve information across moderate sequence lengths. However, at extremely long sequences ($T = 500$), LSTM performance degrades substantially (MSE ≈ 0.1641), indicating that even gated architectures can struggle when temporal dependencies become very long.

GRU exhibits the most robust behavior across all tested sequence lengths. Even at $T = 500$, GRU maintains a low test error (approximately 3.6×10^{-4}), indicating stable gradient propagation and effective long-range memory retention. The simpler gating structure of GRU may contribute to more stable optimization dynamics compared to LSTM.

From a computational perspective, RNN is consistently the fastest model due to its simpler architecture and lower parameter count. Both LSTM and GRU incur higher computational cost due to additional gating operations. While GRU is slightly faster than LSTM for shorter sequences, their computational cost becomes comparable for very long sequences.

Overall, the results align with theoretical expectations regarding gradient stability and confirm the practical advantages of gated recurrent architectures.

Hypothesis Evaluation

H1: Supported. LSTM and GRU significantly outperform vanilla RNN for sequence lengths greater than 50.

H2: Partially supported. GRU achieves comparable or superior accuracy relative to LSTM.

While computational differences are small, GRU demonstrates slightly better efficiency for short-to-medium sequences.

H3: Partially supported. Vanilla RNN performs comparably to gated architectures only for very short sequences ($T = 10$), but degrades rapidly as sequence length increases.

6 Conclusion

This study conducted a systematic empirical comparison of RNN, LSTM, and GRU architectures across varying sequence lengths. The results clearly demonstrate the limitations of vanilla RNNs in modeling long-range dependencies due to vanishing gradients.

LSTM significantly improves stability over moderate sequence lengths but exhibits degradation for extremely long sequences. GRU achieves the most consistent performance across all tested conditions and maintains low error even for $T = 500$, indicating superior robustness in this experimental setting.

These findings reinforce the importance of gating mechanisms in recurrent architectures and highlight GRU as an effective balance between model complexity, computational efficiency, and long-range dependency modeling.