Time Series Prediction using Recurrent Neural Networks (RNNs)

Rishona Daniels - 884083130, Shaked Leslau - 205383474

April 8, 2024

1 Introduction

Any system that evolves with time is a dynamical system. All real-world processes are dynamical systems and it is useful to predict the trajectory of these systems using mathematical models. For example, weather prediction, stock-market prediction, physiological systems, etc. However, modeling of dynamical systems is complicated because of the large number of parameters that affect the trajectory. Also, these systems are very sensitive to the initial conditions and a small change in the initial conditions can have a dramatic impact on the trajectory of the output. Various machine learning approaches are used to generate models of dynamical systems using observed data however, these approaches are data hungry, require long observation times and consume large computational resources.

1.1 Project Goal

The goal of this project is to compare various recurrent neural networks (RNNs) for the prediction of dynamic and chaotic time series data. A comparison will be made between two standard RNN models namely long-short term memory (LSTM), gated recurrent unit (GRU) as well as a novel RNN model called non-linear vector autoregression (NVAR) also known as reservoir computing. The attention-based transformer model will also be compared on time series data.

The datasets used are two differential equation systems that exhibit deterministic chaos.

1. Lorenz - 63 system
2. Mackey-Glass equations

1.1.1 Lorenz-63 system

$$\frac{dx}{dt} = 10(y - x)$$

$$\frac{dy}{dt} = x(28 - z)$$

$$\frac{dx}{dt} = xy - \frac{8z}{3}$$

Here, the lyapunov time is 1.1 time units

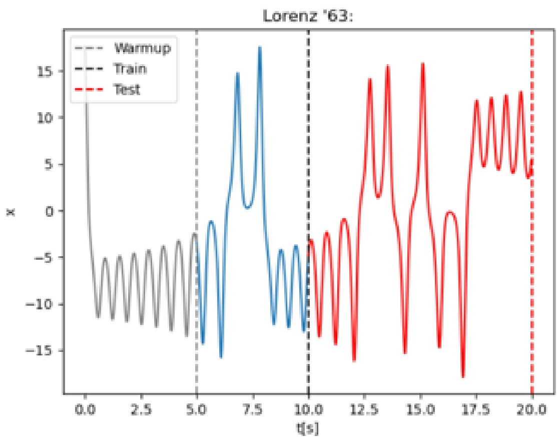


Figure 1: Lorenz-63 (x vs time)

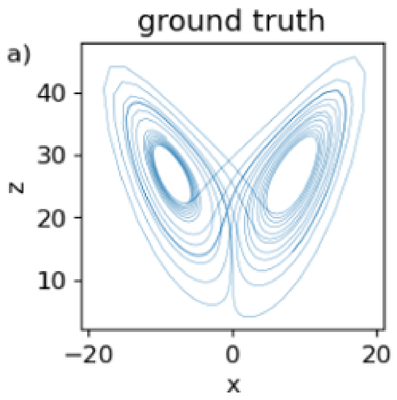


Figure 2: Lorenz-63

1.1.2 Mackey-Glass equations

$$\frac{dx}{dt} = \beta \frac{x_\tau}{1 + x_\tau^\eta} - \gamma x$$

where γ, β and $\eta > 0$

Here, β, γ, τ and η are real numbers and x_τ is the value of x at time $(t - \tau)$

Here, $\beta = 0.2$, $\gamma = 0.1$, $\eta = 10$ and $\tau = 17$

When $\tau \geq 17$ the system is chaotic in nature

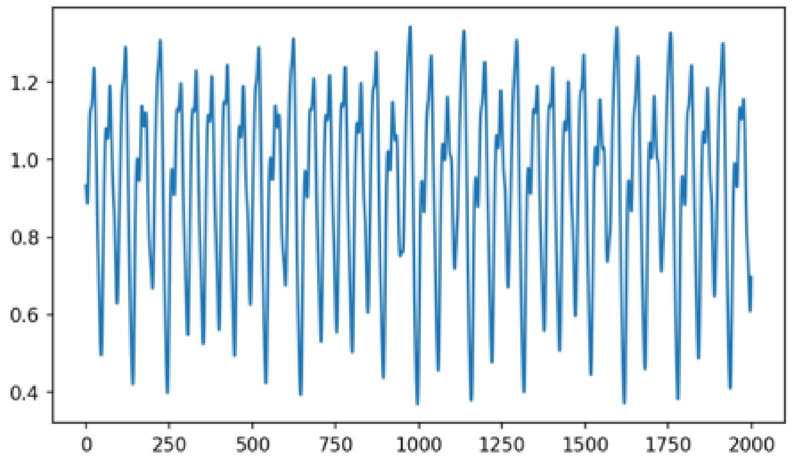


Figure 3: Mackey-Glass Equation

1.2 Motivation/Ethics Statement

The stakeholders of this project include weather forecasting stations, organizations trading stocks, etc. Weather forecasting stations will be able to use these methods to model and predict the weather forecast using observed time-series data. Organizations trading stocks can make decisions based on the predictions of these models An ethical consideration is that these models can be used for deterministic chaotic systems. Real-world systems may not be deterministic so care must be taken when designing these models. Also, the reliability of these models must be thoroughly evaluated before deployment in real-world applications especially considering the systems in question are sensitive to initial conditions.

1.3 Previous Works

Prior works have compared various methods for time series prediction. In particular, in [2] gated recurrent neural networks including long short-term memory (LSTM) and gated recurrent unit (GRU), echo state networks (ESN) and non-linear vector autoregression (NVAR) have been compared. However, the attention-based transformer model has not been included in the comparison.

In this project, we aim to reproduce the results of [2] and use the transformer for time series prediction.

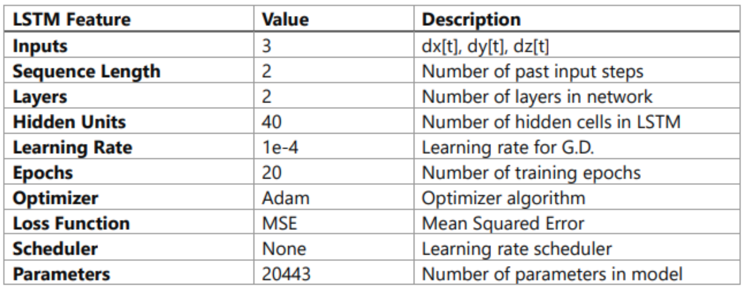
2 Methods

We used PyTorch toolbox to implement the LSTM, GRU, and transformer. For NVAR, we using the implementation given in [1]. They have implemented NVAR using Numpy and their implementations are in their GitHub repository. To provide a valid comparison, we needed to maintain consistency across the data sets, models, and loss functions. We trained each model on the generated Lorenz 63' dataset which we normalized, and tested it on the continuation of the same sequence.

When tuning the hyperparameters, we found that the number of epochs had the greatest effect on the outcome. Additionally, we found that increasing the complexity of the model did not improve the result as expected. We surmise that this is because the dataset was relatively small (on the order of 800 – 2000 samples), and so the model quickly began to overfit, neutralizing any benefits from a highly complex model. Additionally, we found that the results of each of the models varied greatly across training runs, even when the hyperparameters were left constant. The model that varied the most was the transformer. We believe this is because it is very data hungry, and our training data was simply too small to properly train on. Additionally, the nature of the chaotic timeseries makes it very complex to learn, and small differences in initial conditions can drastically change the results.

2.1 Long-short term memory (LSTM)

The Long Short Term Memory network is a type of RNN that is designed to overcome the limitations of traditional RNNs. At its core, an LSTM network comprises a series of memory cells, each equipped with three gates: an input gate, a forget gate, and an output gate. These gates regulate the flow of information within the network, enabling it to selectively remember or forget information over time. Our LSTM has the following features and hyperparameters:



LSTM Feature	Value	Description
Inputs	3	dx[t], dy[t], dz[t]
Sequence Length	2	Number of past input steps
Layers	2	Number of layers in network
Hidden Units	40	Number of hidden cells in LSTM
Learning Rate	1e-4	Learning rate for G.D.
Epochs	20	Number of training epochs
Optimizer	Adam	Optimizer algorithm
Loss Function	MSE	Mean Squared Error
Scheduler	None	Learning rate scheduler
Parameters	20443	Number of parameters in model

Figure 4: LSTM Model Features

After manually tuning the model, we trained it over 20 epochs and evaluated the model on the test set.

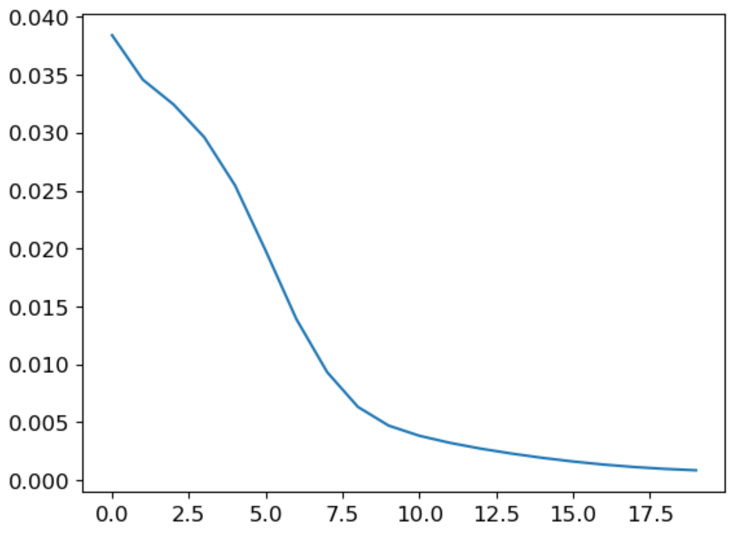
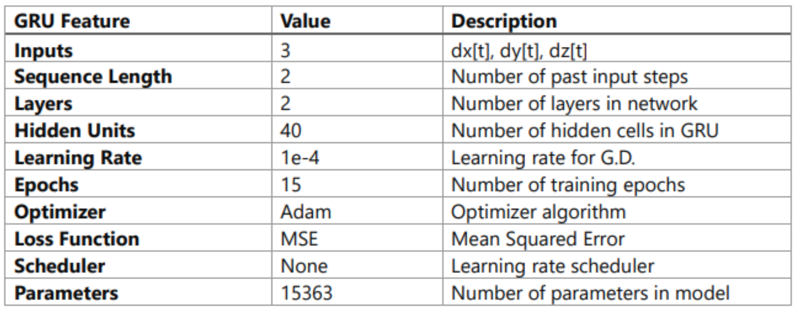


Figure 5: LSTM Training Loss

2.2 Gated recurrent unit (GRU)

The Gated Recurrent Unit (GRU) is a streamlined variant of recurrent neural networks (RNNs), featuring two gates: an update gate and a reset gate. These gates collaborate to regulate information flow, enabling GRU networks to effectively capture long-term dependencies in sequential data. GRUs offer comparable performance to LSTM networks with simpler architecture and computational efficiency. Our GRU has the following features and hyperparameters:



GRU Feature	Value	Description
Inputs	3	dx[t], dy[t], dz[t]
Sequence Length	2	Number of past input steps
Layers	2	Number of layers in network
Hidden Units	40	Number of hidden cells in GRU
Learning Rate	1e-4	Learning rate for G.D.
Epochs	15	Number of training epochs
Optimizer	Adam	Optimizer algorithm
Loss Function	MSE	Mean Squared Error
Scheduler	None	Learning rate scheduler
Parameters	15363	Number of parameters in model

Figure 6: GRU Model Features

After manually tuning the model, we trained it over 15 epochs and evaluated the model on the test set.

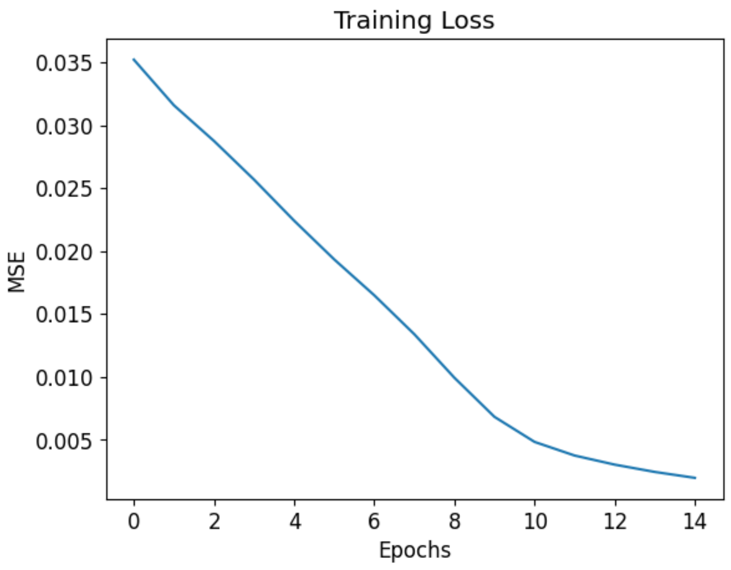
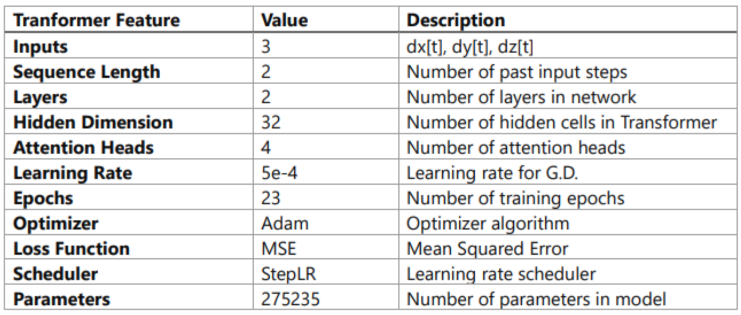


Figure 7: GRU Training Loss

2.3 Transformer

The Transformer architecture revolutionized sequence modeling by employing self-attention mechanisms for parallel processing of input sequences, eliminating the need for recurrent connections. This approach allows tokens to capture global dependencies efficiently, leading to superior performance in natural language processing tasks such as machine translation and text generation. We used the standard encoder-decoder architecture with the following features and hyperparameters:



Tranformer Feature	Value	Description
Inputs	3	dx[t], dy[t], dz[t]
Sequence Length	2	Number of past input steps
Layers	2	Number of layers in network
Hidden Dimension	32	Number of hidden cells in Transformer
Attention Heads	4	Number of attention heads
Learning Rate	5e-4	Learning rate for G.D.
Epochs	23	Number of training epochs
Optimizer	Adam	Optimizer algorithm
Loss Function	MSE	Mean Squared Error
Scheduler	StepLR	Learning rate scheduler
Parameters	275235	Number of parameters in model

Figure 8: Transformer Model Features

After manually tuning the model, we trained it over 23 epochs and evaluated the model on the test set.

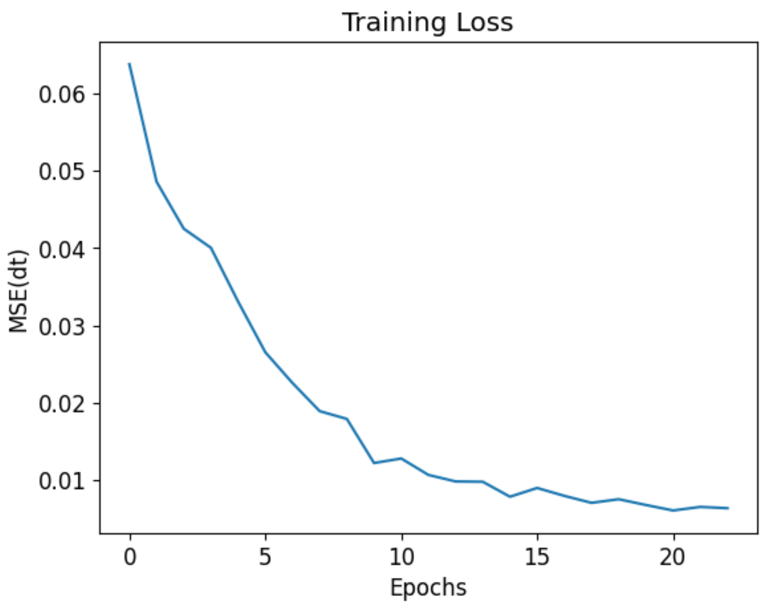


Figure 9: Transformer Training Loss

2.4 Non-linear vector autoregression (NVAR)

Non-linear vector autoregression (NVAR) is a method that applies a linear transformation on the input data and then forms a feature vector using time-delayed observations of the dynamical system. This is then augmented using non-linear functions of these observations. [2] This approach requires fewer hyperparameters and does not require any random matrix initialization.

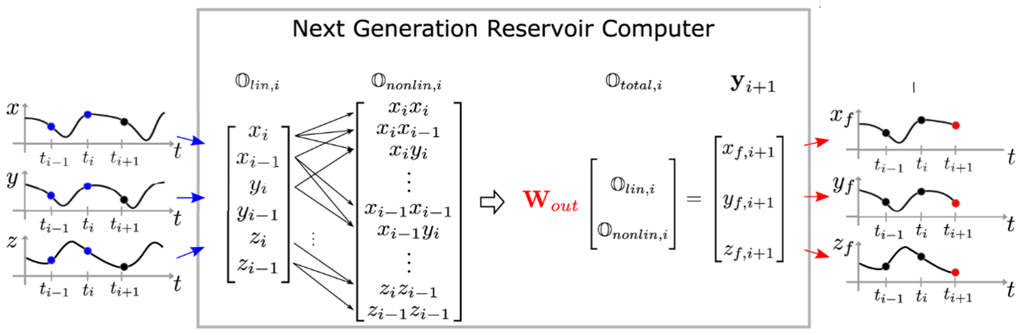

Next Generation Reservoir Computer
$\mathbb{O}_{lin,i}$
$\mathbb{O}_{nonlin,i}$
$\mathbb{O}_{total,i}$
$\mathbf{y}_{i+1}$
$\mathbf{W}_{out}$
x_i, x_{i-1}, y_i, y_{i-1}, z_i, z_{i-1}
x_ix_i, x_ix_{i-1}, x_iy_i, $x_{i-1}x_{i-1}$, $x_{i-1}y_i$, z_iz_{i-1}, $z_{i-1}z_{i-1}$
$x_{f,i+1}$, $y_{f,i+1}$, $z_{f,i+1}$
t_{i-1} t_i t_{i+1} t

Figure 10: Block diagram of Non-linear vector autoregression

3 Experiments

We trained each tuned model uisng the training data, which we normalized and seperated into sequences of length n. We then evaluated each model by feeding it the first sequence of the test data set. We fed the model's prediction back into the input in a sliding window, and then measured the MSE and NRMSE of the respective model's prediction over a single Lyapunov time step. As we see in the figures below, the LSTM, GRU and transformer were able to predict the next step within Lyapunov time but after that quickly starts to diverge. However, NVAR was able to predict the next steps for a longer duration of time. We also observed that the LSTM, GRU and transformer took significantly longer to train than the NVAR.

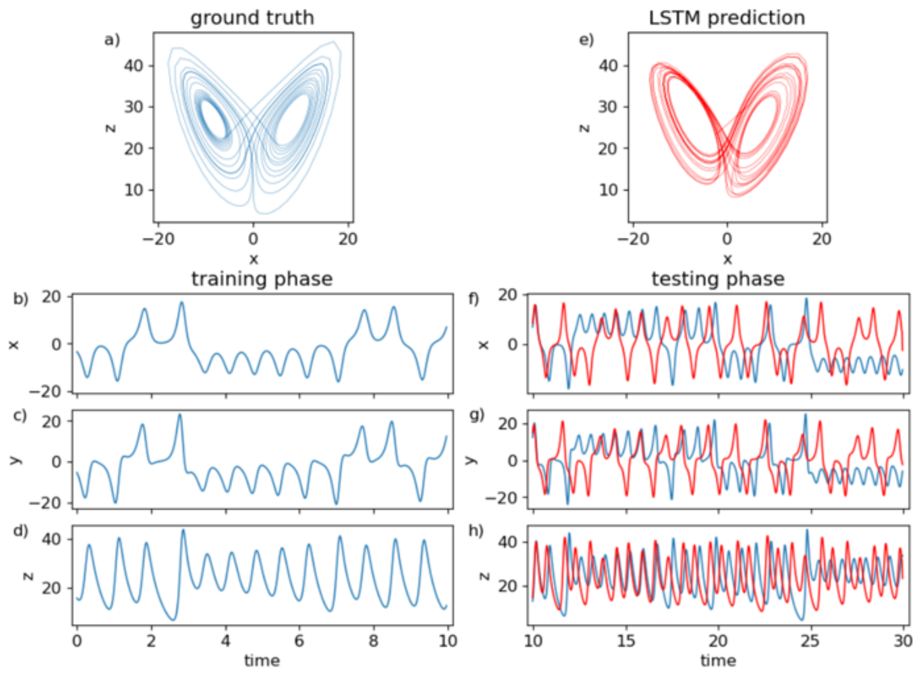


Figure 11: LSTM

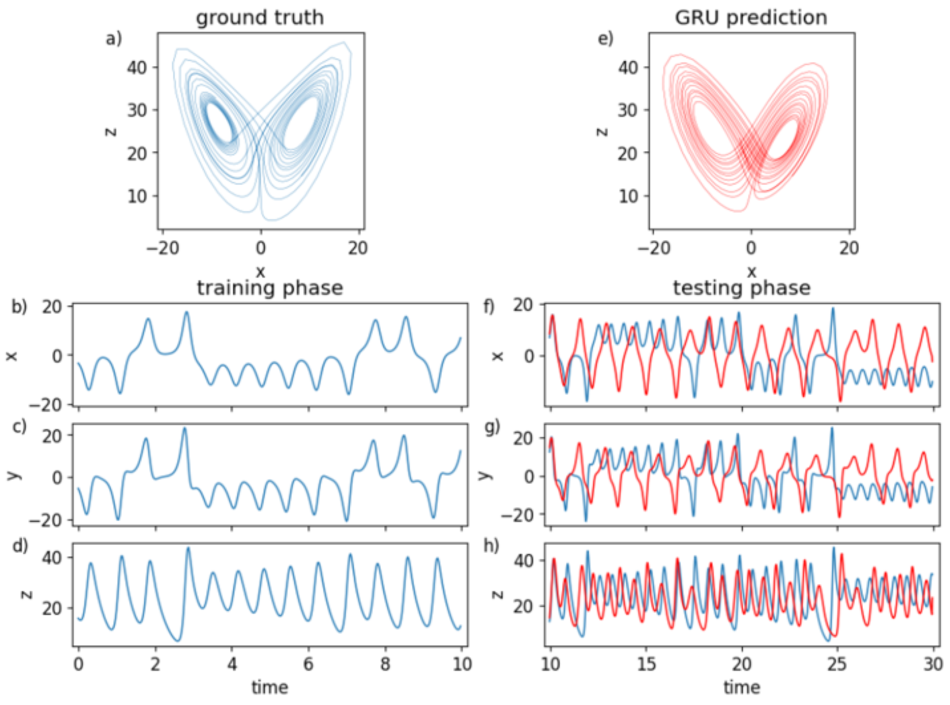


Figure 12: GRU

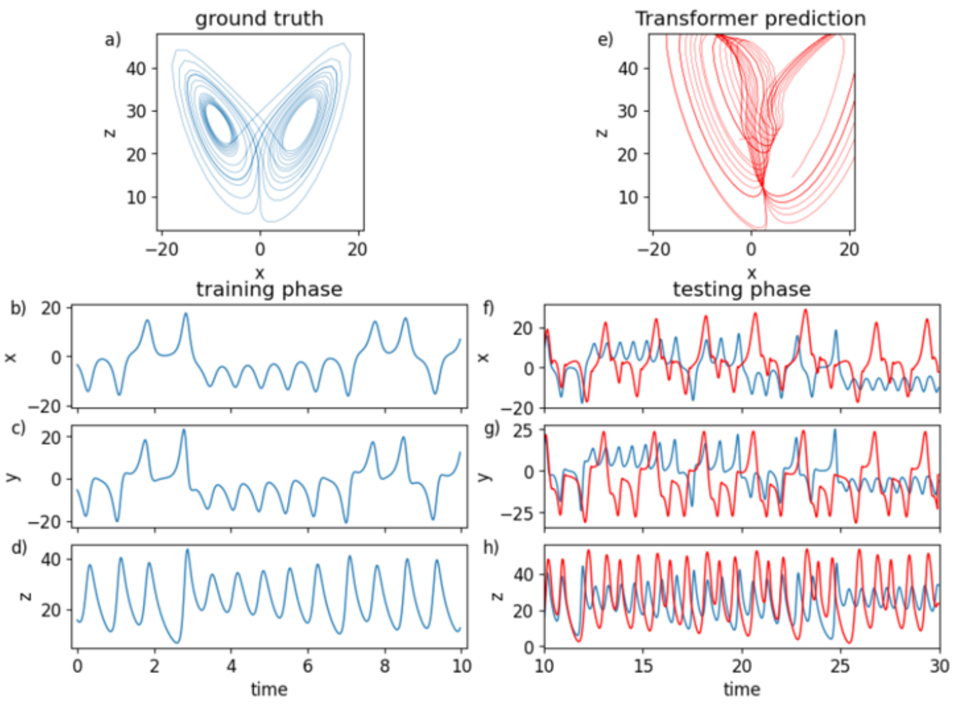


Figure 13: Transformer

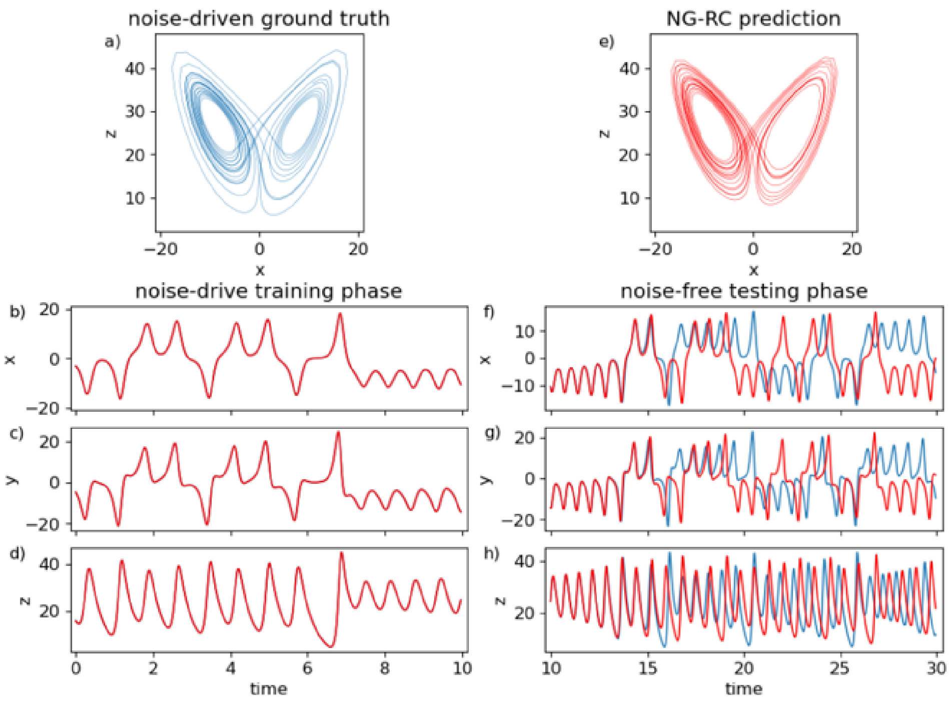
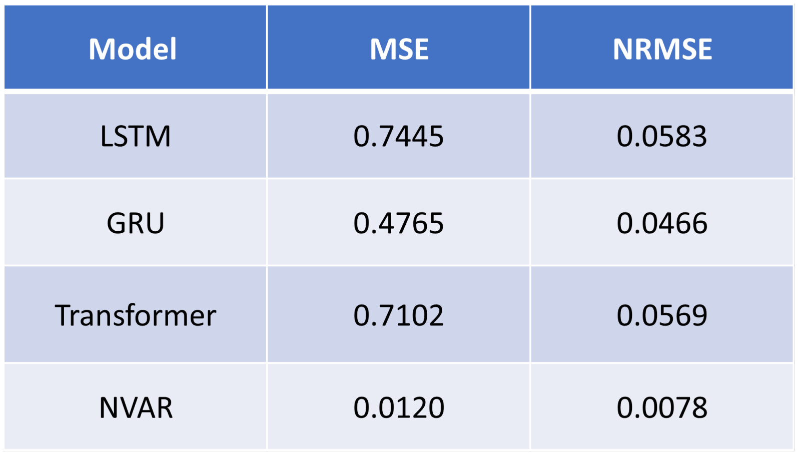


Figure 14: Non-linear vector autoregression

4 Results



Model	MSE	NRMSE
LSTM	0.7445	0.0583
GRU	0.4765	0.0466
Transformer	0.7102	0.0569
NVAR	0.0120	0.0078

Figure 15: Summary of results for Lorenz-63

As we see the table, NVAR outperforms all other methods on the lorenz-63 dataset. It has the lowest mean square error (MSE) and the lowest normalized root mean square error (NRMSE). These results are in line with the results from the literature. The next best model was the GRU followed by the transformer and the LSTM. However, the MSE of these methods was an order of magnitude higher than NVAR. Also, these models had very long training times.

5 Conclusion and Future Work

To conclude, NVAR is the best-in-class method for chaotic time-series prediction as it does not require any random weight initialization. It requires shorter training time, lower error and can predict the trajectory of the sequence for longer time with better accuracy as compared to other models. NVAR method was able to outperform the attention based transformer model as well. Future work includes implementation of the Mackey-Glass equations. More recent models like RKWV and MAMBA-SSM can also be compared to assess their performance.

6 GitHub Repository

All the codes used in this project can be found in the link below

https://github.com/sLeslau/Time-Series-Prediction-using-RNNs

References

[1] Daniel J. Gauthier et al. "Next generation reservoir computing". In: *Nature Communications* 12.1 (Dec. 2021). ISSN: 20411723. DOI: `10.1038/s41467-021-25801-2`.

[2] Shahrokh Shahi, Flavio H. Fenton, and Elizabeth M. Cherry. “Prediction of chaotic time series using recurrent neural networks and reservoir computing techniques: A comparative study”. In: *Machine Learning with Applications* 8 (June 2022), p. 100300. ISSN: 26668270. DOI: `10.1016/j.mlwa.2022.100300`.