Deep learning (INFOMDLR) - Assignment 1

Mahshid Jafar Tajrishi
Graduate School of Natural Sciences
Utrecht University
Netherlands
m.jafartajrishi@students.uu.nl

Bar Melinarskiy
Graduate School of Natural Sciences
Utrecht University
Netherlands
b.melinarskiy@students.uu.nl

Cis van Aken
Graduate School of Natural Sciences
Utrecht University
Netherlands
c.j.f.vanaken@students.uu.nl

Simon van Klompenburg
Graduate School of Natural Sciences
Utrecht University
Netherlands
a.s.vanklompenburg@students.uu.nl

***Abstract*—Time series forecasting of laser-based temporal measurement data presents significant challenges, as frequency and magnitude of oscillations vary over time. In this study, we develop and systematically evaluate three recurrent neural network architectures: RNN, LSTM, and GRU for the task of predicting future measurements. The primary objective was to identify the most suitable model architecture and configuration to accurately capture the temporal patterns in the data. To this end, we employed rigorous hyperparameter tuning using the Optuna framework, varying key parameters such as hidden size, number of layers, learning rate, batch size, and window size. The models were trained using the Adam optimizer, with MAE as the primary loss function to mitigate the impact of noise. Training was enhanced with techniques including rate scheduling and early stopping to avoid overfitting. Experimental results demonstrated that while the GRU model achieved superior performance on training and validation datasets, the LSTM model exhibited better generalization to unseen test data, effectively handling complex signal variations. Overall, the limited generalization ability observed across all three models indicates that more rigorous data preprocessing and systematic hyperparameter optimization are necessary to improve performance on this type of dataset.**

***Index Terms*—RNN, GRU, LSTM, Time series forecasting**

1. Introduction

Predicting future values in temporal measurement datasets is essential across domains like signal processing, forecasting, and automated control systems. Our study focuses on developing a neural network model to predict subsequent data points in a laser-based temporal measurement dataset. The challenge involves identifying the optimal architecture, whether RNN, LSTM, or GRU, and fine-tuning hyperparameters such as window size, layer count, and dropout rates. Using a real-world laser dataset, we will evaluate these deep learning approaches through MSE and MAE performance metrics and a visual comparison of the predictions.

2. Used Models

Time series forecasting involves capturing temporal dependencies and patterns across sequential data points, making Recurrent Neural Networks (RNNs) inherently suited for this task. RNNs maintain a hidden state that evolves over time, enabling the model to learn from previous time steps [1]. In this study, we explored three RNN architectures: Vanilla RNN, Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU) to predict one-step-ahead laser measurements. These models were chosen due to their proven ability to capture temporal dependencies in sequential data [2, 3, 4].

All three models share a similar architecture: a recurrent layer followed by a fully connected layer that outputs a single prediction for each input sequence. To ensure consistency across models, we used the following key parameters:

- ***input_size***: The number of features at each time step, which is set to 1 for this univariate time-series dataset.
- ***hidden_size***: The size of the hidden state, which determines the model's capacity to learn and represent patterns in the data.
- ***num_stacked_layers***: The number of recurrent layers stacked on top of each other. Increasing the depth allows the model to capture more complex temporal dependencies.

Below, we briefly describe each model and the rationale for its selection.

A. Recurrent Neural Networks (RNNs)

RNNs are a class of neural networks designed to process sequential data by maintaining a hidden state that evolves over time. However, vanilla RNNs suffer from vanishing gradient problems, which limit their ability to capture long-term dependencies [5, 6]. Despite this limitation, we included RNNs as a baseline model to compare their performance against more advanced architectures.

B. Long Short-Term Memory (LSTM)

LSTMs address the vanishing gradient problem by introducing memory cells and gating mechanisms that regulate the flow of information [1]. This makes LSTMs particularly effective for capturing long-term dependencies in time-series data. Given the sequential nature of the laser measurement dataset, LSTMs were a natural choice for this task.

C. Gated Recurrent Unit (GRU)

GRUs are a streamlined alternative to LSTMs that use gating mechanisms to control the flow of information without relying on a separate memory cell. They merge the functionalities of the input and forget gates into a single update gate, and use a reset gate to modulate the influence of previous hidden states [4]. This simplification reduces the model's computational overhead while preserving its ability to capture temporal dependencies. GRUs were included to assess whether this balance of efficiency and performance could match the predictive accuracy of LSTMs, especially given our relatively small dataset that might not require an overly complex model.

3. Approach

A. Training Methodology

The laser dataset, consisting of 1000 measurements, was divided into a training set with 750 samples and a validation set with 250 samples. All models were implemented in PyTorch and trained using the Adam optimizer [7], which is well-suited for handling sparse gradients and adaptive learning rates. To try and prevent overfitting, we employed the ReduceLROnPlateau learning rate scheduler, which reduces the learning rate when the validation loss plateaus. Additionally, early stopping was used to terminate training if the validation loss did not improve for 10 consecutive epochs (patience = 10), with training capped at a maximum of 100 epochs.

We used Mean Absolute Error (MAE) as the loss function instead of Mean Squared Error (MSE). MAE is less sensitive to outliers compared to MSE, making it more robust for real-world datasets with potential noise or anomalies [8].

B. Hyperparameter Optimization

Hyperparameter tuning was performed on the train and validation datasets using Optuna [9], a state-of-the-art framework for automated hyperparameter optimization. The following hyperparameters were tuned for each model:

- **Hidden Size**: Number of features in the hidden state, ranging from 128 to 256 in steps of 32.
- **Number of Layers**: Number of stacked RNN / LSTM / GRU layers, ranging from 1 to 3.
- **Learning Rate**: Initial learning rate for the Adam optimizer, sampled logarithmically between 10^{-5} and 10^{-2}.
- **Batch Size:** Number of samples per training batch, with options of 32, 64, and 128.
- **Window Size**: Number of past time steps used as input, ranging from 7 to 20 in steps of 1.

The best hyperparameters varied across the three models, reflecting their architectural differences and computational requirements. For the RNN, the optimal configuration included a hidden size of 224, two layers, a learning rate of 0.0012, a batch size of 128, and a window size of 15. This setup balanced the simplicity of the RNN architecture with sufficient capacity to capture temporal dependencies. In contrast, the LSTM achieved its best performance with a larger hidden size of 256, a single layer, a learning rate of 0.0025, a batch size of 64, and a smaller window size of 10. The larger hidden size and smaller window size highlight the LSTM's ability to effectively model long-term dependencies while requiring fewer input steps. Finally, the GRU performed best with a hidden size of 192, three layers, a learning rate of 0.0030, a batch size of 128, and a window size of 16. The GRU's smaller hidden size and deeper architecture leverage its efficient gating mechanisms to capture complex temporal patterns while maintaining computational efficiency. These differences underscore the trade-offs between model complexity, capacity, and the ability to generalize to the laser measurement dataset.

4. Results

The performance of all three models was systematically evaluated using both MSE and MAE metrics across training, validation, and test datasets. As shown in Table I, the GRU model outperformed the others on training and validation sets (achieving the lowest MSE of 89.336 and 1.123, respectively), followed by LSTM, with the baseline RNN performing worst. However, this pattern changed on the test set, where LSTM achieved superior performance, outperforming GRU by 0.257 in MAE and 29.452 in MSE.

TABLE I: Comparison of MSE and MAE across models (RNN, LSTM, GRU) on train, validation, and test sets. Rankings (in dark blue) indicate relative performance, with (1) being the best and (3) the worst.

Model	MSE (Train)	MSE (Validation)	MSE (Test)	MAE (Train)	MAE (Validation)	MAE (Test)
RNN	233.873 (3)	2.230 (3)	312.536 (3)	4.299 (3)	1.191 (3)	5.316 (3)
LSTM	111.199 (2)	1.738 (2)	116.885 (1)	2.732 (2)	1.060 (2)	3.494 (1)
GRU	89.336 (1)	1.123 (1)	146.337 (2)	2.492 (1)	0.827 (1)	3.751 (2)

This performance shift is visible in Fig. 1, which displays recursive forecasting results over 200 time steps. To initialize the forecasts, we used an initial window from the real signal: for the validation set, this consisted of `window_size` steps from the beginning of the validation set (excluded from the training set), while for the test set, it was the final `window_size` steps of the original training data, reflecting the fact that the test dataset is a direct continuation of the training sequence.

For the validation set, all three models captured the signal's frequency reasonably well, though LSTM and GRU predicted slightly longer wave widths. In terms of amplitude, the RNN aligned well with signal peaks but tended to overshoot over time due to accumulated errors, while LSTM consistently undershot, producing lower peaks, likely due to its higher complexity given the small dataset. The GRU achieved the best overall validation fit, closely matching the actual signal in both frequency and amplitude while requiring less training time.

For the test set, all models performed significantly worse. Initially, they captured frequency and amplitude well for the first few oscillations, with GRU performing best. However, none of the models could correctly predict the drastic drop in the amplitude of the actual data after 60 time steps. After this moment, all predictions differ significantly from the test

data. This discrepancy stems from dataset characteristics: the validation set lacks features such as signal drops followed by magnitude changes, favoring GRU's lightweight architecture. In contrast, the test set's complex patterns, including mid-signal shifts in magnitude, were better handled by LSTM's sophisticated memory mechanisms. This explains why models performed better on validation than test data, and why performance on training data (which included similar amplitude drops) was also affected.

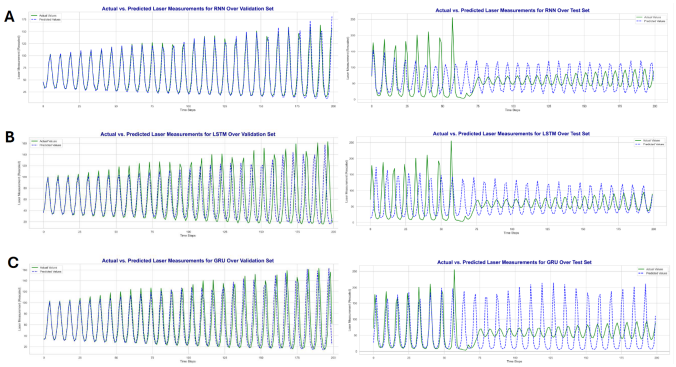

Fig. 1: Actual vs. Predicted Laser Measurements Across Three Model Types on Validation and Test Sets. Comparison of actual (green solid line) and predicted (blue dashed line) laser measurements over 200 time steps using recursive prediction for different recurrent neural network architectures: (A) RNN, (B) LSTM, and (C) GRU. The left column is validation, and the right is test. Each model was trained on historical data and then used to predict 200 steps ahead by recursively feeding each prediction back into the model to generate subsequent forecasts.

Another analysis we performed involved evaluating the impact of different hyperparameter values (as described in Sections 2 and 3.2) on the performance of each model. Figure 2 presents the normalized MAE loss values across different settings for hidden size, number of layers, window size, and batch size, for the GRU, LSTM, and baseline RNN models.

A few general trends emerge from the plots. First, across all three models, a batch size of 64 consistently leads to lower MSE values for both training and validation sets, compared to smaller (32) or larger (128) batch sizes. This suggests that 64 may strike a favorable balance between model update frequency and gradient estimation stability. Interestingly, although GRU and RNN achieve their best overall performance with a batch size of 128, this likely results from interactions with other hyperparameters, rather than the batch size alone. For LSTM, the best-performing configuration indeed uses a batch size of 64, consistent with the overall trend.

When analyzing hidden size and number of layers, we observe that increased model complexity does not necessarily yield better performance. For instance, the RNN achieves its lowest MSE with the smallest hidden size tested (128), suggesting that simpler configurations may be better suited for capturing the signal characteristics in this dataset. Across all models, increasing the hidden size does not consistently lead to lower loss. In some cases, larger hidden sizes are actually associated with worse performance, suggesting that simply increasing model capacity can result in overfitting or inefficiency rather than improvement. Similarly, increasing the number of layers does not consistently improve performance. In fact, for the LSTM, adding a third layer results in a sharp decrease in validation error, highlighting the risk of over-complicating the model architecture.

Finally, zooming in on the window size subplot, we see that the GRU's optimal configuration with a window size of 16 sits in a region of relatively stable performance, avoiding the dramatic spikes seen later in the graph. While the LSTM ultimately performed better on the test set, its selected window sizes of 10 and 15 show interesting contrasts, with size of around 10 appearing to be in a low-loss region, while size 15 coincides with a pronounced performance spike in the plot. The RNN exhibits particularly erratic behavior across all window sizes, suggesting its temporal processing capabilities are more sensitive to this hyperparameter than either GRU or LSTM.

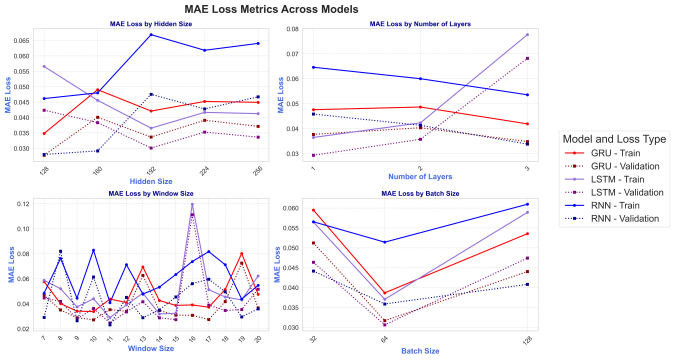

Fig. 2: MAE Loss Metrics Across Models and Hyperparameter Settings. Comparison of normalized MAE loss across GRU, LSTM, and RNN models for both training and validation data, showing performance variations based on different hyperparameters: hidden size, number of layers, window size, and batch size.

5. Conclusion

This study investigated the use of RNN-based architectures, including RNN, LSTM, and GRU, for time series forecasting on a laser measurement dataset. Despite employing models with varying memory learning capabilities and optimizing hyperparameters, the training approach likely resulted in overfitting on the training and validation sets, as reflected in the poor generalization to the test set.

To mitigate overfitting, we applied data augmentation by adding noise to the training set and using the augmented data as a validation set. We also experimented with incorporating dropout into the models. However, due to time constraints, these strategies were not fully optimized. Preliminary results suggest that these methods hold promise for improving generalization and mitigating overfitting, warranting further exploration in future work.

REFERENCES

[1] Alex Sherstinsky. "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network". In: *Physica D: Nonlinear Phenomena* 404 (2020), p. 132306.

[2] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[3] Kyunghyun Cho et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation". In: *arXiv preprint arXiv:1406.1078* (2014).

[4] Rahul Dey and Fathi M Salem. "Gate-variants of gated recurrent unit (GRU) neural networks". In: *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*. IEEE. 2017, pp. 1597–1600.

[5] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. "Learning long-term dependencies with gradient descent is difficult". In: *IEEE transactions on neural networks* 5.2 (1994), pp. 157–166.

[6] Liam Johnston et al. "Revisiting the problem of learning long-term dependencies in recurrent neural networks". In: *Neural Networks* 183 (2025), p. 106887. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2024.106887. URL: https://www.sciencedirect.com/science/article/pii/S0893608024008165.

[7] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[8] Cort J Willmott and Kenji Matsuura. "Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance". In: *Climate research* 30.1 (2005), pp. 79–82.

[9] Takuya Akiba et al. "Optuna: A Next-generation Hyperparameter Optimization Framework". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM. 2019, pp. 2623–2631. DOI: 10.1145/3292500.3330701.