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*—to be added**
***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 while assessing how accurately each model reproduces the signal characteristics in our test data.

2. Used Models

Time series forecasting involves capturing temporal dependencies and patterns across sequential data points. Recurrent Neural Networks (RNNs) are inherently suited for this task, as they maintain a hidden state that evolves over time, enabling the model to learn from previous time steps [1]. However, standard RNNs often struggle with learning long-term dependencies due to issues like vanishing gradients [2].

To address this, we also explored Gated Recurrent Units (GRUs) [3] and Long Short-Term Memory (LSTM) networks [4], which introduce information retention mechanisms that help retain relevant information over longer sequences. GRUs offer a simplified architecture compared to LSTMs while still handling longer dependencies more effectively than vanilla RNNs. By comparing these three architectures, our aim was to evaluate how increasing model complexity and memory capability affect performance on this forecasting task.

All three models follow a similar structure: a recurrent layer followed by a fully connected layer that outputs a single prediction per sequence. We used the following parameters for consistency across models:

- **input_size**: Number of features at each time step.
- **hidden_size**: Size of the hidden state, which determines the model's capacity to learn patterns.
- **num_stacked_layers**: Number of recurrent layers stacked on top of each other. Deeper networks can capture more complex patterns.
- **dropout**: Dropout rate between recurrent layers (used only when more than one layer is stacked), to reduce overfitting.

The LSTM model initializes both hidden and cell states at the beginning of each sequence. GRU and vanilla RNN architectures follow the same external structure but differ in their internal handling of memory and information flow.

3. Approach

A. Technical Details & Choices

We used Optuna to tune 3 previously mentioned parameters: **hidden_size**, **num_layers** and **dropout**. In addition, we also used it to optimize the following 3 hyperparameters:

- **lr**: Learning rate, controls the magnitude of change in weights in an optimization step.
- **batch_size**: Number of sequences in a batch
- **window_size**: The length of a sequence, amount of inputs used to calculate the output.

For the simple RNN model, we decided to use (rectified linear unit) ReLU as an activation function, because it is used often for training Neural Networks.

We used Mean Absolute Error (MAE) for the loss function, as it is easy to understand, and often used in regression problems.

We decided to use Adam optimization as the stochastic gradient descent method, as it decreases the amount of learning steps needed to reach a local minimum. It combines momentum (uses past gradients to accelerate descent process) with the cache from RMSprop (an improvement on Adagrad, reducing the step size in the biggest direction, while preventing the learning rate to become 0 over time).

B. Methodology

4. Results

to be added

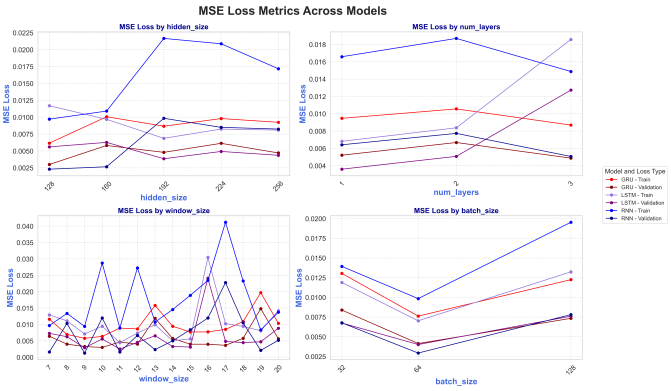

Fig. 1.

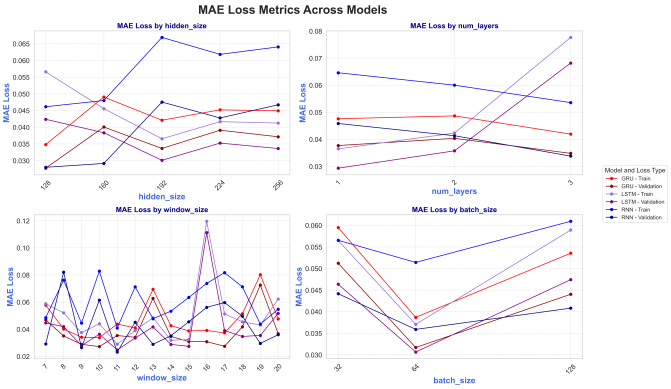

Fig. 2.

5. Conclusion

to be added

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] 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.

[3] 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.

[4] Yong Yu et al. "A review of recurrent neural networks: LSTM cells and network architectures". In: *Neural computation* 31.7 (2019), pp. 1235–1270.

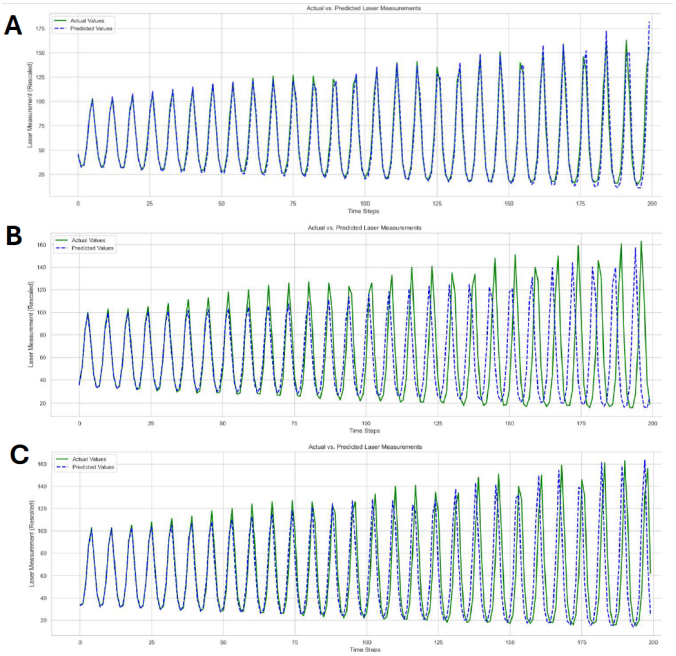

Fig. 3.