Widening the Time Horizon: Predicting the Long-Term Behavior of Chaotic Systems

Yong Zhuang[1*], Matthew Almeida[1*], Wei Ding*, Patrick D Flynn*, Shafiqul Islam† and Ping Chen‡
*Department of Computer Science, University of Massachusetts Boston
Email: {yong.zhuang001, matthew.almeida001, wei.ding, patrick.flynn003}@umb.edu
†Department of Civil and Environmental Engineering, Tufts University
Email: Shafiqul.Islam@tufts.edu
‡Department of Engineering, University of Massachusetts Boston
Email: ping.chen@umb.edu

***Abstract*—The understanding of chaotic systems is challenging not only for theoretical research but also for many important applications. Chaotic behavior is found in many nonlinear dynamical systems, such as those found in climate dynamics, weather, the stock market, and the space-time dynamics of virus spread. A reliable solution for these systems must handle their complex space-time dynamics and sensitive dependence on initial conditions. We develop a deep learning framework to push the time horizon at which reliable predictions can be made further into the future by better evaluating the consequences of local errors when modeling nonlinear systems. Our approach observes the future trajectories of initial errors at a time horizon to model the evolution of the loss to that point with two major components: 1) a recurrent architecture, Error Trajectory Tracing, that is designed to trace the trajectories of predictive errors through phase space, and 2) a training regime, Horizon Forcing, that pushes the model's focus out to a predetermined time horizon. We validate our method on classic chaotic systems and real-world time series prediction tasks with chaotic characteristics, and show that our approach outperforms the current state-of-the-art methods.**

***Index Terms*—long-term prediction, chaotic system, sequential data, dynamical system.**

I. Introduction

In many physical, biological, and human systems the governing equations are known with high confidence, but the analytic solutions may not exist and reliable numerical solutions are often prohibitively expensive because of nonlinearity, feedback, and sensitive dependence on initial conditions [1], [2]. Developing reliable numerical solutions that integrate short length and fast time scales is a long-standing problem. As this class of governing equations arises in many varied applications, means to improve the ability to make meaningful predictions of their future states are of great practical importance.

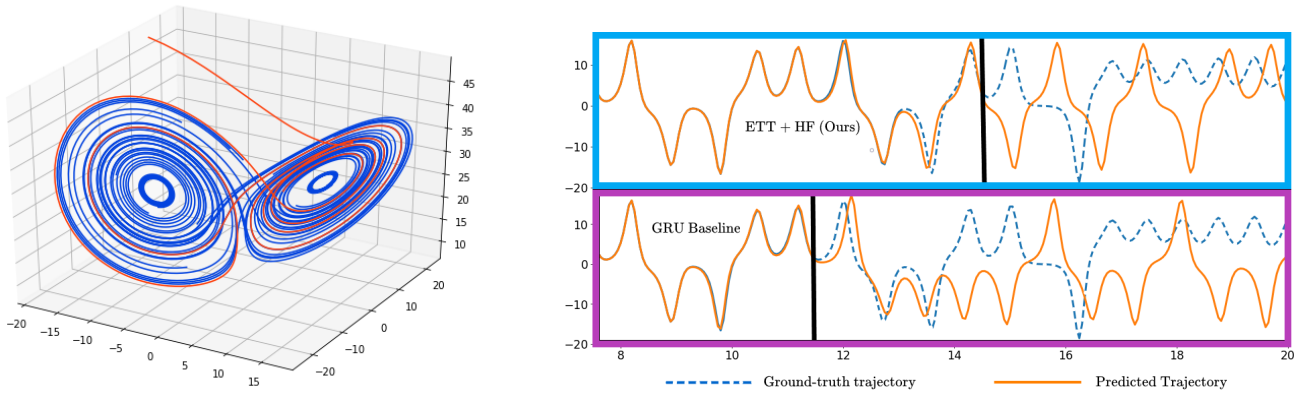
Moreover, many of these highly complex dynamic systems exhibit chaotic behaviors that are highly sensitive to initial conditions. A small observational error—even truncation error caused by binary representation of values tens of digits past the decimal—can grow exponentially in time. As Lorenz [3] aptly noted, for this class of nonlinear systems, approximately close present states do not necessarily map to approximately close states in the future. More importantly, a small predictive error (from a point in a system's phase space, $\mathbf{u_{t_0}}$ to $\mathbf{u_{t_0}} + \mathbf{e}$, where $\mathbf{e}$ is a small error vector) at a time step t_0 may result in exponentially larger future error than that resulting from a larger initial error (in magnitude) in another direction (from $\mathbf{u_{t_0}}$ to $\mathbf{u_{t_0}} - 2\mathbf{e}$, for example) at the same time step.

Machine learning have been widely applied in various areas [4], [5]. In the last few decades, some Machine learning methods have been proposed for long-term sequence prediction [6], [7], [8], [9]. However, reliable prediction of state past a certain time horizon remains extremely challenging, depending on the parameterization of the system and the lead time of the prediction. Our research goal is to push out the time horizon at which reliable predictions can be made as far as possible utilizing new developments from machine learning , and our study has obtained promising results as shown in Fig 1. More specifically, while there can be theoretical bounds for the accuracy of future prediction given an initial error of some size for certain systems (e.g. [1] gives an analysis for the Lorenz '63 system), important performance gain is possible to achieve by training a model to avoid local mistakes that result in dramatic changes to future phase space trajectories. In doing so, we avoid the greatest sources of error, and keep the predicted trajectories as close to the ground truth as possible and the predictions relevant in practice. Any such progress could represent a significant step forward in real-world problem domains.

In this work, we present a new deep learning method for prediction in chaotic systems: it takes the form of a recurrent architecture set up for error trajectory tracing and an accompanying training regime, Horizon Forcing, which allows a neural network to both model the system's state transition function and use it to trace the evolution of its mistakes. By using both in tandem a model is able to properly evaluate the consequences of local mistakes as the underlying system evolves toward the time horizon. We make the following contributions:

- We introduce a new recurrent architecture for error trajectory tracing (ETT) in chaotic systems. It is designed to improve the prediction of such systems by allowing the model to optimize its parameters based on the true,

[1]Equal contribution



(a) Lorenz '63 system, a classic example of chaotic systems

(b) Ground-truth and predicted trajectories

Fig. 1: (a) The Lorenz attractor, an exemplary chaotic system, with an example trajectory pictured in orange. Trajectories move quickly from their initial (randomly initialized) state to the attractor and follow a chaotic orbit around two-lobes. (b) Ground-truth (blue) and predicted (orange) trajectories for our Horizon-Forced model (top) and Teacher Forcing baseline (bottom). For readability y-axis is the y-axis of the Lorenz system and the x-axis indicates trajectory steps scaled to Lyapunov time (defined to be the inverse of the largest Lyapunov exponent, the amount of time to accrue error e. In the Lorenz system, this is $\approx \frac{1}{0.906}$, or 1.103).

longer-term compounding consequences of small initial prediction errors; how the trajectories beginning at the predicted states evolve forward to the time horizon.

- We present Horizon Forcing, a new training regime for optimizing our ETT models. We optimize our cost function based on short-term predictions first, then shift focus to the long term as training progresses. By doing this with shared weights between layers, we further improve the single-step error (a proxy for the system's transition function) because minimizing long-term error necessitates controlling short-term error in chaotic systems.
- We extensively validate our method on a well-studied chaotic system with known dynamics and a set of real-world time series prediction tasks.

II. Related Work

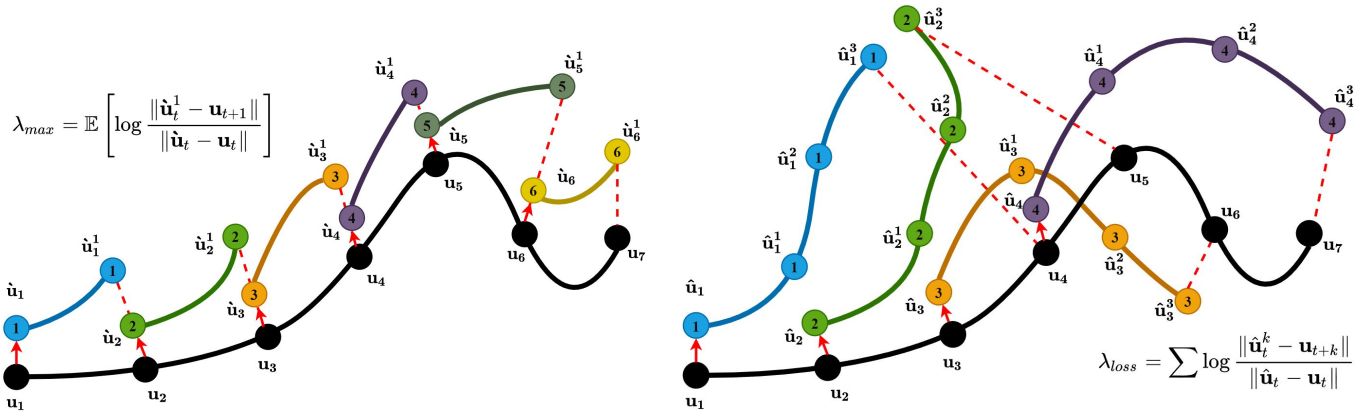
Enormous effort has gone into the understanding and modeling of chaotic nonlinear dynamical systems using observed data [10]. With the explosion of interest in deep learning and proliferation of powerful computing hardware, focus has shifted to the use of powerful models to recreate complex chaotic dynamics. Some recent works [11], [12], [13] directly model the derivatives of the system such that the model learns the vector field over the phase space; other works (including this one) attempt to model trajectories through phase space in a sequence-to-sequence formulation [14], [15]. Below we examine current deep neural methods for chaotic system prediction:

A. Recurrent Approaches

Recurrent Neural Networks (RNNs) have become competitive methods to time series forecasting[12], [15]. They are typically trained with *teacher forcing*, which feed ground-truth values back into the RNN after each step, thus prevents errors that compound over time from harming network convergence while training. However, at inference ground-truth values are not available, so the predictions from previous step must be used, reintroducing the danger of error accumulating over time. Scheduled Sampling [16] proposes to fix this by replacing the teacher-forced ground truth inputs with the model's predicted values with a certain probability. However, [17] show that this sampling method yields a biased estimator - the loss function induced by Scheduled Sampling is not minimized at the true input distribution. In [18], the authors introduce *Professor Forcing*, which uses an adversarial discriminator to minimize the difference between a model being run in teacher-forced mode and free-running (or inference) mode, by using the outputs of each mode and a function of some of the internal states of the model. This method is difficult to train in practice, requiring the optimization of several hyperparameters [19]. Other methods include Zoneout [20], which is a dropout-like training procedure that instead of setting activations to 0, randomly sets a set of activations to be identical to those at the previous time step. This is meant to regularize the transition dynamics; we do not find this to be an appropriate method for a chaotic system because sensitivity is necessary to fit the system's ground truth.

B. Transformer Architectures

Transformer architectures [8] have revolutionized language modeling in recent years [21] and are now being applied to long sequence prediction with success as well. [9]'s music transformer modifies this architecture, introducing an efficient relative attention implementation ($O(N)$) which they apply to a music generation task.



(a) Maximal Lyapunov Exponent

(b) Lyapunov Horizon Loss

Fig. 2: Motivating Figure: Maximal Lyapunov Exponent VS. Lyapunov Horizon Loss. (a) Maximal Lyapunov Exponent: $\grave{\mathbf{u}}$ presents a point generated by a starting point ($\mathbf{u}_t$) with a perturbation (red arrow) and $t \in \{1, 2, 3, 4, 5, 6\}$; (b) Lyapunov Horizon Loss: $\hat{\mathbf{u}}_t$ is a prediction of $\mathbf{u}_t$ with an small error (red arrow). In this example, $t \in \{1, 2, 3, 4\}$ and $k = 3$.

More recently, Informer [22] was introduced explicitly for long sequence forecasting. Informer utilizes a sparse implementation of attention which reduces memory complexity. Informer utilizes a generative decoder to forecast a sequence in parallel in order to avoid the problem of error accumulation.

Autoformer [23] uses an architecture similar to Informer but with a modified efficient autocorrelation-based attention variant and explicitly utilizes properties of timeseries (seasonality and overall temporal trend), achieving similar memory complexity to Informer. These models represent the state-of-the-art for time series forecasting using deep learning, but chaotic systems present unique challenges that our models are designed to overcome.

C. The Broad Learning System and Extreme Learning Machines

In [24], the authors use a manifold constraint to make a Broad Learning System (BLS) [6] model better suited for learning manifolds and reconstructing attractors in chaotic systems. The BLS model is a single-layer architecture designed to learn on a series of "wide" concatenated feature maps over the data, which are themselves mapped to additional *enhancement nodes* via a second mapping function to provide nonlinearity.

Extreme learning machines (ELMs) is another efficient tool for nonlinear dynamic system modeling [7]. It is a single-layer feed-forward architecture where the first weight matrix (which multiplies the input) and biases are initialized randomly and untrained; the post-activation weights are analytically solved by calculating the pseudo inverse of the output matrix. The problem with these two models is their architecture cannot be "deep," which is not able to represent complex dynamics in chaotic systems.

Comparing with these existing methods, our approach has two major advantages: (1) it assumes no prior knowledge of the equations describing the underlying system, and (2) it models how its errors evolve through the system's complex dynamics and accumulate over time. To the best of our knowledge, no other work has attempted (2) without assuming prior knowledge of system dynamics.

III. Methods: Error Trajectory Tracing and Horizon Forcing

A. Problem formulation

Given N sequences states in a chaotic system

$$\mathbb{D} = \left\{\mathbf{u}_{1:T}^{(i)}\right\} = \left\{\left(\mathbf{u}_1^{(i)}, \mathbf{u}_2^{(i)}, ..., \mathbf{u}_t^{(i)}, ..., \mathbf{u}_T^{(i)}\right)\right\}$$

where $i \in 1...N, t \in 1...T$. $\mathbf{u}_t^{(i)}$ presents the t_{th} state of the i_{th} sequence. Each successive state $\mathbf{u}_t^{(i)}$ is a repeated application of a unknown transition function $\mathcal{F}$:

$$\mathbf{u}_t^{(i)} = \mathcal{F}(\mathbf{u}_{t-1}^{(i)})$$

The time series forecasting aims to predict the evolution of the system, represented by the sequence of future states $\mathbf{u}_{T+1:T+Q}^{(i)} = \left(\mathbf{u}_{T+1}^{(i)}, \mathbf{u}_{T+2}^{(i)}, ..., \mathbf{u}_{T+Q}^{(i)}\right)$ given the past $\mathbf{u}_{1:T}^{(i)}$. It is a rolling process, that means when we forecast the future states $\mathbf{u}_{T+Q}^{(i)}$, we forecast one state at a time, and advancing time by one step. A long-term forecasting setting means to predict a long-term future, i.e. a larger Q.

B. Lyapunov horizon loss

In chaos theory, the *maximal Lyapunov exponent* is a measure of the chaotic nature of a system [1], [25]. It can be understood as the expected logarithm of the growth rate of unit errors near a strange attractor (specific to the attractor, which depends on the basin of attraction of initial condition $\mathbf{u}_0$): a negative value of the Lyapunov exponent indicates stability (as the growth rate is not exponential), while a positive exponent indicates exponential increase in small deviations over time.

$$\lambda_{max} = \mathbb{E}\left[\log \frac{\|\grave{\mathbf{u}}_t^1 - \mathbf{u}_{t+1}\|}{\|\grave{\mathbf{u}}_t - \mathbf{u}_t\|}\right] \tag{1}$$

where $\mathbf{u}_t$ and $\grave{\mathbf{u}}_t$ are two nearby points in the phase space of the system (see Fig. 2(a)) and $\mathbf{u}_{t+1}$ and $\grave{\mathbf{u}}_t^1$ are the points they

transition into after one iteration. In practice, an attractor's λ_{max} is estimated numerically by computing a huge number of terms (10^9 or more, in some cases [25]).

λ_{max} gives the degree of chaotic expansion of an entire attractor as the expectation over terms that measure individual local error growth rate. As such, λ_{max} itself is too coarse an estimate for us to use in our modeling process—but the individual terms can provide a valuable measurement of the behavior of the error locally.

We select a time horizon in the near future (by a number of time steps, k) and track how errors at each time step on the trajectory evolve from intermediate states ($\mathbf{u}_t$) to the time horizon ($\mathbf{u}_{t+k}$) (Fig. 2(b)). By modeling how trajectories starting at a predicted state evolve, we can penalize errors that grow into large deviations at the time horizon more harshly than errors of the same magnitude that stay close to the true trajectory by adding a Lyapunov horizon penalty (Eq. 2) into the training loss. This allows our models to not simply account for error k steps ahead, but to optimize using the k-step evolution of 1-step errors. The Lyapunov horizon loss is given by

$$\lambda_t = \log \frac{\|\hat{\mathbf{u}}_t^k - \mathbf{u}_{t+k}\|}{\|\hat{\mathbf{u}}_t - \mathbf{u}_t\|} = \log \frac{\left\|d_t^k\right\|}{\|d_t\|} \tag{2}$$

where d_t^k is the distance from the predicted state to the true state, k steps in the future from a given t.

C. Model architecture: Error Trajectory Tracing

In order to track how errors evolve from intermediate states ($\mathbf{u}_t$) to $\mathbf{u}_{t+k}$ and catch the laypunov horizon loss at time horizon k, we employ deep recurrent neural network cells as building blocks to model the transition function $\mathcal{F}$, and design a tower architecture as shown in Fig 2 to model the future evolution of the chaotic system from the prediction made after a single step. In such a way that we can calculate and backpropagate the Lyapunov horizon loss. We incorporate the Lyapunov horizon loss with squared error as overall loss function, then the network parameters θ can be optimized as shown in Eq. 3.

We tried different recurrent cells and empirically find Gated Recurrent Unit (GRU, [26]) to be superior to LSTM and feedforward units across hyperparameter settings, so consider only those models here.

$$\begin{aligned}
\theta^* &= \arg\min_\theta(\mathcal{L} + \lambda_{loss}) \\
&= \arg\min_\theta \sum_{i=1}^{N} \left[\sum_{t=1}^{T} (\hat{\mathbf{u}}_t^{(i)} - \mathbf{u}_t^{(i)})^2 + \sum_{t=1}^{T-k} \lambda_t^{(i)} \right] \\
&= \arg\min_\theta \sum_{i=1}^{N} \left[\sum_{t=1}^{T} (\hat{\mathbf{u}}_t^{(i)} - \mathbf{u}_t^{(i)})^2 + \sum_{t=1}^{T-k} \log \frac{\|\hat{\mathbf{u}}_t^{(i),k} - \mathbf{u}_{t+k}^{(i)}\|}{\|\hat{\mathbf{u}}_t^{(i)} - \mathbf{u}_t^{(i)}\|} \right] \\
&= \arg\min_\theta \sum_{i=1}^{N} \left[\sum_{t=1}^{T} (d_t^{(i)})^2 + \sum_{t=1}^{T-k} \log \frac{\left\|d_t^{(i),k}\right\|}{\left\|d_t^{(i)}\right\|} \right]
\end{aligned} \tag{3}$$

Model Training. Recurrent neural networks are usually trained using Teacher Forcing, whereby the model receives the ground truth $\mathbf{u}_t$ as input at time t, thus can help to minimize the one-step prediction error and force the RNN prediction[16]. With Teacher Forcing, a GRU cell can be formulated as:

$$\begin{aligned}
z_t &= \sigma(W_z \mathbf{u}_t + U_z h_{t-1} + b_z) \\
r_t &= \sigma(W_r \mathbf{u}_t + U_r h_{t-1} + b_r) \\
s_t &= \phi_h(W_h \mathbf{u}_t + U_h(r_t \odot h_{t-1}) + b_h) \\
h_t &= z_t \odot h_{t-1} + (1 - z_t) \odot s_t \\
\hat{\mathbf{u}}_{t+1} &= W_u h_t + b_u
\end{aligned} \tag{4}$$

where z_t is the update gate, r_t is the reset gate, and h_t is the hidden state. The parameters $\{W_z, W_r, W_h, W_u, U_z, U_r, U_h, b_z, b_r, b_h, b_u\}$ can be updated using backpropagation. Taking W_z as an example (the process is similar for the others), the updates can be computed as follows:

$$\begin{aligned}
\frac{\partial \mathcal{L}_t}{\partial W_z} &= \frac{\partial \mathcal{L}_t}{\partial h_t} \frac{\partial h_t}{\partial W_z} \\
&= \frac{\partial \mathcal{L}_t}{\partial h_t} \sum_{i=1}^{t} \left(\frac{\partial h_t}{\partial h_i} \frac{\overline{\partial h_i}}{\partial W_z} \right) \\
&= \frac{\partial \mathcal{L}_t}{\partial h_t} \sum_{i=1}^{t} \left(\left(\prod_{j=i}^{t-1} \frac{\partial h_{j+1}}{\partial h_j} \right) \frac{\overline{\partial h_i}}{\partial W_z} \right)
\end{aligned} \tag{5}$$

where $\overline{\partial h_j}/\partial W_z$ is the gradient of ∂h_j with respect to W_z while taking ∂h_{j-1} as a constant. And $\partial h_t / \partial h_{t-1}$ is,

$$\begin{aligned}
\frac{\partial h_t}{\partial h_{t-1}} &= \frac{\partial h_t}{\partial s_t} \frac{\partial s_t}{\partial h_{t-1}} + \frac{\partial h_t}{\partial z_t} \frac{\partial z_t}{\partial h_{t-1}} + \frac{\overline{\partial h_t}}{\partial h_{t-1}} \\
&= \frac{\partial h_t}{\partial s_t} \left(\frac{\partial s_t}{\partial r_t} \frac{\partial r_t}{\partial h_{t-1}} + \frac{\partial s_t}{\partial h_{t-1}} \right) + \frac{\partial h_t}{\partial z_t} \frac{\partial z_t}{\partial h_{t-1}} + \frac{\overline{\partial h_t}}{\partial h_{t-1}}
\end{aligned} \tag{6}$$

Error Trajectory Tracing. To force the GRU cells to learn how the error evolves over time and reduce prediction deviations at later steps, we build the Error Trajectory Tracing architecture as follows (See Fig. 3): first, we build a bottom (zero) layer in the form of a standard GRU RNN. But at each step in this layer, we transfer the hidden state and output to two other cells: the next cell in the zero layer and another

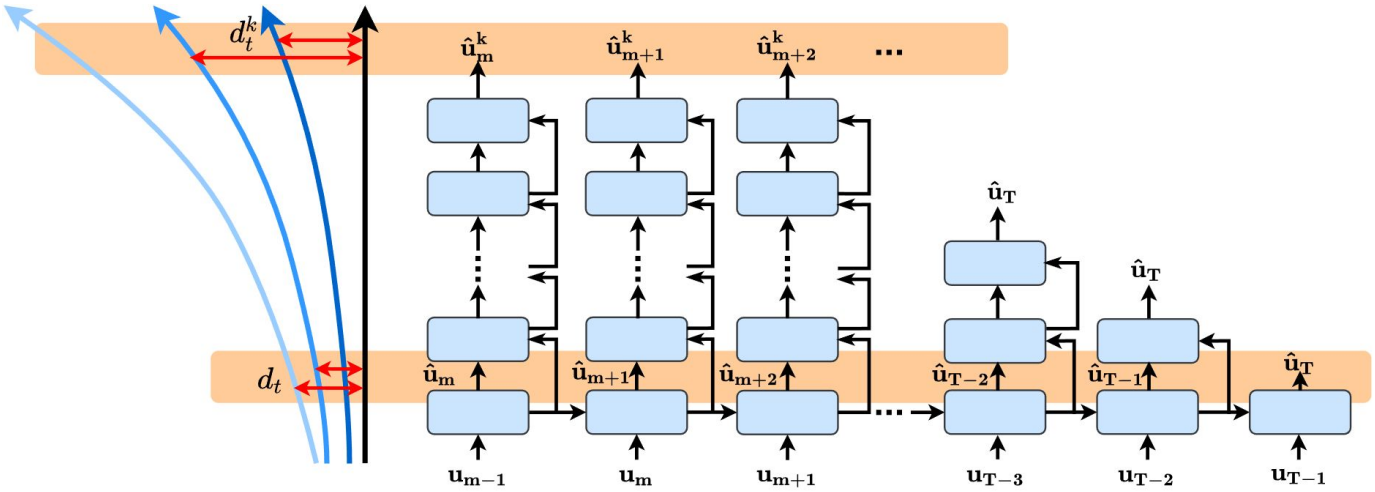


Fig. 3: Our error trajectory tracing architecture for time-horizon prediction. Each rounded rectangle represents a GRU cell, all of which share weights. Arrows entering the side of cells represent hidden state (h_t) transfer, arrows pointing up represent prediction of the next state (an application of transition function $\mathcal{F}$), $\hat{\mathbf{u}}_t$. The trajectories on the left represent modeled trajectories (light blue) against the ground-truth (black), illustrating the improvement resulting from training on a model's own predictions from teacher forcing to horizon forcing. Each tower models the future evolution of the chaotic system from the prediction made after a single step (lower orange-shaded box). Our first-stage training minimizes d_t (from the longer red arrow at the bottom of the figure to the arrow immediately above it). The upper orange-shaded box represents optimization at the time horizon ($k+1$ steps); the later training stage minimizes d_t^k (from the longer red arrow at the top of the figure to the arrow immediately above it), the time horizon loss.

cell (the next level up on the diagram) representing the next state in the trajectory starting at the prediction (the first step on the error trajectory). We then extend each of these towers of GRU cells until we reach the k_{th} (horizon) layer. Each of these GRU "towers" traces the evolution of the trajectory starting with the initial 1-step predictive error from the original trajectory. Each successive GRU cell in the tower represents an additional application of the transition function $\mathcal{F}$. Thus, $\mathbf{u}_t$ in Eq. 4 should be replaced by $\hat{\mathbf{u}}_t$.

Using backpropagation through each GRU "tower", $\partial h_t/\partial h_{t-1}$ can be updated as follows (terms differing from standard backpropagation through time are indicated from below with brackets):

$$\begin{aligned}\frac{\partial h_t}{\partial h_{t-1}} &= \frac{\partial h_t}{\partial s_t}\frac{\partial s_t}{\partial h_{t-1}} + \frac{\partial h_t}{\partial z_t}\frac{\partial z_t}{\partial h_{t-1}} + \overline{\frac{\partial h_t}{\partial h_{t-1}}} \\ &= \frac{\partial h_t}{\partial s_t}\left(\underbrace{\frac{\partial s_t}{\partial \hat{\mathbf{u}}_t}\overline{\frac{\partial \hat{\mathbf{u}}_t}{\partial h_{t-1}}}} + \frac{\partial s_t}{\partial r_t}\left(\overline{\frac{\partial r_t}{\partial h_{t-1}}} + \underbrace{\frac{\partial r_t}{\partial \hat{\mathbf{u}}_t}\overline{\frac{\partial \hat{\mathbf{u}}_t}{\partial h_{t-1}}}}\right) + \overline{\frac{\partial s_t}{\partial h_{t-1}}}\right) \\ &+ \frac{\partial h_t}{\partial z_t}\left(\overline{\frac{\partial z_t}{\partial h_{t-1}}} + \underbrace{\frac{\partial z_t}{\partial \hat{\mathbf{u}}_t}\overline{\frac{\partial \hat{\mathbf{u}}_t}{\partial h_{t-1}}}}\right) + \overline{\frac{\partial h_t}{\partial h_{t-1}}}\end{aligned} \tag{7}$$

All of the GRU cells in this architecture share weights. By tuning them with both the horizontal GRU and the forward evolutions of the error from each step, we are able to use this forward-looking training method to tune our representation to have strong short-and-long-term performance. In next section we will present a novel training procedure for optimal training of our ETT architecture.

D. A novel training procedure: Horizon Forcing

A challenge remains to be addressed before the ETT can be trained to incorporate the Lyapunov horizon loss. As k increases, a predicted trajectory may gradually deviate from the ground truth and then coincidentally approach the ground truth again after some step j. If we set k equal to a step that is greater than j and $\|d_t^k\|$ is small, this small $\|d_t^k\|$ will mislead the training of ETT because the big errors before the k-step are ignored. However, if we want to extend our model's prediction horizon as far as possible, we may need to let ETT be trained well with a large k. Inspired by these, we apply a novel training method to our ETT architecture for better training results. See Fig. 3

1) First, we train the 0_{th} layer (the bottom orange-shaded box in Fig. 3) using standard teacher forcing to reach a strong one-step-ahead baseline. When this layer is well trained, the divergence $d_t^{(i)}$ (in Eq. 3) will decrease and the prediction trajectory will be tightened in to the ground truth trajectory. This will reduce one-step prediction error, the difference $\|\hat{\mathbf{u}}_t^{(i)} - \mathbf{u}_t^{(i)}\|$ in eq. 3.
2) After the $d_t^{(i)}$ training task converges, we choose a small k and start to train the horizon (k_{th}) layer (the top orange-shaded box in Fig. 3). This further reduces the $(k+1)$ step prediction error $d_t^{(i),k}$, optimizing out the large future errors that result from the small errors made by the converged one-step model.

3) By using the resulting model from step 2 as a new baseline, Step 2 can be repeated n times to further extend the time horizon to $n * k$,

A benefit of our Horizon Forcing approach is that when we train the horizon (k_{th}) layer, all of the GRU cells will be updated because their weights are shared. All outputs and states in the ETT architecture are updated during each training session, so minimizing the $(k+1)$ step prediction error will continue minimizing the one step prediction error, which is difficult to achieve when training a standalone GRU.

IV. Experiments

A. Compared Models

We validate the utility of the Horizon Forcing method on a deep recurrent network architecture by using $k = 5$, yielding a teacher-forcing model (1-step ahead), and Horizon-Forced models with horizons at 5, 10, 15, and 20 time-steps ahead for all experiments. The RNN architecture is a single-layer GRU with 256 latent units. We compare our Horizon Forcing training regime with Teacher Forcing and Scheduled Sampling [16] regimes optimizing a standard GRU network with the same cell dimensionality. We run Scheduled sampling under two experimental settings: 1) where we generate a full prediction vector and sample from it at each time step with probability $1 - \epsilon_i$ (which leaves each sampled prediction independent) and where we allow for early stopping identically to other methods (called Scheduled Sampling - Early Stopping or SSES in our results table) and 2) where errors are computed sequentially (and thus are dependent) and we enforce that the entire schedule of sampling probabilities must be completed at training (and which we call Scheduled Sampling - Full Schedule or SSFS). We run all SSFS models with an inverse sigmoid decay schedule, which dramatically outperformed linear or exponential schedules during validation. We also benchmark against BLS, ELM, and three transformer-based deep neural models: Music Transformer (MTF), Informer (IF), and Autoformer (AF). Our BLS model has 100 latent units (5 windows and 20 latent units per window) and 31 enhanced units; ELM has 500 latent units; the music transformer has four stacked encoders, each with four attention heads (64 latent units per head). To avoid the influence of training hyperparameters, we use 100 sequence time steps as input across all experiments and only tune the learning rate, r. For Informer and Autoformer, we train using the same architecture shape and learning rate used in their papers, and feed each model's encoder 100 time steps to ensure all models are working with the same amount of data at inference.

All experiments were performed on a private Linux GPU server with 48 CPU cores, 1TB RAM, and 8 Nvidia 1080ti GPUs (each with ∼11GB memory). Each model was trained using a single GPU, and was implemented in Tensorflow 2.0. We use a batch size of 30 trajectories throughout all experiments. The maximum epochs are set to 200 but early stopping is employed to stop training if no improvement is made after 15 epochs. We also reduce the learning rate by 10% after 5 epochs of no improvement.

TABLE I: Experiment Setting

Data set	Vars	Inference steps	δ_γ	δ_ζ	δ_μ
Lorenz	3	200	3.1065	0.228	0.1105
Accelerometer	3	300	0.2935	0.3635	0.182
Roaming Worm	5	160	2.265	0.8575	0.4615
Gait Force	6	300	158.6875	0.501	0.2565
Electricity	1	300	121.54	0.219	0.0595

B. Datasets

We study datasets corresponding to a well-known Lorenz systems and a suite of real-world time series compiled in [27].

*1) **Lorenz '63 system**:* The Lorenz '63 system is a dynamical system presented by Edward Lorenz in [28] as a means to study some of the chaotic aspects of the atmosphere in tractable equations, see Fig.1 (a).

The system has three variables, X, Y, and Z, which are related by the following system of differential equations (dots denote the derivative of a variable with respect to time):

$$\begin{aligned} \dot{x} &= \sigma(y - x) \\ \dot{y} &= x(\rho - z) - y \\ \dot{z} &= xy - \beta z \end{aligned} \tag{8}$$

When generating data, we set σ, ρ, and β to $10, 28$, and $\frac{8}{3}$. These values were presented in the original '63 paper as settings that yield chaotic dynamics, and are commonly used in literature. We use Δt of 0.05 for each step and a stride of five to divide the sequence into 8,500 training examples and 1,201 testing examples.

*2) **Real-World Datasets**:* [27] compiles a suite of real-world data time series for evaluation of chaotic attractor reconstruction. We use the following data sets, with each training sequence being 100 time steps, split on sequence identifier (entire sequences are assigned to be in exactly one of train or test):

- **Roaming Worm** is a time series that tracks the evolution of the curvature of worm *C. Elegans*. We use a stride of four to section the dataset into 5,000 training sequences and 1,649 testing sequences.
- **Accelerometer** contains accelerometer readings from a gait database for a study participant walking with a smartphone. We use 5,500 train samples and 801 test samples, generated with a stride of two.
- **Gait Force** includes gait force measurements for a walking subject. We use 5,600 train samples and 1,023 test samples, with a stride of nine.
- **Electricity** is a time series of the mean power usage (in kilowatts) by 321 clients of a Portuguese power company from 2011 to 2014, sampled every 15 minutes. We used

a stride of 23 time steps to generate 5,000 train samples and 1,081 test samples.

C. Evaluation and Metrics.

In terms of long-term prediction, minor errors from the early steps tend to compound as the predicted horizon extends. Furthermore, we observe that in some problem domains, the error can accumulate to such a degree that past a certain time step, any resemblance a predicted sequence has with the true sequence (as quantified by an evaluation metric) is coincidental and no longer the result of having a useful long-term representation of the system's dynamics.

For this reason, in this study we employ a set of commonly-used evaluation metrics that compare a single predicted step with a single ground-truth step (see table II), and monitor their change over time. Because we value a model's ability to make useful predictions for as long as possible, we evaluate the competing methods by the expected amount of predictions (number of time steps into the future) that can be made before an error of a certain magnitude (as measured by metric M) occurs. We refer to this as the model's **Prediction Horizon** and formulate it as follows:

$$\mathbb{P}_M = \min_t \left\{ t \;\middle|\; \frac{1}{N}\sum_{i=1}^{N} M\left(\mathbf{u}_t^{(i)}, \hat{\mathbf{u}}_t^{(i)}\right) > \delta_M \right\} \tag{9}$$

Here $\mathbb{P}_M$ is the prediction horizon as measured by metric M for the model making predictions $\hat{\mathbf{u}}_t^{(i)}$.

δ_M denotes the threshold being used. A balance must be struck with respect to this value: if δ_M is chosen to be too low, the prediction horizon will be short and noisy and fail to differentiate between methods that perform well and those that perform poorly. If δ_M's value is taken to be too large, then prediction horizon will be long and also fail to differentiate strong from weak methods: in some cases, it's possible that a high δ_M could result in the threshold never being reached for a given sequence, in which case that sequence's contribution to the predicted range would be its length, greatly increasing the range value in a manner that is not meaningful for evaluation. Therefore, we seek a low threshold that isn't immediately crossed in practice. We find that among the datasets we use and methods we compare, setting δ_M to be the average performance of the best and second-best methods meets our criteria and allows us to avoid the use of arbitrarily selected thresholds. We report the δ_M values used for each dataset in table I, along with the number of variables and inference steps used.

We compute prediction horizon with respect to three metrics: *Root Mean Squared Error* (RMSE), *Mean Normalized Error* (MNE), and *Symmetric Mean Absolute Percent Error* (SMAPE), the computation of which we detail in Table II; t is the time step at which the metric is being calculated and D is the number of dimensions at time step t, indexed by d.

TABLE II: Metrics.

Metric	Expression	Symbols: Expectation & Prediction Horizon
RMSE	$\gamma\left(\mathbf{u}_t^{(i)}, \hat{\mathbf{u}}_t^{(i)}\right) = \sqrt{\frac{1}{D}\sum_{d=1}^{D}\left\|\hat{\mathbf{u}}_{t,d}^{(i)} - \mathbf{u}_{t,d}^{(i)}\right\|^2}$	$\mathbb{E}_\gamma; \mathbb{P}_\gamma$
MNE	$\zeta\left(\mathbf{u}_t^{(i)}, \hat{\mathbf{u}}_t^{(i)}\right) = \frac{1}{D}\sum_{d=1}^{D}\frac{\left\|\hat{\mathbf{u}}_{t,d}^{(i)} - \mathbf{u}_{t,d}^{(i)}\right\|}{\left\|\mathbf{u}_{t,d}^{(i)}\right\|}$	$\mathbb{E}_\zeta; \mathbb{P}_\zeta$
SMAPE	$\mu\left(\mathbf{u}_t^{(i)}, \hat{\mathbf{u}}_t^{(i)}\right) = \frac{1}{D}\sum_{d=1}^{D}\frac{\left\|\hat{\mathbf{u}}_{t,d}^{(i)} - \mathbf{u}_{t,d}^{(i)}\right\|}{\left\|\hat{\mathbf{u}}_{t,d}^{(i)}\right\| + \left\|\mathbf{u}_{t,d}^{(i)}\right\|}$	$\mathbb{E}_\mu; \mathbb{P}_\mu$

We also report the expectation with respect to each metric, computed as follows:

$$\mathbb{E}_M = \frac{1}{N \times T}\sum_{i=1}^{N}\sum_{t=1}^{T} M\left(\mathbf{u}_t^{(i)}, \hat{\mathbf{u}}_t^{(i)}\right) \tag{10}$$

TABLE III: Ablation Study

Dataset	M	Teacher Forcing	ETT20	HF20
Lorenz	$\mathbb{E}_\gamma$	3.944	8.061	**3.053**
	$\mathbb{P}_\gamma$	107	0	**127**
	$\mathbb{E}_\zeta$	0.289	0.589	**0.224**
	$\mathbb{P}_\zeta$	105	0	**126**
	$\mathbb{E}_\mu$	0.141	0.32	**0.108**
	$\mathbb{P}_\mu$	106	0	**126**
Roaming Worm	$\mathbb{E}_\gamma$	3.215	3.225	**2.212**
	$\mathbb{P}_\gamma$	29	0	**60**
	$\mathbb{E}_\zeta$	1.28	1.275	**0.831**
	$\mathbb{P}_\zeta$	27	0	**61**
	$\mathbb{E}_\mu$	0.576	0.644	**0.448**
	$\mathbb{P}_\mu$	32	0	**61**
Accelero-meter	$\mathbb{E}_\gamma$	0.333	0.742	**0.260**
	$\mathbb{P}_\gamma$	92	0	**208**
	$\mathbb{E}_\zeta$	0.413	0.985	**0.320**
	$\mathbb{P}_\zeta$	92	0	**214**
	$\mathbb{E}_\mu$	0.207	0.472	**0.162**
	$\mathbb{P}_\mu$	89	0	**209**
Gait Force	$\mathbb{E}_\gamma$	202.483	183.919	**157.724**
	$\mathbb{P}_\gamma$	93	109	**140**
	$\mathbb{E}_\zeta$	0.651	0.586	**0.5**
	$\mathbb{P}_\zeta$	90	112	**143**
	$\mathbb{E}_\mu$	0.41	0.376	**0.318**
	$\mathbb{P}_\mu$	58	85	**120**
Electricity	$\mathbb{E}_\gamma$	159.618	214.595	**98.447**
	$\mathbb{P}_\gamma$	111	0	**207**
	$\mathbb{E}_\zeta$	0.292	0.436	**0.177**
	$\mathbb{P}_\zeta$	114	0	**209**
	$\mathbb{E}_\mu$	0.122	0.169	**0.078**
	$\mathbb{P}_\mu$	29	0	**100**

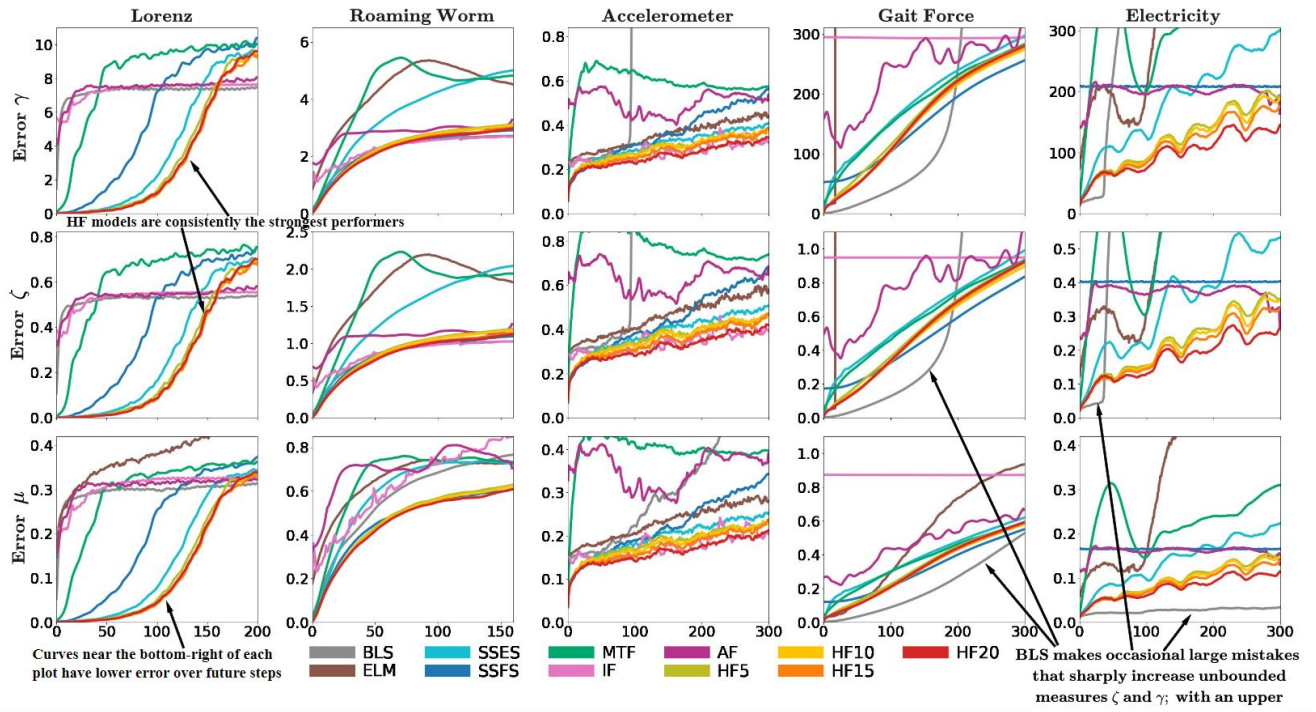


Fig. 4: Error curves illustrating benchmark error accumulation over time for various approaches and a model using Horizon-Forcing across each dataset and metric. We see that the Horizon-Forced models consistently have lower error over a longer time than alternative methods (HF error curves tend to be bellow the others). Code is available at https://github.com/Yong-Zhuang/HorizonForcing

D. Ablation Study

1) ***Horizon-Forced Models and Baseline GRU****:* We first present evidence of the efficacy of our Lyapunov horizon loss. We compare the model training with and without Lyapunov horizon loss, which are ETT with $k = 5$ trained with Horizon Forcing 4 times (HF20) and a GRU model trained with Teacher Forcing, respectively. The results across all datasets are provided in Table III. As we expect, HF20 outperforms the Teacher Forcing across all datasets.

2) ***Error Trajectory Tracing Without Horizon Forcing****:* To demonstrate the need for Horizon Forcing to train our ETT architecture, we run a set of experiments where, instead of iteratively training k-step ahead models (where $k = 5$ in this case) to work our way out to the desired horizon k steps at a time, we directly train to the horizon (20 steps ahead). This choice of k is large enough that the aforementioned challenge apply and performance is poor. We have included these results in Table III as ETT20, and compare them with HF20.

3) ***Choice of Horizon****:* Here, we compare various choices of time horizons by repeating Horizon Forcing $\{1, 2, 3, 4\}$ times with $k = 5$, resulting in models trained with respect to horizons at 5, 10,15, and 20 time steps. Results can be found in Table IV. HF models show robustness to the choice of training metric—with few exceptions, when an HF model is best in one metric, it tends to also be best in others.

Generally, training with a larger time horizon improves performance; on 3 of 5 datasets, HF20 is the best model. When it is not, the results are fairly close across horizons (prediction horizon within 1-2% of the other models), but on those datasets where there are large differences the longer horizon dramatically outperforms the others.

E. Benchmarking

We compare Horizon-Forced models with Broad Learning System (BLS), Extreme learning machine (ELM), Scheduled Sampling - Early Stopping(SSES), Scheduled Sampling - Full Schedule(SSFS), Music Transformer (MTF), Informer (IF), and AutoFormer (AF). Expected values of metrics and their prediction horizon are presented in Table IV, and full error sequences (average error over all test sequences at each time step) are shown in Fig. 4. Lower is better for all expected values and higher is better for all prediction horizon. Horizon-Forced models are consistently better than the alternative methods, and the average error sequences (Fig. 4) show that in some cases (the BLS model on gait force, notably) strong performance in Table IV from another model is actually indicative of a tradeoff with HF. The BLS model on gait force has very strong initial predictive performance but has a sharp increase in average error around 200 time-steps in the future (this is smoothed out by SMAPE, which has BLS as the clear best performer); if predictions beyond that horizon were necessary in practice, Horizon-Forced models would still be preferred. We can also see SMAPE strongly favoring another method (again BLS) on the electricity dataset—BLS has very

TABLE IV: Benchmarking

		Benchmarking ("-" indicates error greater than 10^3)							Horizon Forcing			
Dataset	M	BLS	ELM	MTF	IF	AF	SSES	SSFS	HF5	HF10	HF15	HF20
Lorenz	$\mathbb{E}_\gamma$	7.201	–	8.206	7.305	7.442	3.806	5.619	3.206	3.007	**2.975**	3.053
	$\mathbb{P}_\gamma$	2	0	17	0	0	110	73	121	128	**129**	127
	$\mathbb{E}_\zeta$	0.519	–	0.610	0.530	0.537	0.279	0.409	0.235	0.221	**0.219**	0.224
	$\mathbb{P}_\zeta$	2	0	16	0	0	109	71	121	127	**128**	126
	$\mathbb{E}_\mu$	0.295	0.38	0.294	0.307	0.307	0.135	0.201	0.114	0.107	**0.106**	0.108
	$\mathbb{P}_\mu$	2	0	17	0	0	110	73	123	127	**128**	126
Roaming Worm	$\mathbb{E}_\gamma$	2.228	4.283	4.164	2.294	2.813	3.449	2.240	2.333	2.302	2.239	**2.212**
	$\mathbb{P}_\gamma$	52	14	22	49	14	32	53	53	55	58	**60**
	$\mathbb{E}_\zeta$	0.843	1.745	1.687	0.874	1.086	1.383	0.838	0.885	0.872	0.843	**0.831**
	$\mathbb{P}_\zeta$	52	12	22	49	12	31	56	54	56	59	**61**
	$\mathbb{E}_\mu$	0.565	0.648	0.651	0.626	0.702	0.591	0.462	0.465	0.458	0.449	**0.448**
	$\mathbb{P}_\mu$	42	18	23	30	11	33	55	57	59	**61**	**61**
Accelero– meter	$\mathbb{E}_\gamma$	43.834	0.353	0.592	0.280	0.502	0.312	0.362	0.294	0.293	0.281	**0.260**
	$\mathbb{P}_\gamma$	80	65	4	111	0	113	80	123	124	138	**208**
	$\mathbb{E}_\zeta$	59.822	0.457	0.769	0.341	0.635	0.388	0.439	0.366	0.361	0.345	**0.320**
	$\mathbb{P}_\zeta$	66	45	3	111	0	111	88	123	125	138	**214**
	$\mathbb{E}_\mu$	0.312	0.231	0.399	0.177	0.355	0.196	0.222	0.184	0.180	0.173	**0.162**
	$\mathbb{P}_\mu$	64	45	3	110	0	91	79	123	124	140	**209**
Gait Force	$\mathbb{E}_\gamma$	996.947	–	180.482	293.610	237.578	188.083	**149.072**	162.578	154.797	155.979	157.724
	$\mathbb{P}_\gamma$	**189**	18	109	0	0	105	159	136	144	143	140
	$\mathbb{E}_\zeta$	3.159	–	0.576	0.949	0.769	0.602	**0.466**	0.514	0.488	0.495	0.500
	$\mathbb{P}_\zeta$	**186**	18	112	0	1	107	167	139	147	145	143
	$\mathbb{E}_\mu$	**0.199**	0.497	0.364	0.871	0.465	0.381	0.310	0.328	0.314	0.314	0.318
	$\mathbb{P}_\mu$	**195**	93	80	0	0	77	126	115	120	122	120
Electricity	$\mathbb{E}_\gamma$	–	497.149	330.938	618.958	200.252	184.876	208.360	123.850	119.230	112.004	**98.447**
	$\mathbb{P}_\gamma$	41	6	9	0	0	73	0	121	123	163	**207**
	$\mathbb{E}_\zeta$	27.871	0.818	0.652	0.939	0.369	0.338	0.403	0.223	0.215	0.201	**0.177**
	$\mathbb{P}_\zeta$	39	9	10	0	0	41	0	123	124	166	**209**
	$\mathbb{E}_\mu$	**0.026**	0.425	0.229	0.886	0.16	0.139	0.165	0.096	0.093	0.088	0.078
	$\mathbb{P}_\mu$	**300**	1	5	0	0	23	0	61	63	65	100

low expected SMAPE and the clear lowest SMAPE average error sequence in Fig. 4 (bottom row, far right plot), but this is actually the result of a small number of predicted values with very large error magnitude. The inclusion of the predicted value in the denominator of SMAPE upper bounds its value at 1, which limits the impact of the occasional error spikes on the expected SMAPE, resulting in improved performance by that metric. With its strong performance by SMAPE and weak performance when measured with RMSE or Normalized Error, the value of the model in such a situation would depend on the details of the task and may be cause for concern in practice.

We additionally find that Scheduled Sampling run to its full schedule (SSFS) performs worse than the version that we allow to stop when validation loss stops decreasing (SSES) on four of the seven datasets. SSFS outperforms SSES on the Roaming Worm dataset (remaining lower than the HF models) and on the Gait Force dataset, where its results are excellent (only BLS is better) and HF underperforms. SSFS also fails to properly converge on the Electricity dataset, which we believe is due to the that dataset being a stable time series with the presence of "spikes" in usage that can lead to high predictive error (or not) at random based on sampling chance.

The transformer models (Music Transformer, Informer, and AutoFormer) are not able to match the performance of our Horizon-Forced models, with the improvement most notable over the known chaotic systems. The strange wave-like error curve AutoFormer produces on Lorenz is due to the double-lobed trajectory of the attractor; AutoFormer is designed to use autoregressive attention-based features to process seasonality and trend, and so does not respond to the long trips out on the wings. This cyclical error curve shows that behavior—when the true trajectory is close to the center, the error is relatively low, when the true trajectory is far out on a wing, the error is high. Some transformer models perform at par with HF on some datasets (Informer is excellent on the Accelerometer data, Music Transformer performs well on Gait Force, Informer and Autoformer both predict Roaming Worm well), but it can be seen that the HF models are the most consistent across datasets and metrics.

F. Discussion

We have shown our method effective for use in simulated chaotic systems and real-world datasets. Here, we discuss the situations in which our approach is most valuable.

HF/ETT is designed to iteratively optimize over error trajectories every k steps in the future (i.e., optimizing 1-step, k-step, $2k$-step error, etc), which is ideal for systems with exponential error growth and where trajectories starting at initially close points (say, a ground-truth state in a phase space and a good model prediction of that state) could diverge

dramatically (or not) in response to small changes in their positions.

In such cases, creating an initial model of the dynamics (1-step error), then using that model to sample the error at increasingly distant horizons to enable the learning of the compounding effect (that is known to be present in chaotic systems) is an effective approach. However, systems could exist (especially non-chaotic systems) where the monotonic nature of the exponential divergence we picture in Fig. 3 does not hold, and either 1) 1-step error is strongly correlated with future error and HF/ETT is not necessary or 2) the error evolves in such a way that sampling 1- and k-step errors are not able to capture the behavior of, say, $\frac{k+1}{2}$-step errors due to periodicity or other correlations within the data. We do expect this effect to be small in practice (especially for small values of k).

V. Conclusion

Chaotic systems are found across many fields of study, including climatology, biology, virology, and many others. Improved methods to model the dynamics of such systems have the potential to offer broad utility in a number of applications. Here, we have introduced a new recurrent architecture, error trajectory tracing, and accompanying training regime, Horizon Forcing, for prediction of chaotic systems; instead of merely minimizing local error at each time-step, we monitor how those errors evolve at a time horizon so the model can optimize its parameters based on an estimation of the future cost of a small present error. By comparing with state-of-the-art machine learning methods, we validate our method on a widely studied simulated system and show it to be highly effective in making predictions on chaotic systems with sensitive dependence on initial conditions, and further validate against time series data from a number of real-world problem domains.

VI. Acknowledgements

Funding for this research was provided by NSF grants IIS-2008202 and IIS-2008276. This research was also partially supported by the Oracle grant to the College of Science and Mathematics at the University of Massachusetts Boston.

References

[1] S. H. Strogatz, *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry and Engineering*. Westview Press, 2000.

[2] D. Wang, W. Ding, K. Yu, X. Wu, P. Chen, D. L. Small, and S. Islam, "Towards long-lead forecasting of extreme flood events: a data mining framework for precipitation cluster precursors identification," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2013, pp. 1285–1293.

[3] M. Mitchell, *Complexity: A Guided Tour*. USA: Oxford University Press, Inc., 2009.

[4] D. Simovici and K. Hua, "Data ultrametricity and clusterability," in *Journal of Physics: Conference Series*, vol. 1334, no. 1. IOP Publishing, 2019, p. 012002.

[5] K. Hua and D. A. Simovici, "Dual criteria determination of the number of clusters in data," in *2018 20th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*. IEEE, 2018, pp. 201–208.

[6] C. P. Chen and Z. Liu, "Broad learning system: An effective and efficient incremental learning system without the need for deep architecture," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 1, pp. 10–24, 2017.

[7] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1-3, pp. 489–501, 2006.

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[9] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer," *arXiv preprint arXiv:1809.04281*, 2018.

[10] S. A. Billings, *Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains*. John Wiley & Sons, 2013.

[11] S. H. Rudy, J. N. Kutz, and S. L. Brunton, "Deep learning of dynamics and signal-noise decomposition with time-stepping constraints," *Journal of Computational Physics*, vol. 396, pp. 483–506, 2019.

[12] P. R. Vlachas, W. Byeon, Z. Y. Wan, T. P. Sapsis, and P. Koumoutsakos, "Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 474, no. 2213, p. 20170844, 2018.

[13] K. Hua and D. A. Simovici, "Long-lead term precipitation forecasting by hierarchical clustering-based bayesian structural vector autoregression," in *2016 IEEE 13th International Conference on Networking, Sensing, and Control (ICNSC)*. IEEE, 2016, pp. 1–6.

[14] J. Pathak, A. Wikner, R. Fussell, S. Chandra, B. R. Hunt, M. Girvan, and E. Ott, "Hybrid forecasting of chaotic processes: Using machine learning in conjunction with a knowledge-based model," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 28, no. 4, p. 041101, 2018.

[15] R. Wang, E. Kalnay, and B. Balachandran, "Neural machine-based forecasting of chaotic dynamics," *Nonlinear Dynamics*, pp. 1–15, 2019.

[16] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 1171–1179.

[17] F. Huszár, "How (not) to train your generative model: Scheduled sampling, likelihood, adversary?" *arXiv preprint arXiv:1511.05101*, 2015.

[18] A. M. Lamb, A. G. A. P. Goyal, Y. Zhang, S. Zhang, A. C. Courville, and Y. Bengio, "Professor forcing: A new algorithm for training recurrent networks," in *Advances in neural information processing systems*, 2016, pp. 4601–4609.

[19] J. Martinez, M. J. Black, and J. Romero, "On human motion prediction using recurrent neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2891–2900.

[20] D. Krueger, T. Maharaj, J. Kramár, M. Pezeshki, N. Ballas, N. R. Ke, A. Goyal, Y. Bengio, A. Courville, and C. Pal, "Zoneout: Regularizing rnns by randomly preserving hidden activations," *arXiv preprint arXiv:1606.01305*, 2016.

[21] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[22] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proceedings of AAAI*, 2021.

[23] J. Xu, J. Wang, M. Long *et al.*, "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[24] M. Han, S. Feng, C. P. Chen, M. Xu, and T. Qiu, "Structured manifold broad learning system: A manifold perspective for large-scale chaotic time series analysis and prediction," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 9, pp. 1809–1821, 2018.

[25] J. C. Sprott, *Chaos and time-series analysis*. Citeseer, 2003, vol. 69.

[26] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[27] W. Gilpin, "Deep reconstruction of strange attractors from time series," *Advances in neural information processing systems*, 2020.

[28] E. N. Lorenz, "Deterministic nonperiodic flow," *Journal of the atmospheric sciences*, vol. 20, no. 2, pp. 130–141, 1963.