Sequence Model Comparison on Sunspot Dataset

Gabriel Urbaitis

March 6, 2025

1 Introduction

In this lab, the goal was to train various sequence models, including Recurrent Neural Networks (RNNs), Gated Recurrent Units (GRUs), Long Short-Term Memory (LSTM) networks, and the state-of-the-art Mamba model on NASA sunspot data. The models were implemented using Pytorch, various hyperparameters were tested to optimize for RMSE accuracy, and based on the results, a final configuration was chosen for each model for comparison. Models were compared using RMSE, R^2, inference time, and training time.

2 Background

2.1 Recurrent Neural Networks (RNNs)

An RNN performs its computations in a cyclic manner, where the same computation is applied to every sample of a given sequence. The idea is that the network should be able to use the previous computations as some form of memory and apply this to future computations. If input to the network is passed vertically through input hidden and output layers, a weight matrix w is passed vertically as the output of each timestep to the next layer, and a weight matrix v is passed horizontally as memory each time step to the next node in the same layer each timestep.

2.2 Long Short Term Memory (LSTMs)

LSTM networks add additional gating units in each memory cell, including a forget gate, an input gate and an output gate. These help to prevent the vanishing/exploding gradient problem and allow the network to retain state information over longer periods of time. An augmentation from RNNs is that instead of just passing the weight matrix v horizontally, the LSTM passes a cell state vector and the same output vector it passes to the next layer to the next node in the same layer. In other words the output vector is passed in two directions. Information can be added or deleted to the cell state vector via the forget and input gates.

2.3 Gated Recurrent Units (GRUs)

GRUs take similar input to LSTMs, but there is no cell state. GRUs don't need the cell layer to pass values along. The calculations within each iteration ensure that the ht values being passed along either retain a high amount of old information or are jump-started with a high amount of new information. This is because unlike LSTMs, GRUs combine the forget and input into a single update gate.

2.4 Mamba

Mamba is a selective state space model. SSMs are recurrent models that selectively process information based on the current input. This allows them to focus on relevant information and discard irrelevant data. Mamba employs a hardware-aware algorithm that exploits GPUs, by using kernel fusion, parallel scan, and recomputation. The implementation avoids materializing expanded states in memory-intensive layers, thereby improving performance and memory usage.[1]

3 Experiment

3.1 Hyperparameter Experimentation

32 possible configurations were tested on the four models, 4 simply extended the initial configurations from 10 to 30 epochs which showed no improvement. 16 others were on batch size and loss function, which were applied uniformly across the four models. The remaining 12 were configurations chosen specifically for each model based on their unique hyperparameters.

Baseline settings were as follows.

For all models: Batch Size: 64, Loss Function: 0.5 MSE + 0.5 RMSE, Epochs: 10, Optimizer: Adam (lr = 0.001)

For individual models: LSTM: Hidden Dim: 64, Num Layers: 2, Dropout: 0.2, Batch First: True RNN: Hidden Dim: 64, Num Layers: 2, Activation: Tanh, Batch First: True GRU:Hidden Dim: 64, Num Layers: 2, Batch First: True Mamba: d_model: 64, d_state: 16, d_conv: 4, expand: 2

The Uniform configurations were applied first. Proceeding downward, each change replaced a previous one or added to it if it was different. Meaning, each started at .5MSE .5 RMSE loss function, then .5MSE .5 RMSE loss function + batch size 32, then .7MSE .3 RMSE loss function + batch size 32, etc.

Model	Category	Train Loss	Val Loss	Train Time (s)	Inf Time (s)	RMSE	R^2 Score
LSTM	0.5 MSE 0.5 RMSE	0.034555	0.037452	13.21	0.0470	0.0596	0.8863
LSTM	Batch 32	0.033314	0.039112	7.40	0.0553	0.0618	0.8780
LSTM	0.7 MSE 0.3 RMSE	0.021268	0.026610	8.18	0.0684	0.0627	0.8744
LSTM	Just MSE	0.004428	0.004866	7.49	0.0680	0.0593	0.8873
RNN	0.5 MSE 0.5 RMSE	0.033895	0.035664	5.35	0.0478	0.0580	0.8923
RNN	Batch 32	0.033925	0.043775	9.10	0.0508	0.0684	0.8505
RNN	0.7 MSE 0.3 RMSE	0.021579	0.022800	9.40	0.0736	0.0575	0.8941
RNN	Just MSE	0.004245	0.005012	8.28	0.0523	0.0585	0.8905
GRU	0.5 MSE 0.5 RMSE	0.034378	0.035737	16.31	0.0860	0.0600	0.8849
GRU	Batch 32	0.033392	0.036921	25.52	0.1179	0.0597	0.8861
GRU	0.7 MSE 0.3 RMSE	0.021550	0.023535	24.36	0.1978	0.0589	0.8890
GRU	Just MSE	0.004091	0.007137	23.86	0.2068	0.0670	0.8563
Mamba	0.5 MSE 0.5 RMSE	0.034824	0.034667	1.75	0.0149	0.0571	**0.8958**
Mamba	Batch 32	0.033967	0.043892	2.63	0.0249	0.0666	0.8580
Mamba	0.7 MSE 0.3 RMSE	0.022697	0.026558	3.07	0.0240	0.0614	0.8794
Mamba	Just MSE	0.004238	0.005040	2.58	0.0236	0.0575	0.8944

Table 1: Comparison of Hyperparameter configuration applied uniformly

Out of the best uniform configurations, Mamba with 0.5 MSE 0.5 RMSE had the highest R^2 score with 0.8958, followed by RNN with 0.7 MSE 0.3 RMSE (batch 32) with .8941, GRU with 0.7 MSE 0.3 RMSE (batch 32) with .8890 and lastly LSTM with Just MSE (batch 32) with 0.8873.

As the configurations for each ended at Just MSE + batch size 32 before moving to the individualized configurations, this became the new baseline for the individualized changes. As with the uniform configurations, the changes for the individualized configurations were cumulative as well.

Model	Configuration	Train Loss	Val Loss	Train Time (s)	Inf Time (s)	RMSE	R^2 Score
LSTM	hidden_dim 128	0.003926	0.004466	26.13	0.2124	0.0567	**0.8972**
LSTM	hidden_dim 256	0.004041	0.005192	88.59	0.4637	0.0586	0.8903
LSTM	Dropout .3	0.004133	0.004800	100.23	0.5262	0.0598	0.8855
LSTM	Layer normalization	0.004527	0.006717	22.39	0.1393	0.0669	0.8568
RNN	Relu 2 layers	0.004400	0.005619	9.20	0.0463	0.0602	0.8841
RNN	Relu 3 layers	0.004219	0.005258	12.26	0.0633	0.0617	0.8781
RNN	Tanh 3 layers	0.004422	0.005276	12.70	0.1121	0.0606	0.8825
GRU	hidden_dim 48	0.004103	0.004827	19.63	0.1002	0.0579	0.8927
GRU	hidden_dim 32	0.004122	0.004853	18.11	0.1422	0.0576	0.8940
GRU	Dropout .1	0.004148	0.004884	17.89	0.1345	0.0591	0.8881
Mamba	D state 32	0.004154	0.004812	3.67	0.0369	0.0575	0.8943
Mamba	Expand = 4	0.004180	0.004775	2.90	0.0325	0.0600	0.8848

Table 2: Comparison of Hyperparameter configuration applied individually

Out of the best individualized configurations, LSTM with hidden dim 128 had the highest R^2 score with .8972, followed by Mamba with D state 32 and expand 2 with .8943, GRU with hidden dim 48 with .8927 and lastly RNN with Relu and 2 layers with .8841.

3.2 Choosing final configurations

The final configuration for each model was chosen by combining the highest hyperparameter configurations for both the uniform and individualized trials. The results are below.

Model	**Training Time (s)**	**Inference Time (s)**	**RMSE**	**R^2 Score**
LSTM	35.20	0.1318	0.0566	0.8974
RNN	12.87	0.0692	0.0578	0.8931
GRU	22.44	0.1194	0.0594	0.8871
Mamba	1.97	0.0229	0.0578	0.8931

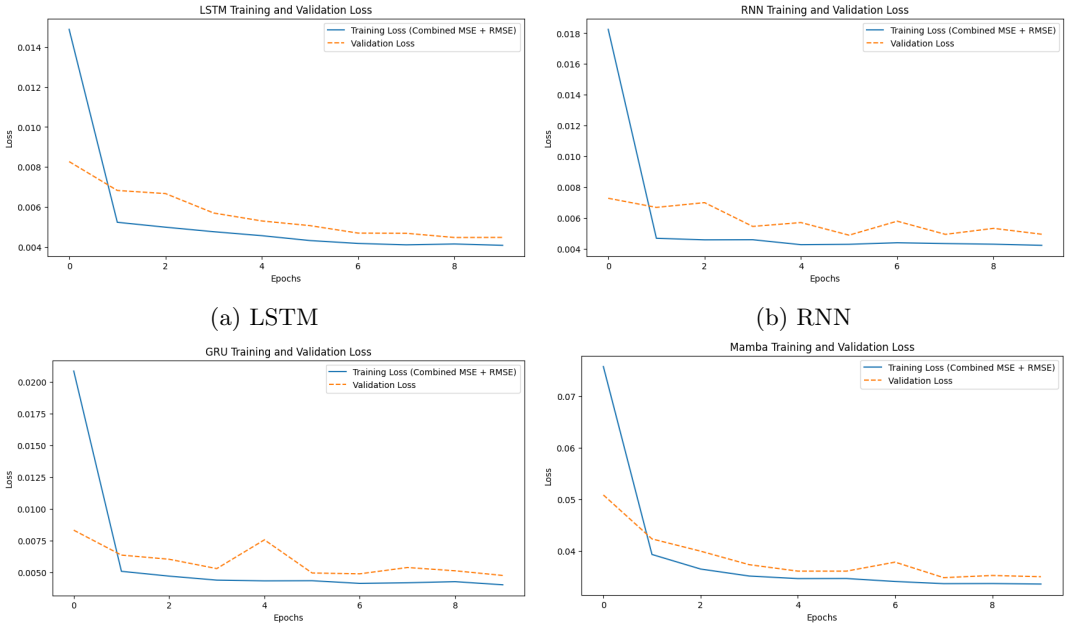
Table 3: Final Configuration Model Comparison

Not all combinations resulted in higher accuracies, whether lower RMSE or higher R^2. This was because R^2 could vary by .005 each run on the same configuration. It is somewhat noteworthy, however, that the combined configuration did improve the best overall configuration from the previous two trial sets, albeit only by .0002 R^2 score. A closer look at each of the final models is taken in the following section.

4 Results

First, Loss Curves will are examined, followed by a comparison of each model with regard to RMSE, R^2, training time, and inference time, and finally predictions from each model are visualized against the actual sunspot values.

4.1 Loss Curves



(a) LSTM

(b) RNN

(c) GRU

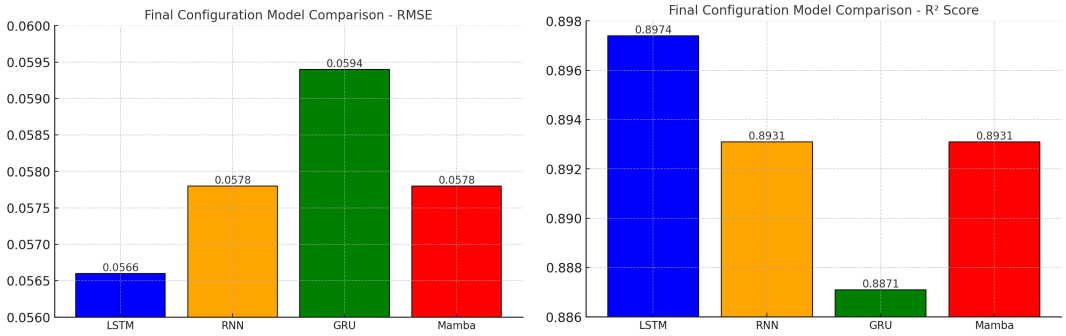
(d) Mamba

Figure 1: Loss Curves for Different Models

All models show a rapid initial decrease in training loss, indicating effective learning the first epoch. Training loss starts highest in Mamba, indicating it may struggle more in early training stages.

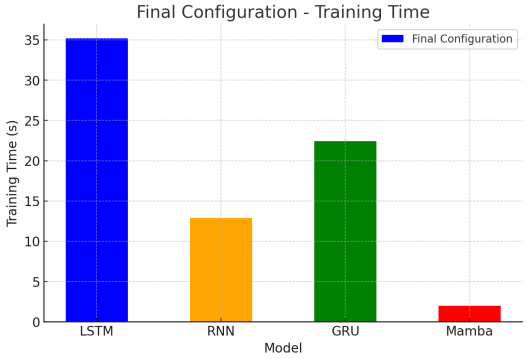
Validation loss is relatively stable, meaning the models did not suffer from instability but also did not generalize perfectly. LSTM's validation loss followed a similar trend to its training loss but stabilized at a slightly higher level which suggests good learning with minor overfitting. After 4 epochs, RNN's validation loss did not improve significantly which indicates RNN did not generalize as effectively as LSTM. GRU's validation loss fluctuated at epoch 3-5, which may indicate some instability. Mamba's validation loss is the highest of the 3 which could suggest weaker generalization.

4.2 RMSE, R^2, Training Time, and Inference Time Comparison

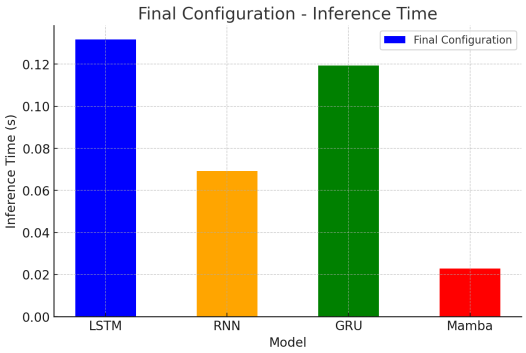


(a) Model Comparison - RMSE

(b) Model Comparison - R^2 Score



(c) Model Comparison - Training Time



(d) Model Comparison - Inference Time

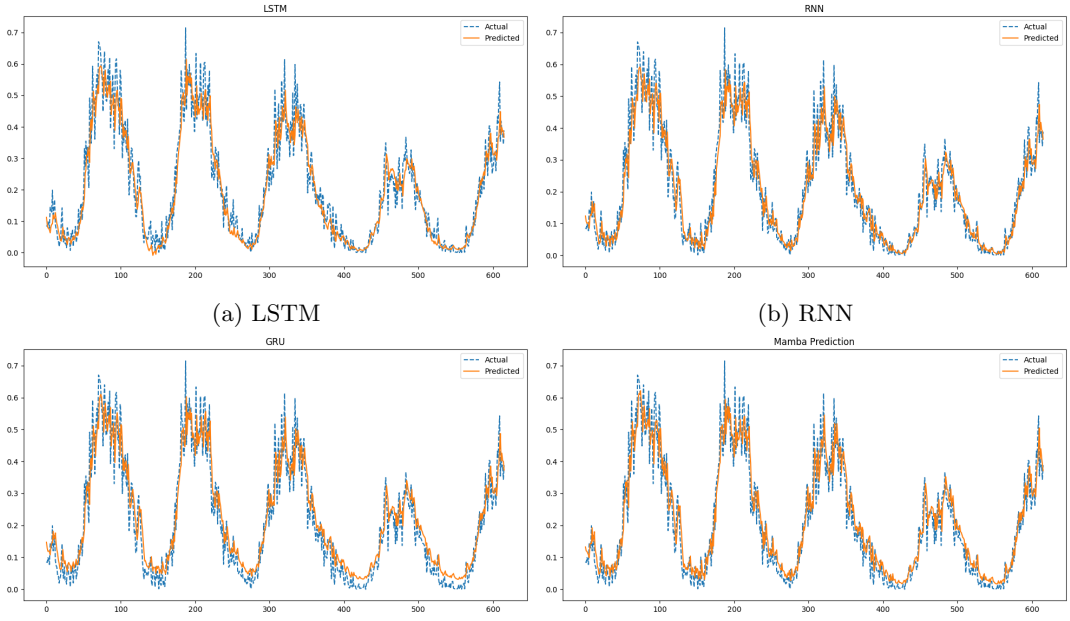
Figure 2: Final Configuration Model Comparisons and Performance Metrics

RMSE and R^2 are compared together, as higher R^2 accuracy means lower error and thus lower RMSE. GRU had the highest error and lowest accuracy in the final configuration runs, RNN and Mamba tied in the middle, and LSTM had the highest accuracy and lowest error.

The trends were the same for Training and Inference time for all 4 models, with LSTM having the highest times, followed by GRU, then RNN and finally Mamba. The ratios were closer though for inference time, indicating there is likely a significant process that all 4 have to run at inference time.

Overall, LSTMs seem to be the best choice if time is not a factor, but otherwise Mamba is able to get somewhat close in far less time, though as discussed earlier, its generalization may be a concern.

4.3 Predictions vs Actual



(a) LSTM (b) RNN

(c) GRU (d) Mamba

Figure 3: Predictions vs Actual for Different Models

LSTM seems to capture the overall structure of the data best, though the smoothness of the predictions indicates it might be filtering out some noise. RNN and Mamba have similar performance, but struggle more with the high variation regions, indicating less adaptability to rapid changes. GRU has the opposite problem, it handles the high variation regions well, but in the valleys consistently has a gap where it predicts higher values than actual.

4.4 Challenges

Overfitting occurs when a model learns the training data too well, but fails to generalize to new unseen data. GRU had the most trouble with overfitting, and this is likely because it has a larger number of parameters, which makes it harder to find a generalizable configuration.

The vanishing gradient problem occurs when gradients have to propagate backward through many time steps, they shrink exponentially, making it difficult for early layers to learn meaningful representations. RNNs have an issue with this as they struggle to learn long-term dependencies and only capture short-term patterns. In the data, this can be seen as increasing the number of layers had a negative effect on the accuracy. The LSTM and GRU's forget and update gate, respectively, help to mitigate this by allowing gradients to pass through longer timesteps without vanishing.

Finding optimal values for hyperparameters like hidden dimensions, batch size, dropout rate, and learning rate required significant experimentation. Small changes in parameters had noticeable impacts on RMSE and R^2 performance. Combining RMSE and MSE as a loss function introduced complexity, and it was unclear whether this offered real predictive benefits compared to using a single metric.

5 Conclusion

Several types of sequence models, including Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), Gated Recurrent Units (GRU), and the Mamba model—were trained and evaluated on NASA's sunspot dataset. Each model was tested with various hyperparameters to determine optimal settings based on metrics including RMSE, R^2 , training time, and inference time.

LSTM provided the best overall accuracy, performing well in predicting sunspot activity but required the longest training and inference times. The Mamba model was the fastest in both training

and inference, though its accuracy was slightly lower compared to LSTM. RNNs performed reasonably well in accuracy and speed but struggled with capturing longer-term patterns, likely due to vanishing gradient problems. GRUs showed promising theoretical advantages but experienced issues such as overfitting and instability during training.

In summary, LSTM is a good choice when accuracy is prioritized over computation time, while Mamba offers a valuable trade-off with fast performance and acceptable accuracy. Future work could explore combining multiple models or further hyperparameter tuning to enhance predictions and efficiency.

References

[1] Albert Gu and Tri Dao. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. *arXiv preprint*, arXiv:2312.00752, 2023. Available at: https://arxiv.org/abs/2312.00752