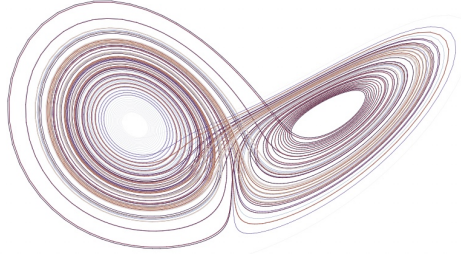


Introduction to Neural Networks for Predicting Chaotic Systems

Sebastian M.D.

May 2024



1 Introduction

Chaotic systems are deterministic systems which are highly sensitive to initial conditions, making long term predictions challenging. One such system aimed to model atmospheric convection is called the Lorenz 63 system, envisioned by Edward Lorenz in 1963. Traditionally, numerical methods have been used to predict these systems, but recent research has explored neural networks as an alternative solution. This is a summary of the paper: "Which Neural Network architecture is Optimal for Predicting Chaotic Dynamics?" (2024, Delgado S.).

2 Theory

2.1 The Lorenz System

The Lorenz system is a set of three mathematical equations defined as follows:

$$\frac{dx}{dt} = \sigma(y - x), \quad \frac{dy}{dt} = x(\rho - z) - y, \quad \frac{dz}{dt} = xy - \beta z$$

In these equations, x , y , and z are variables that change over time, and σ , ρ , and β are constants. When we use these values, the equations create a pattern called the Lorenz attractor, see figure 1. This pattern looks like a butterfly's wings and shows how the system evolves over time. Even though the system is deterministic, its behavior is highly sensitive to initial conditions, meaning that a small change in initial starting values x , y , z , can lead to radically different x , y , z values after a certain t timesteps. This is why the Lorenz system is a great example of chaos theory.

2.2 Neural Networks

Neural networks are a type of machine learning model inspired by how the human brain works. They consist of layers of interconnected nodes, which act like neurons, processing input data to produce an output. The connections between these nodes have weights and biases, which

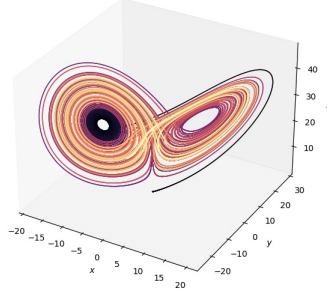


Figure 1: The Lorenz attractor visualized for 100 time units

are parameters that decide what will get passed to the next node. The weights and biases are adjusted during the learning process.

To network starts out with random parameters, then by measuring the error in prediction it will make small adjustments to these parameters that minimize the error. To measure the error in predictions, a cost function is used. The Mean Squared Error (MSE) is commonly used in regression tasks. By adjusting the weights in this way, the neural network learns to make accurate predictions on new, unseen data over time.

2.3 RNNs

A Recurrent Neural Network (RNN) is a type of neural network designed to recognize patterns in sequences of data. Unlike traditional neural networks, RNNs have connections that form loops. This allows them to maintain a 'memory' of previous inputs, which helps in processing sequential data like the Lorenz system.

In an RNN, each step in the sequence has a hidden state that captures information from previous steps. When an input is fed into the RNN, it updates its hidden state and produces an output. This hidden state is then passed to the next step along with the next input, enabling the network to remember past information.

Long Short-Term Memory (LSTM) networks are an improved RNN architecture that is used in this study, LSTMs use special units called gates to control the flow of information and improve long term memory.

2.4 Transformers

Transformers are a type of neural network introduced in the paper "Attention is All You Need". Unlike previous models, they don't read data step-by-step but process the whole sequence at once. This allows them to work faster and handle more complex tasks.

The key idea in Transformers is the self-attention mechanism. Each word in a sentence is turned into three vectors(three lists of numbers) that describe what information the word is looking for, what information the word offers and the information the word actually contains. The self-attention mechanism compares these vectors to find out how much each word should pay attention to the others. For example, in the sentence "The cat sat on the mat," the word "cat" may realize it is the subject of the sentence and start looking for the object, "mat", thus self-attention helps find the relationships between the words in a sentence. The attention mechanism is usually applied in parallel blocks that form a layer. These layers are repeated a set number of times to form the full transformer network.

2.5 Reservoir Computing

Reservoir Computing (RC) is a type of Recurrent Neural Network (RNN) that is especially good at predicting chaotic systems. The main idea is that only part of the network is trained, making it faster and solving some problems found in traditional RNNs.

Echo State Networks (ESNs) are a popular example of RC. In ESNs, the network has a random, sparsely connected part called the reservoir. The reservoir’s state at time t , $x^{(t)}$, is updated based on the previous state $x^{(t-1)}$, the input $u^{(t)}$, and the feedback from the previous output $y^{(t-1)}$. Only the output layer is trained, typically using a method like ridge regression, to minimize the error between the predicted and actual outputs.

3 Method

This project involves creating, training, and evaluating machine learning models using Python, the PyTorch library and the ReservoirPy library. The project’s code is available on GitHub at: <https://github.com/SebCodesTheWeb/lorents-net>.

3.1 Generating Data

Data was generated using a numerical method called the RK4 method. This method simulates the Lorenz system with random initial positions, and then records and saves the positions of the system as it moves through time in a CSV file. This data was then used to train the neural networks.

3.2 Setting up the models

The LSTM RNN model was set up using PyTorch’s inbuilt `nn.LSTM` class. The Adam optimizer was used to train the parameters. The Transformers model was also built using PyTorch and trained similiary to the LSTM RNN model. The ESN model was implemented using the ReservoirPY library and was trained with inbuilt ridge regression optimizer from ReservoirPy.

3.3 Hyperparameter Optimization

Hyperparameters are parameters that control the training process of a neural network. Hyperparameters could decide things like the number of layers, learning rate, training time, and network size of the model. A library called Optuna was used to find the best hyperparameters for the Transformers and LSTM RNN models. This process involves running multiple trials and selecting the best combination of parameters, it’s computationally heavy and was trained for 100 iterations per model on four RTX 4090 GPUs.

Once the models were trained they were then evaluated by predicting 500 time steps of the Lorenz system and comparing the predictions to the RK4 method. To evaluate the models, their Lorenz attractor shape, trajectory and MSE was taken into account.

4 Results

4.1 Lorenz attractor

4.2 Trajectory

View figures 5, 6 and 7 in Appendix A.

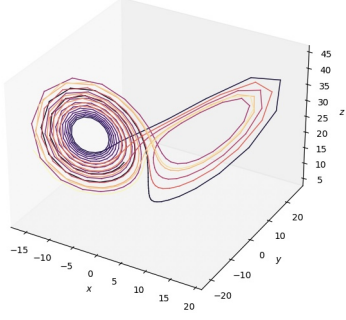


Figure 2: LSTM RNN

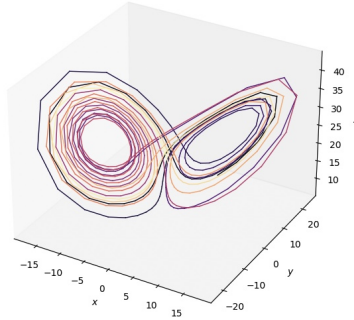


Figure 3: ESN

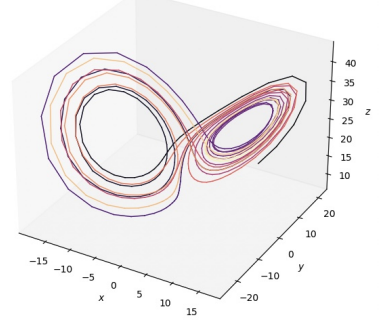


Figure 4: Transformers

4.3 MSE

The average MSE for the LSTM RNN was 50.1579.

The average MSE for the ESN was 48.7797.

The average MSE for the Transformers network was 29.1590.

5 Discussion

The figures show that all models have learned the Lorenz patterns, producing attractors with two wings around two focal points. However, there are differences in accuracy. The Transformers model has the most accurate angle between the wings, but the proportions are off. The ESN model has better proportions between the two rings which are supposed to be of close to equal radius. Overall, the ESN model appears to have the most accurate Lorenz attractor shape.

The trajectory plots indicate different levels of accuracy. The LSTM RNN deviates from the true path around timestep 40-50. The ESN model diverges around timestep 60-100. The Transformers model diverges around timestep 100-120. Considering the input sequence lengths, the ESN can predict accurately for 70-80 timesteps, the LSTM RNN for 30-40 timesteps, and the Transformers model for about 60 timesteps.

5.1 Conclusion

The results suggest that the LSTM RNN was the worst performing model with the highest MSE and the shortest prediction accuracy. The ESN had the most accurate Lorenz attractor and the longest prediction accuracy, but the Transformers model had a lower MSE. Although the ESN model showed potential with less optimization effort, further research is needed to optimize and compare these models conclusively. The study indicates that the ESN is slightly better than the Transformers model, with the LSTM RNN performing the least effectively.

Future research could explore the NVAR reservoir computing variant and further optimize ESN hyperparameters. For the Transformers model, experimenting with different positional embeddings and self-attention mechanisms could be beneficial. This could help determine the full potential of both the ESN and Transformers models in predicting chaotic systems.

Appendix

A Predicted trajectories

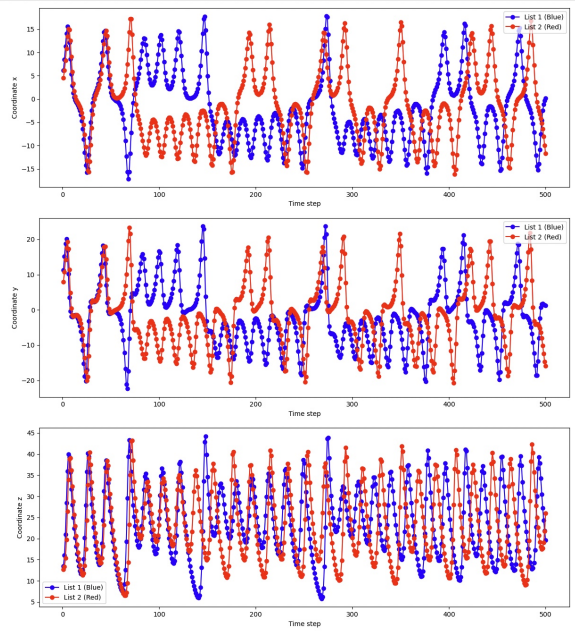
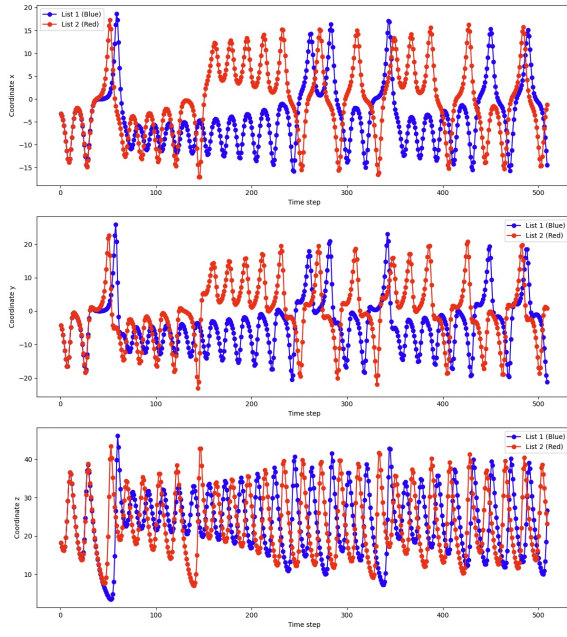


Figure 5: LSTM RNN path (true path is red, model path is blue)

Figure 6: ESN path (true path is red, model path is blue)

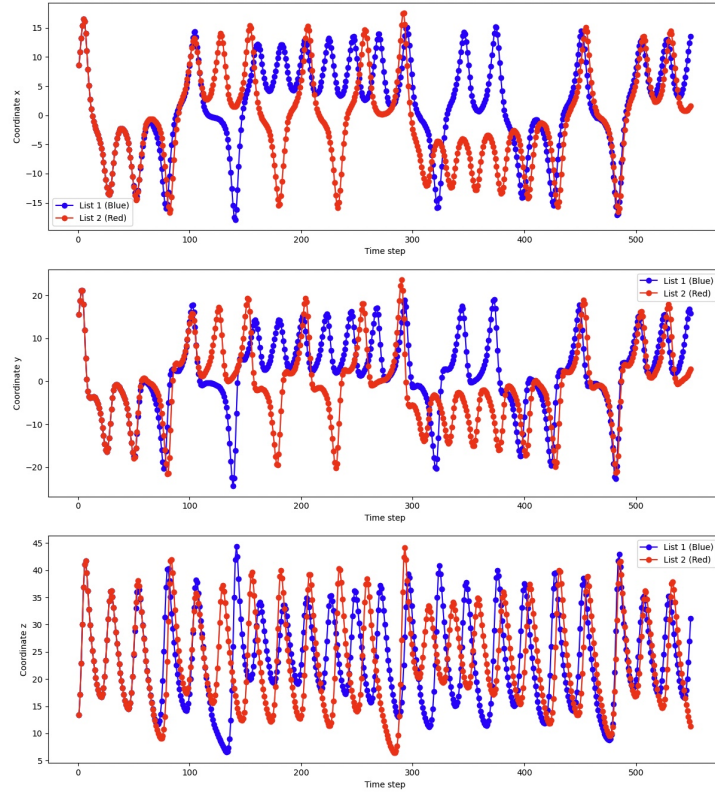


Figure 7: Transformers path (true path is red, model path is blue)