DSML Winter-Semester 2024

Final Project Report

Heyuan Chi (4742393)

February 11, 2025

In this report, we use the Transformer model to fit the Lorenz-63 and Lorenz-96 systems. We compare mean squared errors and distances between power spectral densities of true and predicted sequences. In experiments, we find it can capture the features in short period. However, the errors will accumulate forecast long periods. We also conduct repeated training to assess model robustness. We train models with multiple random initializations and the results are similar. All source code and additional information can be found at `https://github.com/HeyuanChi/DSML_final.git`.

I would like to request a **grade** for this course. According to the grading policy for individual submissions (50%), I have selected the following five exercise sheets to be graded: Sheet 2, Sheet 3, Sheet 4, Sheet 7 and Sheet 8. Please consider these sheets along with my final project for the final grade evaluation. If any further information is required, please feel free to contact me.

Contents

1 Introduction **1**

2 Chosen Paper Overview **1**

3 Method **3**
3.1 Model and Parameters 3
3.2 Evaluation 3

4 Training **3**
4.1 Training Data and Batching 3
4.2 Hardware and Schedule 4
4.3 Optimizer and Regularization 4

5 Results **4**
5.1 Lorenz-63 System 4
5.2 Lorenz-96 System 6
5.3 Repeated Test 7

6 Conclusion **9**

1 Introduction

In the lecture, we learned the basic knowledges of dynamical systems theory and learned how we can use neural networks to solve ODE systems. For noisy and chaotic systems, recurrent neural networks are not very good at capturing long-term features. And we may meet difficults of vanishing gradients. Based on the paper Deep Transformer Models for Time Series Forecasting: The Influenza Prevalence Case, we decided to use Transformer models to fit dynamical systems.

The Transformer model is good at processing long sequences. It use multi-head attention to capture global and local features. This is different from RNN model, which is processing data step by step. Such a structure is more scalable and fast, and can also be accelerated by GPU parallelization.

We apply the Transformer to Lorenz systems (Lorenz-63 and Lorenz-96). We train on data with 5% noise and test on clean data. We choose to use MSE loss and Adam optimizer in traing. Then we calculate the distances between power spectral density of the true and predicted sequences.

2 Chosen Paper Overview

The paper is mainly about forecasting influenza-like illness (ILI) time series data. It first introduced older methods like state space models, compartmental models (SIR), traditional time series models (AR, ARIMA), and sequence models (RNN, LSTM). Then the authors used the Transformer model to predict time series. They thought this model can better capture long-term features and understand different types of information all together.

The main idea of the paper is to adapt the Transformer for time series forecasting. The authors built the modle with the structure in the original pape and made some small changes to suit the purpose. Then they train and test the model on CDC ILI data from 2010 to 2018. In experiments, they found that Transformer can capture dependencies across the whole time series at once with the self-attention mechanism. This is the reason why it can learn long-term trends and repeated features. And because it can run parts of the data in parallel (unlike RNN), training and inference are faster with a GPU.

The paper only made a few changes to the original Transformer. The model still has an encoder–decoder structure with key parts:

1. Encoder: A inear input layer with positional encoding, and several encoder layers. Each encoder layer has a self-attention module and a feed-forward module, plus batch normalization.

2. Decoder: A linear input layer, several decoder layers, and a linear mapping (output) layer. Each decoder layer has a self-attention module, a feed-forward module, and another encoder–decoder attention module. The encoder–decoder attention module uses the encoder output along with the decoder's state. It uses Masks to make sure not using future time points to predict, but only depend on past data. The final softmax layer is removed because the it is a

regression model, not a classification model.

I find some possible problems with this design. It's strange to put the linear input layer, the positional coding, and the linear mapping layer all inside the encoder or decoder. It is different from the original model and may let the readers confuse. And it is strange that they did not use positional coding after the linear input layer in the decoder.

The authors designed simple and straightforward experiments. Their aim is mainly measuring how well they used the Transformer to predict ILI data. They did three experiments as showing below:

1. Predictions after one week using only ILI data: The first experiment was basic. They just used the Transformer model and tried to predict the influenza index (ILI ratio) for the next week from the last 10 weeks data. To see how effective Transformer really is, they also specifically compared it with ARIMA, LSTM, and the Seq2Seq model with an ATTENTION mechanism. On the evaluation method, they used two indicators, one is the Pearson correlation coefficient, and the other is the commonly used RMSE (root mean square error). From the experimental results, Transformer really performs the best. The prediction accuracy is slightly higher than other deep models. And the error is also significantly lower than other methods. For example, the RMSE is about 42% lower than ARIMA, 27% lower than LSTM, and about 8% lower than Seq2Seq with ATTENTION. The authors feel that Transformer performs so well, most likely because its self-attention mechanism is able to capture those complex dynamic relationships in the data more efficiently, whereas the normal attention mechanism is a bit weaker in this regard.

2. Inclusion of multivariate data for prediction: The authors of the second experiment thought that using only ILI data might not be enough, and they tried to add features like 'week number', which can reflect seasonal changes, as well as the first-order and second-order differences of the data, in the hope of improving the prediction accuracy a little bit more. But the result was a bit unexpected: after adding these new features, the prediction effect was only a little bit better, almost the same as without. This may indicate that the Transformer itself has already learnt enough information through its powerful attention mechanism, and the additional data is actually a bit too much.

3. Prediction with Time Delay Embedding (TDE): The third experiment is designed to test if the method Time Delay Embedding could be used to improve model. TDE is expanding the original data to higher dimensions so that the model can capture hidden dynamics in data more easily. In the experiments, they tried different dimensions. They found that the best prediction results were obtained when the dimensionality was around 8. The lowest RMSE was around 0.6. They also mention that the result is similar with some other studies on epidemic prediction. Those previous studies usually found the best dimensions were around 5 to 7 as well.

These three experiments' results show that the Transformer model is very good at predicting time series. The self-attention mechanism is especially good at finding meaningful features in data. It did not improve the results very much when adding more features to the data. Although it did help a little.

The result of the TDE experiments were positive as well, very similar to others. We can conclude that Transformer is good at learning deep features. It also shows that self-attention works well and can help model nonlinear dynamic systems.

3 Method

3.1 Model and Parameters

Our model basically follows the design in the chosen paper. But we did one small change: we add positional encoding after the decoder's input layer, like the original Transformer. It ensures that the decoder can effectively use the temporal order of the inputs.

We set the dimensions of input and output both to the dimension of the original data (3 for Lorenz-63 and 20 for Lorenz-96). The input linear layer maps the input to a *d_model*-dimensional vector. Then it goes through the Transformer blocks. The model uses multi-head attention with *nhead* heads. At has *num_layers* layers in both the encoder and decoder. A feed-forward network of dimension *dim_feedforward* processes the attention outputs in each layer.

3.2 Evaluation

We use the mean squared error (MSE) between the predicted and true trajectories in training, and for checking basic performances. In addition, we also compare the power spectral density (PSD) of sequences to better evaluate the results. In detail, we first do Fourier transformation on each sequence. Then we calculate the square of the amplitude to get the PSD. And then we process the signal by a smoothing function with the chosen parameter σ. We can use the PSD to see if the model matches the dominant frequencies and overall spectral characteristics of the real system. We can also calculate the PSD distance. This is better than using MSE alone.

4 Training

4.1 Training Data and Batching

We trained our model on time series data from the two dynamical systems Lorenz-63 and Lorenz-96. For Lorenz-63, the data is 3-dimensional. And 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. We use the Adam optimizer with $\beta_1 = 0.8$, $\beta_2 = 0.98$, and a warmup stage of 200 steps. We also apply dropout at a rate of *dropout* after each sublayer in both the encoder and decoder to reduce overfitting. 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.

We begin with base parameters (*base*), then vary one parameter at a time to assess its impact on the predictive performance. Table 1 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.

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.

We also provide further visual evidence of how the best model tracks the Lorenz-63 trajectory. We plot the predicted vs. true time series for each state dimension

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.

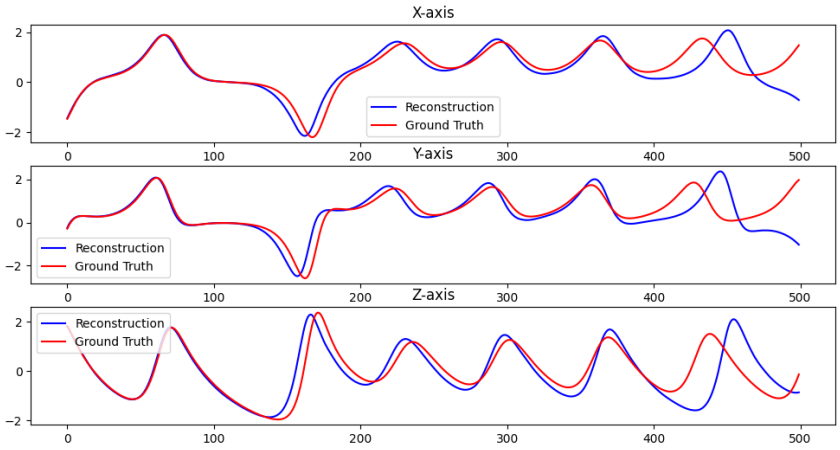


Figure 1: Comparison of predicted vs. true time series for each dimension (X, Y, Z) of the Lorenz-63 system. The solid blue curves represent the ground truth, while the dashed red curves denote the model predictions.

(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.

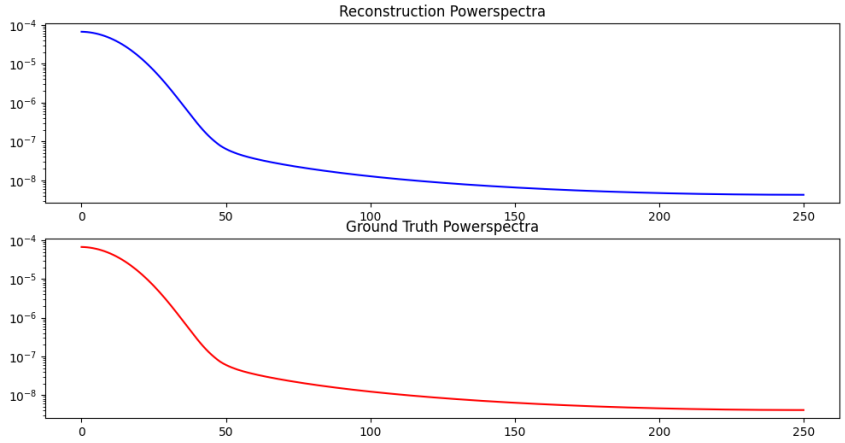


Figure 2: Power spectral density (PSD) comparison between the ground truth and the Transformer-based model for a selected test trajectory.

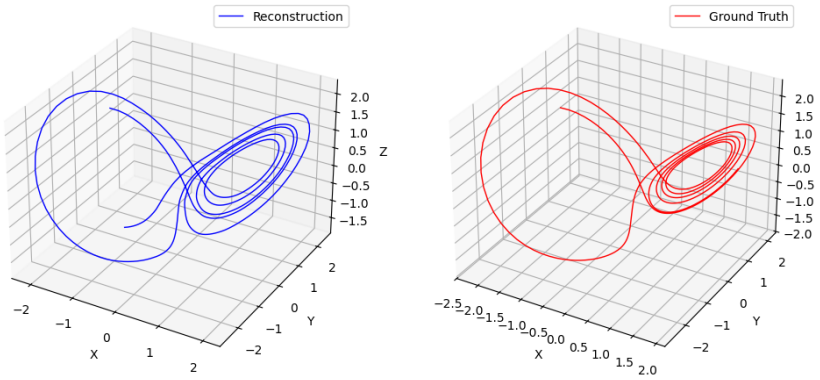


Figure 3: 3D phase-space comparison of the Lorenz-63 attractor.

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.

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.

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

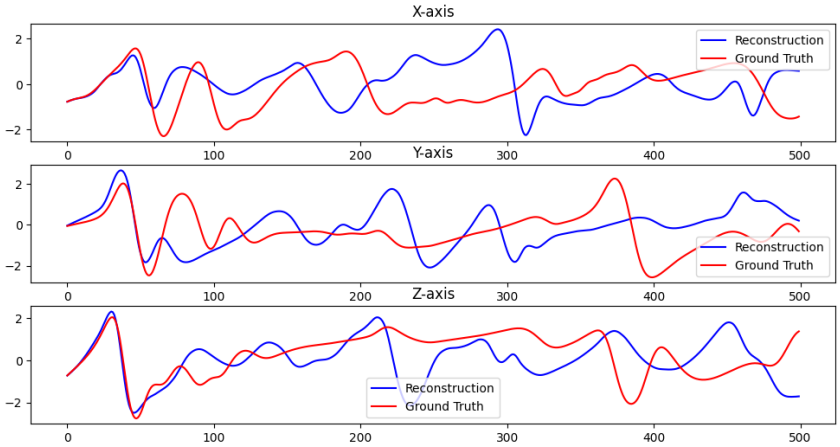


Figure 4: Comparison of predicted vs. true time series for three representative dimensions of the Lorenz-96 system.

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 (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

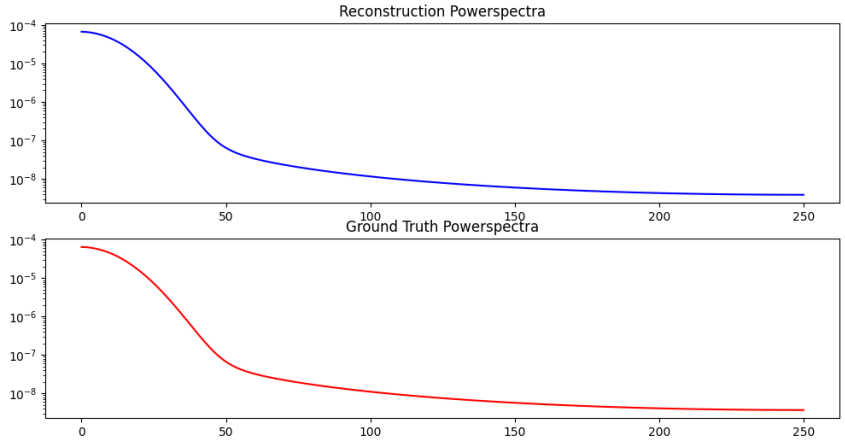


Figure 5: Power spectral density (PSD) comparison between the ground truth and the model for Lorenz-96

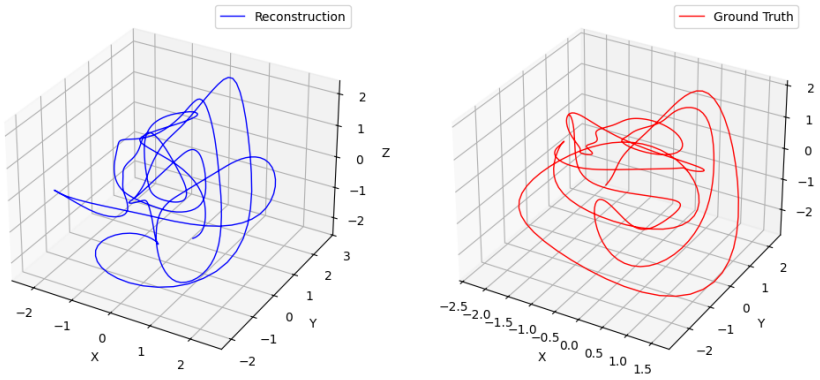
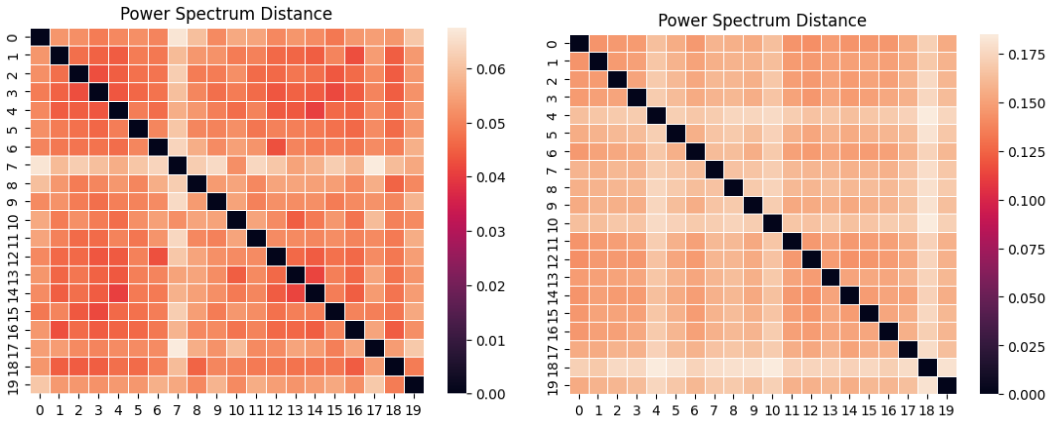


Figure 6: 3D phase-space projection of the Lorenz-96 attractor using the first three dimensions.

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 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.



(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.

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.