DSML Winter-Semester 2024

Final Project Report

Heyuan Chi (4742393)

February 11, 2025

This report explore the use of Transformer models for predicting two dynamic systems, Lorenz-63 and Lorenz-96, under noisy data conditions. Our experiments compare mean squared errors and power spectral densities between predicted and ground truth trajectories, illustrating that Transformers can capture the main dynamical features, particularly in short-horizon forecasts. Despite increasing prediction errors over extended ranges our results show strong alignment in the dominant frequency components. We also conduct repeated training to assess model robustness, finding that although random initializations produce small variations, the underlying learned structures are generally consistent. All source code and additional information can be found at `https://github.com/HeyuanChi/DSML_final.git`.

Contents

1 Introduction

In the lecture, we learned the basics of dynamical systems theory and looked at how neural networks can solve ODE systems. For noisy, chaotic systems, traditional recurrent neural networks often struggle with issues like vanishing gradients and capturing long-term dependencies. Inspired by the paper **Deep Transformer Models for Time Series Forecasting: The Influenza Prevalence Case**, we decided to use Transformer models to predict dynamical systems.

The Transformer model, built on self-attention, excels at handling long sequences. Unlike RNNs, which process data step by step, Transformers use multi-head attention to pick up global and local patterns, making them scalable and fast—especially with GPU acceleration.

In this project, we apply the Transformer to reconstruct Lorenz systems (Lorenz-63 and Lorenz-96). We train on data with 5% noise and test on clean data by comparing the power spectral density (PSD) of the true and predicted sequences.

This report is organized as follows: Section 2 reviews the chosen paper; Section 3 explains our model implementation; Section 4 presents our experiments and results; and Section 5 concludes with a summary.

2 Chosen Paper Overview

The paper focuses on forecasting influenza-like illness (ILI) time series data. It starts by reviewing older methods such as state space models, compartmental models (SIR), traditional time series models (AR, ARIMA), and sequence models (RNN, LSTM). The authors propose using the Transformer model which is originally developed for natural language processing for time series prediction. They expect this approach to better capture long-term dependencies and combine different types of information.

The main contribution of the paper is adapting the Transformer for time series forecasting. The authors follow the original Transformer design and test it on CDC ILI data from 2010 to 2018. With its self-attention mechanism, the Transformer can compute dependencies across the entire time series at once, helping it capture long-term patterns and periodic features. Moreover, since it can process data in parallel (unlike RNN), training and inference are faster with GPU support.

The paper makes only a few small changes to the original Transformer. The model uses an encoder–decoder structure with these key components:

1. Encoder: A linear input layer with positional encoding, followed by several encoder layers. Each encoder layer includes a self-attention module and a feed-forward module, along with layer normalization.

2. Decoder: A linear input layer, followed by several decoder layers and a final linear mapping layer. Each decoder layer features a self-attention module, a feed-forward module, and an extra encoder–decoder attention module to combine the encoder's output with the decoder's state. Masking is applied to ensure that predictions for future time points only rely on past data. The final softmax layer is removed since the task is regression, not classification.

I have a couple of concerns with this design. Including the linear input layer, positional encoding, and linear mapping layer within the encoder/decoder might confuse readers, as it deviates from the original model. Also, not adding positional encoding after the decoder's linear input layer seems questionable.

For their experiments, the authors tuned the model's hyperparameters based on the data and kept the model setup simple. They ran three experiments:

1. Predicting the ILI ratio one week ahead using 10 weeks of historical data.
2. Forecasting using multivariate time series data (feature vectors).
3. Forecasting with time delay embedding.

Performance was measured using Pearson correlation and RMSE, and the Transformer was compared against models like ARIMA, LSTM, and an attention-based Seq2Seq model. For one-step prediction, the Transformer achieved better Pearson correlation and lower RMSE than traditional models, especially with larger datasets or longer time spans. Adding extra features improved accuracy slightly, but not dramatically. They also found that a time delay embedding dimension of around 8 produced the best RMSE, which aligns with typical experiences in dynamical systems.

In summary, the paper demonstrates that the Transformer is a promising tool for forecasting time series data. Its main contributions include showing that self-attention works well for long-range time series and offering a framework that can be used for modeling nonlinear dynamical systems.

3 Method

3.1 Model and Parameters

Our model closely follows the design in the chosen paper, with one main difference: we add positional encoding right after the decoder's input layer, matching the original Transformer setup. This small adjustment ensures that the decoder can effectively leverage the temporal order of the inputs.

Since the Lorenz systems we study have three state variables, we set the input and output dimensions to 3 (Lorenz-63) and 20 (Lorenz-96) . A linear layer maps the three-dimensional input to a *d_model*-dimensional vector, which then goes through the Transformer blocks. The model uses multi-head attention with *nhead* heads, and it has *num_layers* layers in both the encoder and decoder. Inside each layer, a feed-forward network of dimension *dim_feedforward* processes the attention outputs.

3.2 Evaluation

For training and basic performance checks, we use the mean squared error (MSE) between the predicted and true trajectories. However, to better capture the behavior of chaotic dynamical systems, we also compare the power spectral density (PSD) of the predicted and true sequences. Specifically, we take the Fourier transform of each sequence, square its magnitude to obtain the PSD, and then apply a smoothing function with a chosen parameter σ to handle noisy or highly oscillatory signals. By examining the PSD, we can see if the model matches the dominant frequencies and overall spectral characteristics of the true system, providing a more comprehensive assessment than MSE alone.

4 Training

4.1 Training Data and Batching

We trained our model on time series data from two well-known dynamical systems: Lorenz-63 and Lorenz-96. For Lorenz-63, the data is three-dimensional, while for Lorenz-96, it is 20-dimensional. Each training set contains 100,000 time steps with 5% noise added. The test sets have the same dimensions but are noise-free.

During training, we feed sequences of length n (i.e., $\mathbf{x}_1, \ldots, \mathbf{x}_n$) as input to the encoder. The subsequent m steps ($\mathbf{x}_n, \ldots, \mathbf{x}_{n+m-1}$) go to the decoder, and the model predicts the next m steps ($\mathbf{x}_{n+1}, \ldots, \mathbf{x}_{n+m}$). In each epoch, we randomly select N segments (batch size) of n-step inputs plus m-step decoder inputs from the training set, ensuring that each mini-batch spans continuous segments of the time series.

4.2 Hardware and Schedule

All experiments were run on one machine with one NVIDIA A800 GPU with 80 GB of memory. We train each model for 5,000 epochs, which takes about 10 minutes. During inference, generating a 10-step prediction sequence takes approximately 0.1 seconds.

4.3 Optimizer and Regularization

We follow the optimizer settings from the chosen paper and the orignal paper, using the Adam optimizer with $\beta_1 = 0.8$, $\beta_2 = 0.98$, and a warmup stage of 200 steps. To reduce overfitting, we apply dropout at a rate of *dropout* after each sublayer in both the encoder and decoder. This regularization approach helps stabilize training and prevents the model from memorizing noise in the data.

5 Results

5.1 Lorenz-63 System

In this subsection, we present our experiments on the Lorenz-63 system using various Transformer-based architectures. We keep the same training procedure and data settings as described earlier, with a batch size of 512, an input sequence length of $n = 200$, and a prediction horizon of $m = 10$. After training each model, we evaluate its performance on 1000 randomly selected test sequences, each requiring a 500-step forecast. We then compute three main metrics:

1. MSE_10: The mean squared error over the first 10 prediction steps.

2. MSE_500: The mean squared error over all 500 prediction steps.
3. PSE: The power spectral error, computed by comparing the PSD of the predicted trajectories with that of the ground truth.

Hyperparameters. We begin with base parameters (*base*), then vary one parameter at a time to assess its impact on the predictive performance. Table **??** summarizes the settings and results. All rows other than *base* list only the changed parameter(s); all unspecified parameters remain the same as the base.

Model	d_{model}	n_{head}	n_{layer}	dim_{ff}	$P_{dropout}$	MSE_{10}	MSE_{500}	PSE
base	16	16	3	64	0.1	**0.0003**	**0.6834**	**0.0426**
(A)	32					0.0007	0.7785	0.0485
(B)		4				0.0005	0.9987	0.0524
		8				0.0004	1.1538	0.0566
(C)			2			0.0011	1.3282	0.0675
			4			0.0009	0.8563	0.0481
(D)				32		0.0005	1.0387	0.0513
				128		0.0005	1.2484	0.0602
(E)					0.0	0.0013	1.1301	0.0642
					0.2	0.0007	1.0069	0.0510

Table 1: Hyperparameter settings for the Lorenz-63 system. *base* indicates our initial parameters. (A) to (E) indicate our test parameters of changing one different parameter. MSE_10 is the mean squared error over the first 10 steps, MSE_500 is the mean squared error over 500 steps, and PSE is the power spectral error. All values are rounded to four decimal places.

Based on the metrics, the *base* model stands out as one of the best-performing configurations. In particular, its balance of *d_model*, *nhead*, *num_layers*, and moderate *dropout* appears well-suited to capturing the chaotic dynamics of the Lorenz-63 system.

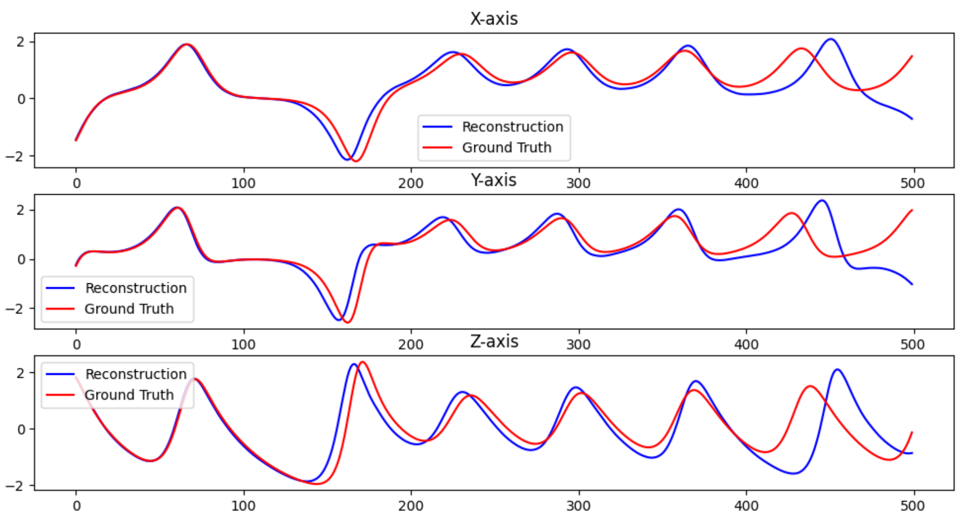


Figure 1: Comparison of predicted vs. true time series for each dimension (X, Y, Z) of the Lorenz-63 system. The solid blue curves represent the ground truth, while the dashed red curves denote the model predictions.

We also provide further visual evidence of how the best model tracks the Lorenz-63 trajectory.

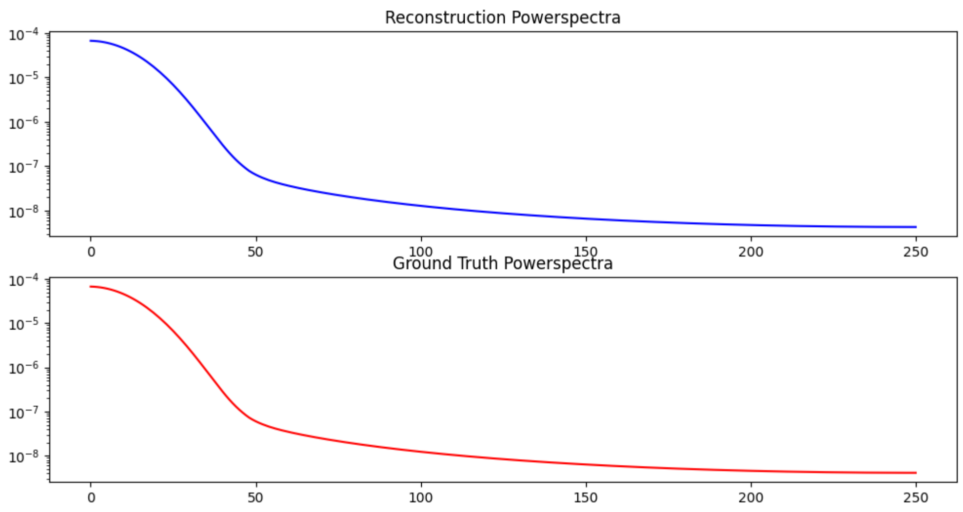


Figure 2: Power spectral density (PSD) comparison between the ground truth and the Transformer-based model for a selected test trajectory.

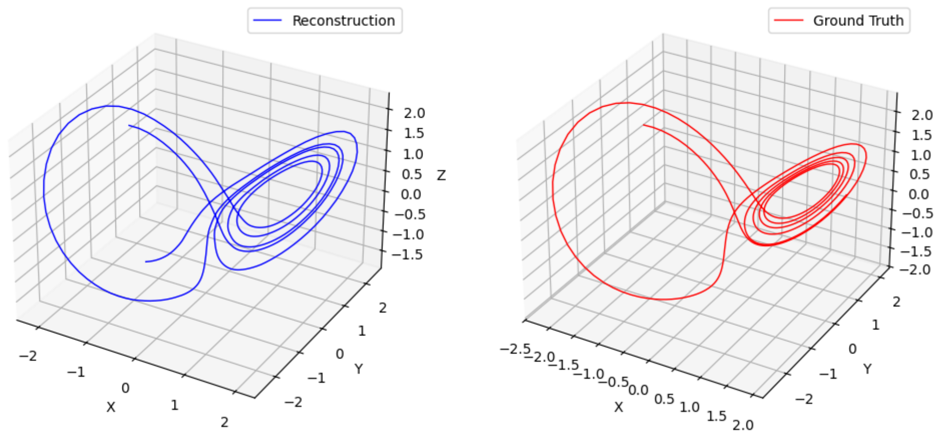


Figure 3: 3D phase-space comparison of the Lorenz-63 attractor.

We plot the predicted vs. true time series for each state dimension (x, y, z) in Figure 1, compare their power spectra in Figure 2, and examine the corresponding attractors in 3D phase space in Figure 3 to gauge the model's ability to replicate chaotic behavior over extended horizons.

From the results shown in Figures 1–3, the model demonstrates highly accurate short-horizon forecasts, with minimal deviation from the ground truth up to around 100 steps. Beyond that point, prediction errors gradually increase due to the accumulation of small discrepancies, yet the power spectral analysis remains largely consistent with the true system. Furthermore, the 3D phase-space plots indicate that the Transformer is capable of reconstructing the characteristic double-scroll attractor of the Lorenz-63 system, even though certain deviations appear over longer prediction horizons.

5.2 Lorenz-96 System

In this subsection, we extend our experiments to the Lorenz-96 system, which consists of 20 state variables. We use the similar training setup as for Lorenz-63 except of $n = 80$. The system's higher dimensionality makes the learning task more challenging. As before, we evaluate each model on 1000 randomly chosen test sequences, forecasting 500 steps ahead and measuring both mean squared errors and power spectral errors. Table 2 summarizes the key hyperparameter settings and their performance.

Table 2: Hyperparameter settings for the Lorenz-96 system.

Model	d_{model}	n_{head}	n_{layers}	dim_{ff}	$dropout$	MSE_{10}	MSE_{500}	PSE
base	256	8	8	1024	0.1	**0.0041**	**1.6629**	**0.1466**
(A)	128					0.0044	1.7491	0.1484
(B)		16				0.0053	1.7598	0.1484
		32				0.0140	1.8382	0.1496
(C)			16			0.6249	1.8736	0.2234
(D)				2048		0.0981	2.0240	0.1821
(E)					0.0	0.0042	1.7874	0.1479
					0.4	0.0049	1.7685	0.1490

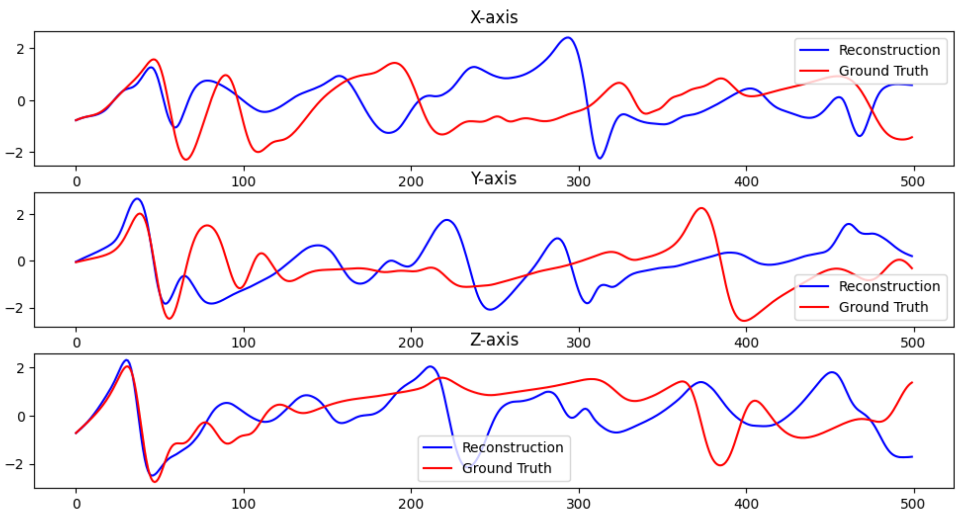


Figure 4: Comparison of predicted vs. true time series for three representative dimensions of the Lorenz-96 system.

Figure 4 contrasts the predicted versus true trajectories for three representative dimensions (out of 20) from a selected test sequence. Although the model captures short-horizon behavior reasonably well, the accumulated errors become more noticeable after approximately 50 steps, worse than the Lorenz-63 case. Nevertheless, the comparison of power spectral densities in Figure 5 suggests that the principal frequency components remain well-aligned with the ground truth. To visualize the attractor structure, Figure 6 shows a 3D phase-space projection using the first three variables which is worse than Lorenz-63 case.

5.3 Repeated Test

To further assess the stability and consistency of our Transformer-based approach, we implemented a routine that trains 20 separate models on each dataset—using different random seeds or initial conditions for each trial—and then computed the average power spectrum distance across these 20 runs. Specifically, we repeated the same training procedure for the Lorenz-63 and Lorenz-96 systems, obtaining 20 independently trained models for each.

After gathering the predictions from all 20 models, we calculated the power spectral density

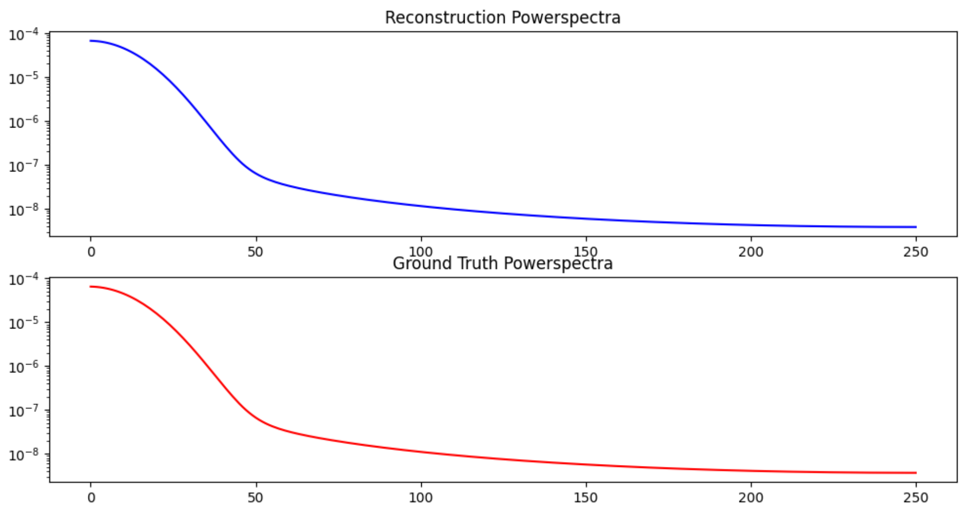


Figure 5: Power spectral density (PSD) comparison between the ground truth and the model for Lorenz-96

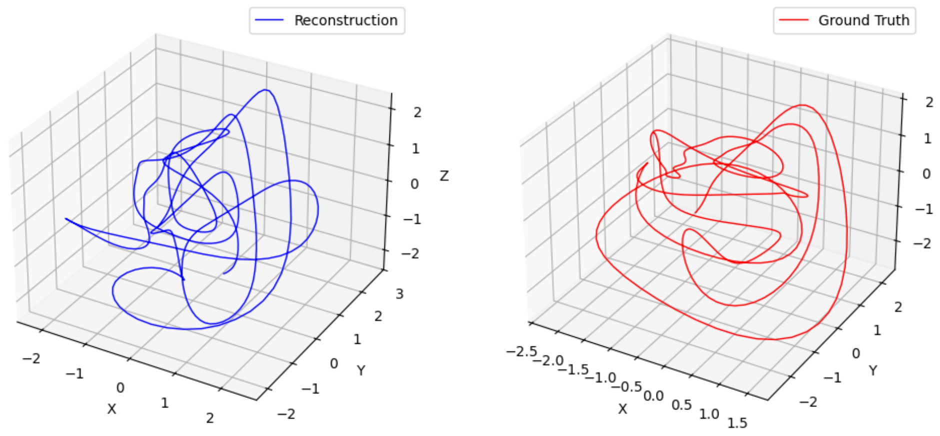


Figure 6: 3D phase-space projection of the Lorenz-96 attractor using the first three dimensions.

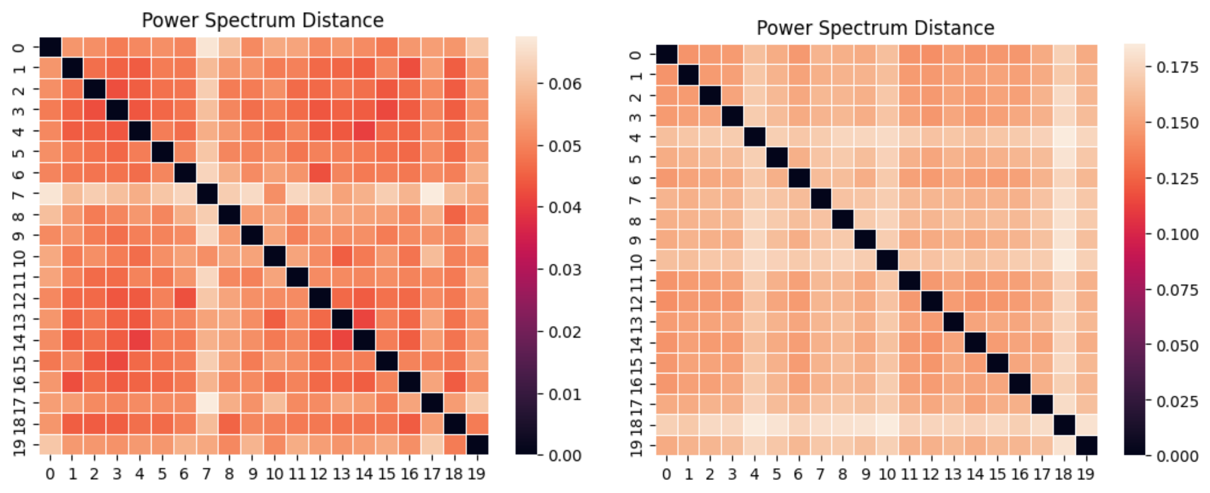
(PSD) for each model's forecasted trajectory and compared it to the true PSD. We then averaged the resulting power spectral errors (PSE) across the 20 models. Table 3 shows the mean and standard deviation of the PSE for both the Lorenz-63 and Lorenz-96 systems.

Table 3: Average PSE over 20 independently trained models. The standard deviation provides a measure of robustness across different random initializations.

System	**Mean**	**Std**
Lorenz-63	0.0509	0.0050
Lorenz-96	0.1597	0.0095

We can further examine the model-to-model PSE distances via heatmaps. Figure 7 show the pairwise power spectrum distance between all pairs of models, where each cell represents the difference in PSD between two distinct runs.

From Figures 7a and 7b, the diagonal elements are naturally zero because each model compared with itself yields no difference in the predicted PSD. Off-diagonal values show how the random initializations lead to slightly different solutions: for the Lorenz-63 models, most pairwise distances remain under 0.06, confirming the moderate variation indicated by the standard deviation



(a) Lorenz-63 (b) Lorenz-96

Figure 7: Pairwise power spectrum distance among the 20 independently trained models for (a) Lorenz-63 and (b) Lorenz-96. The diagonal cells are each model compared to itself (distance = 0), while the off-diagonal cells illustrate how different random seeds can yield slightly varied spectral predictions.

in Table 3. By contrast, the Lorenz-96 heatmap shows generally higher distances (ranging up to about 0.18), which aligns with the higher mean and standard deviation of PSE in the 20-model ensemble. Notably, certain "clusters" of seeds appear, reflecting the possibility that some initializations converge to similar local minima, while others diverge slightly. Nevertheless, the overall heatmap patterns suggest that the models collectively learn consistent spectral characteristics for both Lorenz-63 and Lorenz-96, despite randomization in initialization.

6 Conclusion

In summary, we have shown that Transformer-based models can effectively learn and predict both the Lorenz-63 and Lorenz-96 systems. Our experiments demonstrate that the self-attention mechanism captures the primary dynamics of chaotic systems, as evidenced by strong short-horizon accuracy and good alignment in power spectral densities. Nevertheless, error accumulation remains evident when predicting over longer time spans, particularly for the higher-dimensional Lorenz-96 system, highlighting the inherent difficulty of forecasting chaotic behavior.

Additionally, our repeated tests indicate that, despite variations in initialization, the learned representations are relatively robust: the average power spectrum error exhibits moderate variance across multiple training runs. Overall, these findings affirm the viability of Transformers for modeling nonlinear dynamical systems.