Chaos
An Interdisciplinary Journal
of Nonlinear Science

Enhanced Long Short-Term Memory Architectures for Chaotic Systems Modeling: An Extensive Study on the Lorenz System

Roland Bolboacă[1, a)] and Piroska Haller[1, b)]
Electrical Engineering and Information Technology Department, George Emil Palade University of Medicine, Pharmacy, Science, and Technology of Targu Mures, Târgu Mureş 540139, Romanina

(Dated: 26 November 2024)

Despite recent advancements in machine learning algorithms, well-established models like the Long Short-Term Memory (LSTM) are still widely used for modeling tasks. This paper introduces an enhanced LSTM variant and explores its capabilities in Multiple Input Single Output (MISO) chaotic system modeling, offering a large-scale analysis that focuses on LSTM gate-level architecture, the effects of noise, non-stationary and dynamic behavior modeling, system parameter drifts, and short- and long-term forecasting. The experimental evaluation is performed on datasets generated using MATLAB, where the Lorenz and Rössler system equations are implemented and simulated in various scenarios. The extended analysis reveals that a simplified, less complex LSTM-based architecture can be successfully employed for accurate chaotic system modeling without the need for complex deep-learning methodologies. This new proposed model includes only three of the four standard LSTM gates, with other feedback modifications.

Chaotic systems are widely studied in various domains, including mathematics, engineering, and computer science. These systems often exhibit strange and unpredictable behaviors and are extremely sensitive to initial conditions. These behaviors are observed in many natural systems, such as fluid dynamics and celestial mechanics, and have been identified in several artificially constructed systems as well. Due to their complexity and non-linear chaotic behaviors, modeling such systems often becomes challenging, leading to high-complexity solutions. This paper introduces an enhanced Long Short-Term Memory (LSTM) variant for Multiple Input Single Output (MISO) chaotic system modeling, validated through an extensive experimental evaluation. The valuable results of this study can be utilized to deploy simple and efficient models with promising capabilities, including low sensitivity to noise and robustness to system parameter changes and non-stationarity. As revealed by extensive analysis, the proposed enhanced LSTM variant can be successfully used for both short-term and long-term forecasting, even in the presence of missing values.

I. INTRODUCTION

Chaos theory is a branch of mathematics and physics that analyzes complex dynamical systems with unpredictable behaviors caused by their extreme sensitivity to initial conditions[1]. Today, however, chaotic systems are studied across a wide range of domains and areas of study, including, but not limited to, Mathematics[2], Engineering[1], Education[3], Medicine[4] and Computer Science[5]. These systems exhibit chaotic behaviors, behaviors that can be observed in a multitude of natural systems as well, including fluid dynamics, celestial mechanics, and even artificially constructed systems that include traffic flow or financial markets. Moreover, as stated by numerous authors, the concept of chaos has been identified in a multitude of systems in a wide variety of domains[1,6].

On a less granular scale, chaotic systems are a subclass of the larger class represented by nonlinear systems. As described by Schoukens and Ljung[7], any system that deviates from linearity is considered nonlinear. Generally, nonlinear systems are characterized by the absence of linear relationships between inputs and outputs. The process of creating simpler representations of the underlying relationships of these systems is known as modeling. Well-known modeling techniques include white-box, black-box, and gray-box modeling. White-box modeling involves constructing and using the differential equations that govern the dynamics and behavior of the system, black-box modeling approaches usually involve creating system representations utilizing the observations collected from such systems, and gray-boxes are a mixture of the previous two methodologies[7].

This paper explores the modeling of chaotic Multiple Input Single Output (MISO) systems, which are described by the relationship between multiple inputs, hidden internal states, and a single output. Mathematically, these systems can be represented as $v(t) = f(u(t), h(t); \theta)$ or, alternatively, as $v(t) = f(u(t), h(t); \theta(t))$, where $v(t)$ is the output, $u(t)$ represents the external inputs, $h(t)$ denotes the hidden state variables, and θ represents the parameters of the system, which may also vary with time introducing non-stationairty.

Furthermore, this paper addresses data-driven modeling techniques, using machine learning approaches, namely Long Short-Term Memory (LSTM) models. In our previous work, we explored the influence of hyperparameters on the performance of LSTMs for nonlinear systems modeling[8], where we proposed an enhanced LSTM variant with promising results. This model was later successfully utilized for anomaly detection in nonlinear systems[9].

In this paper, a continuation of our previous work extends the applicability of such models to chaotic systems as well, with an extensive and comprehensive architectural analysis.

[a)]Electronic mail: roland.bolboaca@umfst.ro
[b)]Electronic mail: piroska.haller@umfst.ro

This applicability is demonstrated by training the models with data generated from a single set of initial conditions with a fixed parameter value to multiple initial conditions with additional parameter values, and testing them with datasets generated from a wide variety of initial conditions. Moreover, this paper explores new LSTM-based architectures and the possibility of further utilizing the resulting models in the direction of chaotic systems monitoring to detect any intervention on variables or system parameters based on the prediction errors. In summary, the major contributions of this large-scale study are as follows.

- Introduction of an enhanced LSTM model for MISO chaotic systems modeling, named LSTMTF. This model incorporates the previously observed output variable as an additional input during training and inference.
- A large scale study that includes an extensive empirical evaluation of the enhanced LSTMTF model across various scenarios and configurations, including gate-level architectural analysis, the influence of different types of noise, and short- and long-term prediction capabilities.
- Evaluation of the modeling performance for non-stationary and dynamic behaviors of the system under the variable time-dependent parameter condition.
- Design, development and validation of simpler and more efficient LSTM-based architecture based on the previous extensive experimental assessment.
- Results that illustrate the modeling and prediction capabilities of these models, even when trained with data generated from a single set of initial conditions and tested with data generated from various random initial conditions. The models presented in this paper contain a single hidden layer with few hidden units.

This remainder of the paper is organized as follows. Section II presents relevant related studies. The proposed methodology is presented in Section III. This is followed by experimental results in Section IV. Results pertaining a new LSTM-based architecture are presented and discussed in Section V. Finally, the paper concludes in Section VI.

II. RELATED WORK

Although nowadays chaotic system modeling and chaotic time series prediction pose interesting and challenging research directions, this field has been extensively explored for more than two decades. Recurrent neural network-based solutions have been shown to yield promising results in this direction[10–12]. For example, Zhang and Xiao[13] investigated the application of classic Recurrent Neural Network (RNN) architectures to model time series originating from chaotic systems. The authors proved that such architectures are able to perform both one-step and multi-step ahead accurate forecasts.

In a different direction, Liu-Schiaffini et al.[14] proposed a recurrent neural operator (RNO) to detect tipping points in non-stationary and chaotic dynamical systems. Their proposed model is trained on pre-tipping dynamics and is further utilized to detect tipping points in the future by utilizing an uncertainty-based approach. The authors validated their model in multiple scenarios, including Lorenz system datasets.

Patel and Ott[15] explored machine learning techniques to predict tipping points in chaotic dynamical systems. The authors showed that reservoir computing-based (RC) approaches trained on pre-tipping point orbits of non-stationary dynamical systems are capable of useful tipping point and post-tipping point predictions. Similarly to the work[14], the authors also demonstrated their approach on the Lorenz system. An interesting observation made by the authors highlights the fact that such approaches require careful and extensive hyperparameter tuning for accurate predictions of tipping points. Similarly, Patel et al.[16] explored machine learning approaches for chaotic dynamical systems prediction. The authors also introduced a hyperparameter optimization methodology for the utilized RC model. Their approach was validated in different scenarios, including the Lorenz system. It is worth noting that the authors did not perform prediction or forecast on every point, rather only on the tipping points.

An LSTM-based modeling approach for the Lorenz system was proposed by Dubois et al.[17]. This solution includes a deep learning model composed of LSTM and dense layers, capable of accurate predictions compared to simpler methods, such as the Kalman filter. The proposed model takes as input all the variables and predicts future values of the same variables. The authors also considered additional model inputs that consist of the first and second derivatives. Furthermore, they showed that LSTM-based solutions can effectively predict multiple steps ahead. One notable mention is that the authors plan to extend their work with hyperparameter tuning approaches for the proposed model.

Considering that our paper performs an extensive benchmarking and performance analysis for the LSTM model, we will also explore other studies that considered similar models for comprehensive evaluations. In this direction, Thomas Brueuel, a member of Google's research department, analyzed the performance of LSTM-based classifiers[18]. In his work, the author engages several aspects, including hyperparameters and non-liniarities, and their effects on the classification performance of an LSTM model. The list of studied hyperparameters includes the number of hidden units, the learning rate, and the size of the training mini-batch. The classification performance is evaluated on two popular handwriting datasets, i.e., MNIST and UW3. Among the results of this study, it is worth mentioning the dependence of the models on the learning rate. Conversely, the size of the mini-batch influenced the performance to a lesser extent. Furthermore, softmax training obtained superior results compared to the least squares method. The final conclusion of this study revealed that the models without peephole connections yielded the best performance.

Among the few existing large-scale studies on the performance of LSTMs, we find the work of Greff et al.[19]. This study focused on speech and handwriting recognition, and

polyphonic music modeling. The authors studied various LSTM architectures that include gate disabling and removal of activation functions. The authors utilized the standard activation functions, i.e., the sigmoid and hyperbolic tangent. Additionally, similarly to the previous study, the effects of various hyperparameters were analyzed, including the learning rate, hidden layer size, momentum, and input noise. One of their findings indicates that the learning rate had the most effect on performance, this followed by the number of LSTM cells on the hidden layer. Moreover, the authors showed that adding noise to the model inputs increased the training time and also decreased performance. Overall, in this study, the authors tested eight model variants and showed that standard LSTMs performed similarly analogized to the various architectures tested. It is worth mentioning that the hyperparameter analysis did not consider all values within well-defined ranges but rather utilized random searching techniques for each value. Furthermore, not all LSTM gates and architectures were considered.

The modeling performance of LSTM-based solutions was also highlighted in a recent paper by Al Nassan et al.[20]. Here, the modeling and prediction performance of a standard deep learning LSTM model is tested on the Lorenz system dataset. Although the proposed LSTM-based solution obtained notable results in terms of MSE and RMSE, even compared to similar approaches, the paper lacks important details about the utilized architecture and hyperparameters. This makes it close to impossible to recreate the experimental evaluation settings. Similarly to Dubois et al.[17], the authors utilized the three variables time-series as input, to predict the following values of the three variables.

In a slightly different direction, Farzad et al.[21] investigated various activation functions for an LSTM model used for classification tasks. The authors tested a total of 23 functions for the input, forget and output gates. This study utilized three datasets, including IMDB, MNIST and Movie Review. An interesting result showed that less known activation functions, including Elliott, modified Elliott, and softsign, yielded slightly better results compared to most popular activation functions. Furthermore, the activation functions that worked in the $[-0.5, 1.5]$ range obtained better results, and expanding the codomain range produced a generally better model classification performance.

In addition to the studies presented above, this work advances the long line of research in the direction of data-driven machine learning-based approaches for chaotic systems modeling. In this paper, an enhanced LSTM model, named LSTMTF, is presented. This model takes as input the external measured system input variables, together with previously observed output variable, and predicts the next value of the system's output variables, in MISO systems. This model is extensively evaluated from an architectural point of view, including the effects of noise, long- and short-term predictions, and in-depth analysis of the LSTM model gates. The LSTM gate analysis also includes the activation evolution of the gates over time and future LSTM architecture simplification and complexity reduction by disabling certain gates in the model cells. Compared to other studies, this in-depth systematic analysis specifically targets stationary and non-stationary chaotic system modeling using such approaches. Additionally, compared to other deep-leaning solutions with higher computational complexity, this paper presents a simpler model with a single hidden layer, which can be further simplified by the deactivation of either the entire cell gate, or the previously observed output true values.

Others have pointed out the importance of hyperparameter analysis for recurrent models. Although this paper addresses an architectural evaluation, in our previous work we addressed this final issue for various types of LSTM-based models[8], including feature analysis techniques. As such, we consider this work to be a continuation of an extensive and complete LSTM model analysis, together with possible application in the direction of chaotic systems modeling and anomaly detection.

III. METHODS

This section presents the standard LSTM model alongside the enhanced version proposed for chaotic systems modeling. It also includes a description of the analyzed system, e.g., the Lorenz system. In addition, extensive experimental scenarios are thoroughly described, along with the evaluation metrics used.

A. Long Short-Term Memory Model

LSTM models, initially proposed by Hochreiter and Schmidhuber in 1997[22], were designed as a solution to address the vanishing and exploding gradient problem encountered in vanilla versions of RNNs. Over time, LSTM models gained substantial popularity and became increasingly preferred in the field. LSTMs feature memory units that can capture both long- and short-term dependencies in time-series data. A visual representation of the standard LSTM cell is illustrated in Figure 1.

A typical LSTM layer comprises blocks containing memory cells and four gates, including input, candidates, output, and forget gates. Moreover, each block features two recurrent connections, namely the hidden state and the cell state representing short-term and long-term memory, respectively. The four gates control the flow of information to and from the memory cell. As implied by their names, the forget gate manages discarded information, the input gate regulates information to be stored, and the output gate computes the current output of the cell. Although some papers only describe three gates, the input gate is composed of two components which also include the cell candidate computation gate.

Without loss of generality, in the remainder of this paper, the term cell gate will refer to the cell candidate gate, as illustrated in Figure 1. The LSTM cell equations for the gates are as follows.

$$\mathbf{F}(t) = sigm(\mathscr{W}_F \mathbf{X}(t) + \mathscr{U}_F \mathbf{h}(t-1) + \mathbf{b}_F). \quad (1)$$

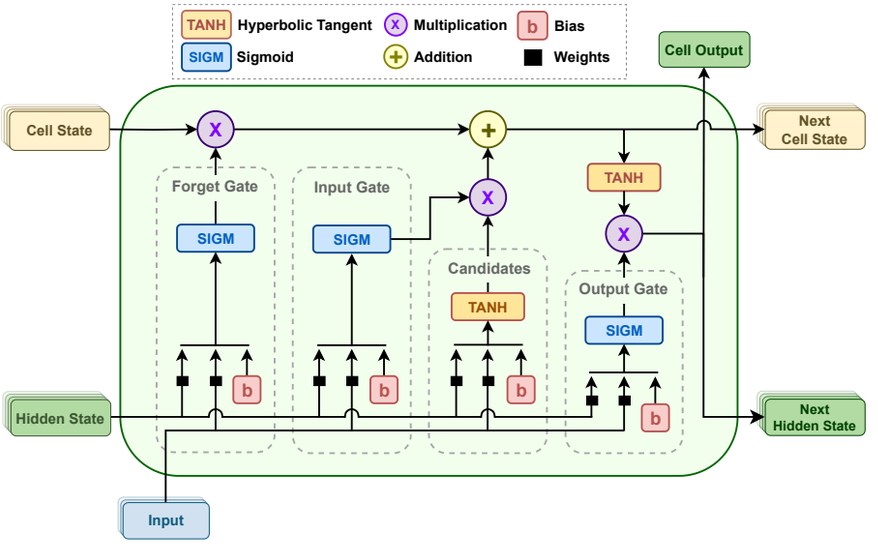

FIG. 1: Illustration of a single LSTM Cell together with the gates, connections, activation functions, weights, and internal states.

$$\mathbf{I}(t) = sigm(\mathscr{W}_I\mathbf{X}(t) + \mathscr{U}_I\mathbf{h}(t-1) + \mathbf{b}_I). \quad (2)$$

$$\mathbf{A}(t) = tanh(\mathscr{W}_A\mathbf{X}(t) + \mathscr{U}_A\mathbf{h}(t-1) + \mathbf{b}_A). \quad (3)$$

$$\mathbf{O}(t) = sigm(\mathscr{W}_O\mathbf{X}(t) + \mathscr{U}_O\mathbf{h}(t-1) + \mathbf{b}_O). \quad (4)$$

$$\mathbf{c}(t) = \mathbf{F}(t)\cdot\mathbf{c}(t-1) + \mathbf{I}(t)\cdot\mathbf{A}(t). \quad (5)$$

$$\mathbf{h}(t) = \mathbf{O}(t)\cdot tanh(\mathbf{c}(t)). \quad (6)$$

Here, the forget gate is shown in Equation 1, the input gate in Equation 2, the calculation of the cell state in Equation 5, the candidates of the cells in 3, the output gate in 4 and the computation of the hidden state in 6. Moreover, $\mathscr{W}$, $\mathscr{V}$, and $\mathscr{U}$ denote the weight matrices, X is the input vector, $\mathbf{A}$ denotes the vector of new candidates for the cell state, c is the current cell state, and b denotes the bias vectors.

The enhanced LSTM variant, named LSTMTF, as we originally proposed it[9], incorporates the previous true output value as an additional input, during the training and inference stages. This approach was inspired by the original Teacher Forcing (TF) algorithm[23]. Additional information on LSTMs with TF is available in previous work[8,9], where it was demonstrated that such models can outperform various other existing techniques.

The equations for LSTMTF at time t are presented below. Equation 7 denotes the forget gate, Equation 8 denotes the input gate, the cell state computation is shown in Equations 11, the cell candidate gate is presented in Equation 9, Equation 10 shows the output gate computations, and Equation 12 illustrates the hidden state. Additionally, Equation 13 describes the computation of the model's predicted output at time t, further denoted as $\hat{y}(t)$.

$$\mathbf{F}(t) = sigm(\mathscr{W}_F\mathbf{X}(t) + \mathscr{V}_F y(t-1) + \mathscr{U}_F\mathbf{h}(t-1) + \mathbf{b}_F). \quad (7)$$

$$\mathbf{I}(t) = sigm(\mathscr{W}_I\mathbf{X}(t) + \mathscr{V}_I y(t-1) + \mathscr{U}_I\mathbf{h}(t-1) + \mathbf{b}_I). \quad (8)$$

$$\mathbf{A}(t) = tanh(\mathscr{W}_A\mathbf{X}(t) + \mathscr{V}_A y(t-1) + \mathscr{U}_A\mathbf{h}(t-1) + \mathbf{b}_A). \quad (9)$$

$$\mathbf{O}(\mathbf{t}) = sigm(\mathscr{W}_O\mathbf{X}(t) + \mathscr{V}_O y(t-1) + \mathscr{U}_O\mathbf{h}(t-1) + \mathbf{b}_O). \quad (10)$$

$$\mathbf{c}(t) = \mathbf{F}(t)\cdot\mathbf{c}(t-1) + \mathbf{I}(t)\cdot\mathbf{A}(t). \quad (11)$$

$$\mathbf{h}(t) = \mathbf{O}(t)\cdot tanh(\mathbf{c}(t)). \quad (12)$$

$$\hat{y}(t) = \mathscr{P}_{\hat{y}}\mathbf{h}(t) + b_{\hat{y}}. \quad (13)$$

Here, $\mathscr{P}$ denotes the weight matrix, X is the input vector, $\mathbf{A}$ denotes the vector of new candidates for the cell state, c is the current cell state and b denotes the bias vectors. Let H represent the number of units on the hidden layer. Given a system with m number of measured signals, for any output all other $m-1$ signals are considered input in the model. If there are multiple outputs in the system, for each output a MISO system is created, as described above. Including the additional input, the number of inputs will be equal to m. The input vector X will be of size $X \in \mathbb{R}^{m\times 1}$. The dimensionality of the weight matrices is as follows: $\mathscr{W} \in \mathbb{R}^{H\times(m-1)}$, $\mathscr{V} \in \mathbb{R}^{H\times 1}$, $\mathscr{U} \in \mathbb{R}^{H\times H}$ and $\mathscr{P} \in \mathbb{R}^{1\times H}$ As nonlinear activation functions, the Sigmoid and Hyperbolic Tangent functions are utilized, as they appear in Equations 1–6. A visualisation of the LSTMTF

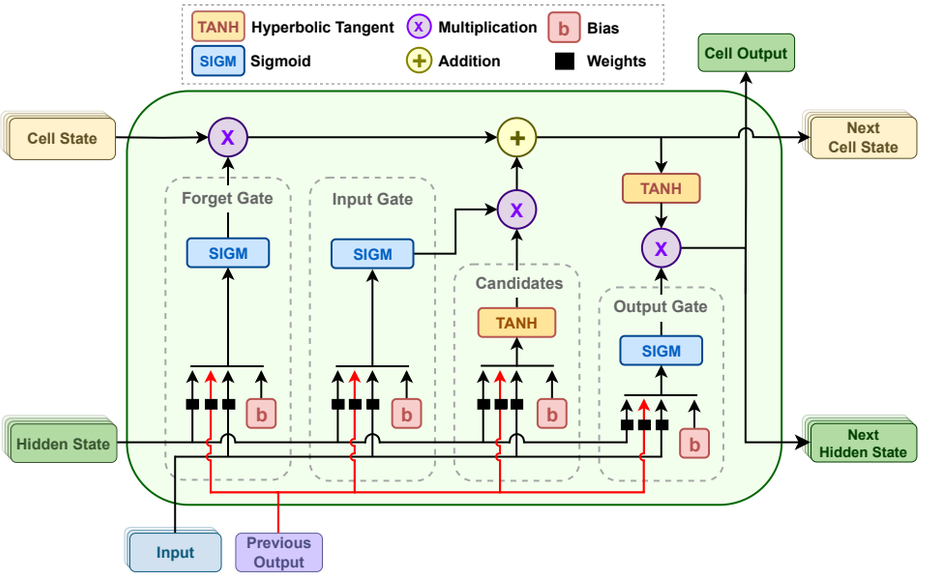

FIG. 2: Illustration of a single LSTMTF Cell together with the gates, connections, activation functions, weights, and internal states.

cell architecture, with the previous observed output variable fed as additional input, is illustrated in Figure 2.

Let $\mathbf{Y}$ denote the vector of observed output values, where $[y(0), y(1), ..., y(t)] \in Y$. Likewise, let $\hat{Y}$ denote the vector that contains the LSTMTF's predictions, where $[\hat{y}(1), \hat{y}(2), ..., \hat{y}(t)] \in \hat{\mathbf{Y}}$. The prediction error vector is computed as the difference between Y and $\hat{Y}$. The enhancement introduced to the LSTMTF model does not include additional recurrent connections from the output layer to the hidden layer; thus the Back Propagation Through Time (BPTT)[19,24–26] algorithm is utilized, where the backward propagation equations remain unchanged.

B. Lorenz System Data Generation

The Lorenz system, as originally proposed in 1963, represents a simplified model for atmospheric convention and the unpredictable behavior of the weather. In the original paper[27], the author introduced a theory to model the dynamics of a fluid warmed from below and cooled from above. In the field of chaos theory, this system remains a paradigm, as it effectively captures a wide variety of features of chaotic dynamics. Close initial conditions lead to very different trajectories, making the Lorenz system a chaotic dynamical system[6,28].

The Lorenz system equations are shown in Equation 14. This system is defined by the following parameters σ, ρ, and β. In the current context, σ represents the Prandtl number, ρ represents the Rayleigh number, and β is a geometric factor[6]. As originally described, the values of the parameters σ, ρ, and β are set, respectively, to 10, 28 and 8/3. The Lorenz system can perform complex and non-stationary dynamics based on the parameters. Generally, a stationary variant of the system is studied with parameter values held constant, while non-stationary variants are explored with dynamic and time-dependent values of ρ, with σ and β held constant[28].

$$\begin{cases} \dfrac{dx}{dt} = \sigma(y - x) \\ \dfrac{dy}{dt} = x(\rho - z) - y \\ \dfrac{dz}{dt} = xy - \beta z. \end{cases} \tag{14}$$

Recall the introduction section, where the general form of chaotic MISO systems was described as $v(t) = f(u(t), h(t); \theta(t))$. For the Lorenz system, variables x and y are considered system inputs while z is considered the system output. Specifically, for the Lorenz system, the previous equation can be rewritten as $z(t) = f(x(t), y(t), h(t); \sigma, \beta, \rho(t))$. As previously stated, to introduce non-stationarity, the ρ parameter values are varied over time. Additionally, the current paper will also consider a noisy system. As such, various types of noise are added at each time step, as presented in the following subsection.

The datasets used for the experimental evaluation are generated using MATLAB, i.e. utilizing the ODE45 solver[29] with multi-step integration. The ODE45 solver function implements the Runge-Kutta method with a variable time step for efficient computations. For each respective scenario, the simulations start with random initial conditions in the range [0, 1]. The time span for each simulation ranges from 1 to 100 with a time step of 0.01.

For each experimental scenario, as described in the next sections, individual datasets are constructed. Each simulation contains 9901 data points, and each dataset contains 100 simulations with random initial conditions. For each scenario, each dataset contains a total of 990,900 data points. If, for example,

a scenario includes variants with and without noise or multiple parameter values, each unique noisy dataset will contain 100 simulations of 990,900 observations. For the experiments in subsesction III C, the testing sample rates differ, resulting in fewer data points. To account for this difference, the number of simulations is increased to 300, resulting in 2,970,300 observations. For scenarios where the parameter ρ varies in time, for each case, a separate dataset is created.

A visualization of the generated data is illustrated in Figure 3. Here, 5000 data points are shown in three scenarios: without noise, with noise, and a non-stationary scenario.

C. Noise, Forecasting and Data Resampling Analysis

The first analysis focuses on the effects of the data accuracy and availability on the model identification and prediction capabilities. Considering real-life applications, where measurements contain different types of noise resulting from sensor readings or the communication medium, the model is analyzed in the presence of random Gaussian and uniformly distributed noise at various levels. Moreover, a scenario is explored with the assumption that system measurements at higher sample rates might be available only for short periods, or certain operating conditions might suffer major changes in time resulting in subsequent missing values. This results in a low availability of measured output data during inference. In addition, this set of experiments also explores the prediction capabilities of models trained and evaluated with different sampling rates.

Noise Influence Analysis

In this scenario, the models are trained with noise-free observations and tested in the presence of two types of noise at different intensities. The Gaussian or normal distributed noise is further denoted as $\mathcal{N}(m,s)$, where m denotes the mean and s the standard deviation. The uniformly distributed noise is further denoted as $\mathcal{U}$ $[\mathcal{U}_l, \mathcal{U}_h]$, where $\mathcal{U}_l$ and $\mathcal{U}_h$ denote the upper and lower limits of the values. For each type of noise, at each time step, a random value is generated and added to the observed values. The noise variables are denoted as ξ_x, ξ_y and ξ_z. Consequently, the Lorenz system equations are updated as follows:

$$\begin{cases} \dfrac{dx}{dt} = \sigma(y-x) + \xi_x(t) \\ \dfrac{dy}{dt} = x(\rho - z) - y + \xi_y(t) \\ \dfrac{dz}{dt} = xy - \beta z + \xi_z(t). \end{cases} \tag{15}$$

Forecasting Analysis

Here, the models are trained with data generated at a fixed sample rate. During inference, the models are evaluated under the assumption that the observed output true values become available at some point in the future. To account for the missing values between the true observed values, the previously predicted output values are utilized as additional input for subsequent time steps. Specifically, in this set of experiments, the multistep forecasting capabilities of the model are tested under the assumption that the predicted value (e.g., the output variable) is available only at various rates.

Data Resampling Analysis

Multiple models are individually trained on datasets sampled at numerous rates and tested on datasets sampled at various other rates. This addresses real-life scenarios where system measurements might be sampled at different frequencies during inference due to system behavioral changes that could appear in time. The prediction method explored here is one step ahead, while long-term forecasting was addressed above. Here, the testing subsets are created from the original files, by only selecting observations at the given frequencies.

D. Parameter Influence and Non-stationarity Analysis

Here, the influence of changes in the system parameter values is analyzed. This involves training the model with fixed values for a given parameter and performing inference with dynamic parameter values. In a realistic scenario, some parameters might be explicitly or implicitly hidden and may exhibit stochastic behaviors. Moreover, in chaotic systems, changes in parameter values can lead to unpredictable and chaotic behaviors or transitions from one-piece to multi-piece chaotic attractors[16].

Parameter Influence Analysis

Certain external interventions might only affect such parameters, indirectly affecting the system‘s behavior. As a result, the system might enter transient states or even faulty or anomalous states. This set of experiments underlines the limitations of the model in terms of accurately predicting the next values when such interventions occur.

Here, the model is trained with fixed values for a given parameter, and tested with various other parameter values. Additionally, the model is also trained with a set of time series data generated with multiple parameter values. In this case, the training and testing datasets are generated with multiple fixed values for the parameters. Specifically, the parameter value is increased using a step-wise methodology. Starting from an initial value, the parameter is incremented by a fixed amount every 10 simulations, with initial conditions chosen randomly in each simulation. This indefinitely causes the lack of trend in the generated data. Nevertheless, the next set of experiments will explore the case where the parameter is also time-dependent, introducing not only non-stationarity but also additional chaotic and unpredictable behavior in the system.

Non-stationarity Analysis

In the current scenario, the model is trained with observations generated with various fixed parameter values and tested with parameter values generated using time-dependent exponential and sin functions. Following a similar approach to[15] and as stated above, this approach introduces a variational trend in the data. In[16] and[15], the authors utilize similar time-dependent functions for the system parameters.

The two approaches to computing the parameter values at each time step are illustrated in Equations 16 and 17.

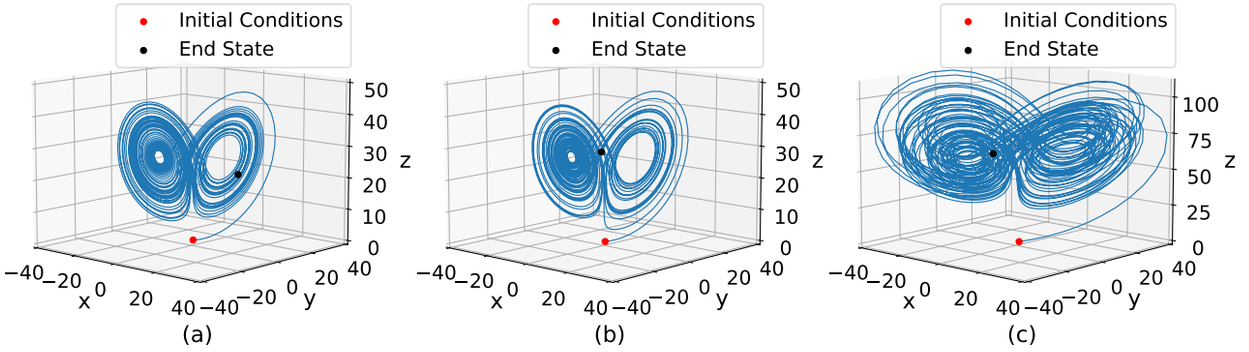

FIG. 3: Three-dimensional trajectory of the Lorenz system with various random initial conditions: (a) Noiseless, (b) Noisy (Gaussian Noise), (c) non-stationary (sin).

$$\rho(t) = \rho_0 + \rho_1 \cdot e^{t/\tau}. \tag{16}$$

$$\rho(t) = \rho_0 + \rho_1 \cdot sin(\gamma * t) + \mu * t. \tag{17}$$

In Equation 16, the values for ρ_0, ρ_1 and τ are constant throughout each simulation. Similarly, the parameters ρ_0, ρ_1, γ and μ from Equation 17 are also fixed.

E. LSTMTF Gate Analysis

In this scenario, three sets of experiments are performed. In the first set of experiments, the activation values of the LSTMTF and LSTM gates are analyzed during inference. The second set of experiments, as an extension to the previous set, investigates the effects of disabling the additional input, represented by the previous observed output value, in the four LSTMTF cell gates. In the third and last set of experiments, the effects of disabling the LSTMTF cell gates are analyzed.

LSTM Models Gate Activations Analysis

Gate activation values are extracted for the four gates during inference on a testing subset. This is analyzed for both architectures, standard LSTM and LSTMTF. These values are extracted after the computation of their respective activation functions, namely the sigmoid and hyperbolic tangent, as described by Equations 1 - 12 and presented in Figures 1 - 2. This set of experiments is required to understand the differences between the two models, starting from the cell gate level.

LSTMTF Gate Modifications Analysis

Here, for each gate individually, and all possible gate combinations, the previous observed true value is disabled (removed) while the exogenous inputs are kept. The effects of these procedures are analyzed when the additional input is disabled during training and inference. Among the gate combinations, the following are included, disabling the previous output true value individually only for input, output, forget gates, cell candidates, input-forget, forget-output, input-forget-cell candidates. A visual representation is also shown in Figure 2. When the previous output true value is disabled in the gates, Equations 7, 8, 9, 10, become the same as Equations 1, 2, 3, 4, for each specific case. For example, when acting on the forget gate, the LSTMTF cell will be described by Equations 1, 8, 9, 10. The same applies to all other scenarios.

LSTMTF Gate Disabling Analysis

Disabling the gates changes the LSTMTF equations as follows. In the case where the Forget gate is disabled, the cell state (e.g., the long-term memory) is not updated with any previous information; thus, it retains everything from the preceding time steps. In this case, Equation 11 is changed as shown in Equation 18 and Equation 7 is not utilized.

$$\mathbf{c}(t) = \mathbf{c}(t-1) + \mathbf{I}(t) \cdot \mathbf{A}(t). \tag{18}$$

In contrast, when the input gate is disabled, the cell state is partially updated with the new information, only from the cell candidates, as described in Equation 19. However, the forget gate still updates the cell state, and Equation 8 is not utilized.

$$\mathbf{c}(t) = \mathbf{F}(t) \cdot \mathbf{c}(t-1) + \mathbf{A}(t). \tag{19}$$

Next, while the output gate is disabled, the cell state is updated by both the input and forget gates, and the hidden state (e.g., short term memory) becomes equal to the cell state. Although this reduces the memory connection to the forthcoming time steps, the hidden state is still updated differently, compared to a simpler RNN model with a single recurrent connection. The change in the LSTMTF cell equations appears in the hidden state computation, as shown in Equation 20.

$$\mathbf{h}(t) = tanh(\mathbf{c}(t)). \tag{20}$$

In the final case, where the cell candidate is disabled, the cell state values are directly updated by the forget gate and input gate. These are differentiated only by point-by-point multiplication in the case of the forget gate, and point-by-point addition for the input gate. This is presented in Equation 21, and Equation 9 is not utilized.

$$\mathbf{c}(t) = \mathbf{F}(t) \cdot \mathbf{c}(t-1) + \mathbf{I}(t). \tag{21}$$

As mentioned in the previous section, a similar approach was performed in[19], where the authors tested the effects of disabling only one gate at a time for an LSTM-based classifier. In contrast, this paper tests the combination of multiple simultaneous disabled gates in LSTM regression models. Although numerous combinations are tested, there are extreme cases where disabling three gates basically converts the LSTM model into a simple RNN architecture.

IV. EXPERIMENTAL ANALYSIS RESULTS

This section presents the experimental results for each scenario as described above. The experimental assessment was implemented in Python, using Keras[30] with the TensorFlow backend[31]. The evaluation was performed on a Lenovo Legion laptop featuring an AMD Ryzen 5 5600H CPU, 32 GB DDR4 RAM, running the Windows 10 PRO. To access the LSTM gate activation values and to change the architecture at the gate level, the Keras and Tensorflow implementations were modified. To obtain deterministic results, the initial parameters of the models (e.g., weight matrices) could be generated using the same seed. However, the randomization of the initial seed induces realistic and probabilistic results. To minimize the possibility of biased results due to random weight initializations, each experiment was repeated 10 times. The results represent the average of these 10 repetitions for each scenario.

A. Evaluation Metrics

To measure the prediction performance of the deployed models, the following metrics are utilized throughout this paper: Mean Squared Error (MSE) as shown in Equation 22, the symmetric Kullback-Leibler divergence[32] (e.g., Jeffreys[33]) (SKL), from Shannon's entropy family[34], as shown in Equation 23, and the coefficient of determination R^2 illustrated in Equation 24.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2. \tag{22}$$

$$SKL(P,Q) = \sum_i (P_i - Q_i) ln \frac{P_i}{Q_i}. \tag{23}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}. \tag{24}$$

In Equations 22 and 24, y_i represents the observed value, $\hat{y}_i$ denotes the predicted value, $\bar{y}$ is the mean of the observed values, and n represents the number of observations. As for SKL divergence, in Equation 23, P and Q represent the distributions of the model predictions and true values.

While MSE is self-explanatory, the SKL is utilized to measure the divergence between predicted and actual probability distributions, offering a way to quantify the similarity between two distributions. Since the SKL is symmetric, it ensures that the divergence in either direction -from the observed to the predicted values or vice versa- is treated equally.

B. Model Parameters

The training hyperparameters and the architecture of the LSTM and LSTMTF models are as follows: Input sequence length: 40, training minibatch size: 32, training epochs: 100, learning rate: 0.01, learning rate decay rate: 0.04, learning rate decay steps: 40, hidden layers: 1, model layers: 3 (i.e., 1 linear input, 1 hidden, 1 regression output), hidden units 32, weights initialization Glorot Normal[35], output delay for input: 1, prediction mode: one-step and multi-step ahead. In addition, the model is configured to run in stateful mode. Namely, the hidden and cell states are not reset after each sequence, but only after each experiment. The remainder of the parameters and hyperparameters are used with their respective default values. As previously stated, the behavior of models for different hyperparameters has been addressed in previous work[8]. The inputs for the standard LSTM model are x(t) and y(t), while the output is z(t). For the LSTMTF model, the inputs are x(t), y(t), and z(t-1) while the output is z(t) (see Equation 15).

The utilized LSTM models are trained using a sequence-based methodology. As Keras does not inherently support model conversion, for point-by-point prediction and forecast, after training, a new model was created and the parameters were transferred. These parameters include weights, biases, and the initial states of the model, e.g., hidden and cell states. The initial states are considered as the final values that result after the model is trained.

C. Baseline Model Results

Before presenting the results, the prediction performance of the two models is illustrated in Figure 4. The observed and predicted values are shown at various resolutions, ranging from 50 to 5000 time steps. In this case, noise-free datasets were used for both training and testing. The models were trained on 15,000 data points and tested on 990,900 observations across 100 simulations. The enhanced prediction capabilities of the LSTMTF model are clearly visible compared to the standard LSTM model. Numerically, over the unseen testing set, the results are as follows. For the LSTM model the MSE was 0.001726, SKL was 0.201782, and R^2 was 0.941223. For the LSTMTF model the MSE value was 0.000057, SKL value was 0.002072, and the R^2 value was 0.998056. In this case, the LSTMTF model outperformed the vanilla model by over 100 times. These results are further used as reference values for the following experimental scenarios.

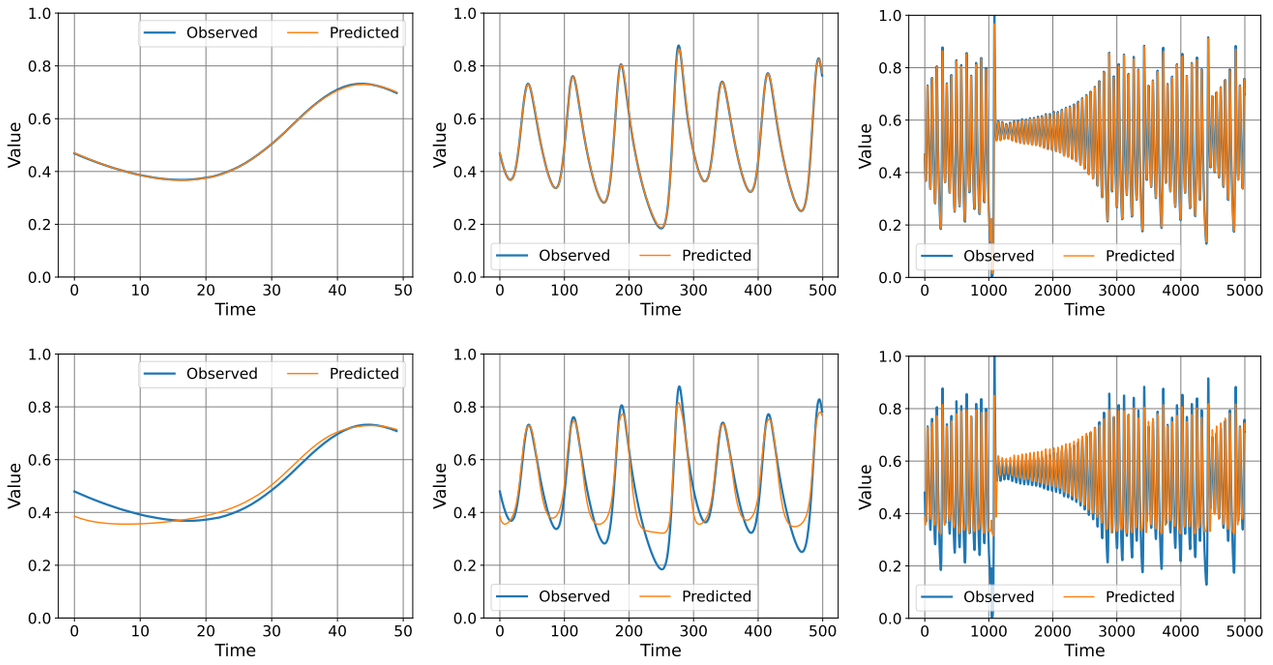

FIG. 4: Illustration of the LSTMTF (Top row) and LSTM (Bottom row) prediction performance over 50, 500, and 5000 values from the testing set.

D. Noise, Forecasting and Data Resampling Analysis Results

For this set of experiments, the Lorenz system parameters used for data set generation are as follows: $\rho = 28$, $\sigma = 10$, and $\beta = 8/3$.

Noise Influence Analysis Results

The effects of various types of noise on the performance of both the LSTMTF and LSTM models are shown in Table I. In this context, $\mathscr{N}$(Mean, Standard dev.) represents Gaussian (normal) noise, and $\mathscr{U}$ [min, max] represents uniform noise generated within the [min, max] range. The noise values were randomly generated and added to each observation at each time step, as previously discussed.

Although there were small variations in performance, neither the LSTMTF nor the LSTM model was significantly affected by the addition of noise, even at higher levels. This finding was consistently confirmed across all three metrics. Overall, the models demonstrated robustness against both types of noise, even at higher intensities. The robustness to noise was also confirmed by other research using data from other fields[19], namely speech and handwriting recognition, and polyphonic music modeling. However, these findings can be valuable for solutions that utilize noise for data privacy mechanisms, such as the works of Roman et al.[36,37].

Forecasting Analysis Results

Table II shows the long-term forecasting results of the LSTMTF model. The model was trained and tested with various combinations of data sample rates. During inference, the previously observed output variable is fed to the model at different rates, while in between the previously model predictions are given to the model.

The sample rates, as shown in Table II, represent how many values are forecast before using the previously observed true value. At the 0.01 rate, all true values are used. At the 0.02 rate, the previously observed output variable is used as additional input after every predicted value. For the 0.1 rate, the observed value is used after every 10 predictions. At the 0.5 rate, 50 predictions are made before using the true output value. Finally, the "all" column indicates that the previously predicted value is always used, and the true output value is never utilized.

The models trained with all available observations (e.g., 0.01 sample rate) achieved the best performance, even when forecasting multiple steps ahead or using only the previously predicted value as an additional input. Specifically, the best performance was observed when predicting 2 and 10 steps ahead, with the MSE increasing from 0.000039 to 0.000285 and the SKL rising from 0.000195 to 0.003040. Conversely, the worst-performing models were those trained with data sampled at a rate of 0.5, as significant information was discarded, preventing the models from capturing and modeling the complex relationships necessary for accurate regression. These results indicate that in this specific scenario, the models require more data for effective training. However, as shown in Table II, when higher sample rates are used for training, the LSTMTF model can be employed for long-term forecasting and still outperform the standard LSTM model in one-step-ahead prediction. Nonetheless, these results highlight both the forecasting limitations and strengths of the models, which may be advantageous depending on the context of data avail-

TABLE I: LSTMTF and Standard LSTM prediction performance with various noise levels. The models were trained with 15,000 data points and tested on 990,900 data points (100 simulations).

	LSTMTF			Standard LSTM		
	MSE	**SKL**	R^2	**MSE**	**SKL**	R^2
$\mathcal{N}(0, 0.1)$	0.000036	0.001132	0.998748	0.001882	0.284222	0.936115
$\mathcal{N}(0, 1)$	0.000056	0.002209	0.998073	0.001753	0.338749	0.940510
$\mathcal{U}$[-0.01, 0.01]	0.000048	0.002078	0.998344	0.001927	0.189490	0.934297
$\mathcal{U}$[-0.03, 0.03]	0.000045	0.001360	0.998443	0.001789	0.302790	0.939179
$\mathcal{U}$[-1, 1]	0.000050	0.001829	0.998291	0.002136	0.262704	0.927557

TABLE II: LSTMTF MSE and SKL prediction performance with various sample rates for training and different rates for the observed/predicted previous output value fed as input. This experiment was performed over 990,000 data points.

Missing Values Rate		**Testing**									
		0.01		**0.02**		**0.1**		**0.5**		all	
		MSE	**SKL**	**MSE**	**SKL**	**MSE**	**SKL**	**MSE**	**SKL**	**MSE**	**SKL**
Training	**0.01**	0.000057	0.002072	0.000060	0.002442	0.000285	0.003040	0.001385	0.025248	0.001717	0.03250
	0.02	0.000319	0.009235	0.000671	0.015010	0.004982	0.030222	0.020681	0.354019	0.022821	0.48674
	0.1	0.010525	0.062329	0.015951	0.070772	0.043521	0.356247	0.060211	0.963892	0.061859	1.05310
	0.5	0.029262	0.547036	0.020187	1.280027	0.016595	1.400291	0.015822	1.663186	0.015630	1.94703

ability in real-life scenarios.

Data Sample Rate Analysis Results

A key distinction between this set of experiments and the previous one is that only the observed values are fed to the models at varying rates. Specifically, the models perform one-step-ahead predictions using the observed values as input and not the previously predicted values. Tables III and IV show the experimental results for both the LSTMTF and LSTM models, measured utilizing the MSE and SKL metrics.

Similarly to the previous set of experiments, the models trained with the data sampled at a 0.01 rate obtained the best overall results in terms of SKL for the LSTMTF model. For the standard LSTM model the variability in the results was higher. In terms of MSE there were some cases where models trained with a lower sample rate obtained better results. For example, the model trained with the 0.05 rate obtained better results when tested against data sampled at the 0.1 rate. The results also revealed a considerable decrease in SKL compared to the previous set of experiments, especially for the 0.5 rate. Tables III and IV show a general downward trend in performance as the sample rate increases, confirmed for both models. Overall, the LSTMTF model outperformed the standard LSTM model in all cases.

E. Parameter Influence and Non-stationarity Analysis Results

The Lorenz system parameters are:

- For the parameter influence analysis: $\sigma = 10$, $\beta = 8/3$, $\rho = (25, 50, 140, 200)$, $\rho = [25 : 200]$ step 25, and $\rho = [25 : 200]$ step 40,
- For the non-stationarity analysis: $\sigma = 10$, $\beta = 8/3$, $\rho_0 = (50, 100, 150)$, $\rho_1 = 6$, $\tau = 30$, $\gamma = 0.03$, $\mu = 0.2$ (as they appear in Equations 17 and 16).

For the datasets with multiple ρ and ρ_0 values, 10 simulations were performed for each value. As previously stated, the parameter values were chosen following the recommendations in[16] and[15].

Parameter Influence Analysis Results

Recall that in this scenario, the performance of the models was tested in scenarios where training was conducted using data generated using fixed ρ parameter values and tested with observations generated using different values for ρ. The values for the parameter were 25, 50, 140, 200, and a set in which the observations were generated with the full range of values [25, 200] in increments of 25. In this final scenario, 10 simulations were performed for each ρ value. Additionally, models were trained and tested with observations generated using ρ values both inside and outside the chaotic region and chaotic transitions, with increases and decreases of 10% and 20%, respectively. These regions were originally identified by Patel et al.[16].

The results are shown in Tables V - VIII, and visually illustrated in Figure 5. Figure 5 shows the prediction performance in terms of observed and predicted values for both models, e.g., LSTM and LSTMTF over 500 and 2000 time steps. Here, the training data were generated using a ρ value of 140 while the testing data were generated with an increase of $\rho \pm 20\%$. As observed, the predictions of the LSTMTF model remain

TABLE III: LSTMTF MSE and SKL prediction performance with various sample rates for training and testing. This experiment was performed over 990,000 data points and at each step the previous observed output value was fed as input.

Sample Rate		Testing LSTMTF							
		0.01		0.02		0.1		0.5	
		MSE	SKL	MSE	SKL	MSE	SKL	MSE	SKL
Training	**0.01**	0.000057	0.002072	0.000428	0.004020	0.021097	0.005559	0.076294	0.036720
	0.02	0.000279	0.009252	0.000071	0.002532	0.019940	0.012317	0.076407	0.074320
	0.1	0.010545	0.070649	0.010164	0.081778	0.000323	0.009805	0.070399	0.178227
	0.5	0.036476	0.919100	0.032298	0.669868	0.031136	0.406737	0.006140	0.195781

TABLE IV: Standard LSTM MSE and SKL prediction performance with various sample rates for training and testing. This experiment was performed over 990,000 data points (100 simulations).

Sample Rate		Testing Standard LSTM							
		0.01		0.02		0.1		0.5	
		MSE	SKL	MSE	SKL	MSE	SKL	MSE	SKL
Training	**0.01**	0.001980	0.319153	0.004181	0.450796	0.023820	2.430288	0.040091	1.468175
	0.02	0.007885	0.359545	0.001268	0.166016	0.023505	2.691661	0.046363	1.391035
	0.1	0.048826	0.640523	0.037939	0.502226	0.001038	0.076833	0.051433	1.357317
	0.5	0.013032	2.857927	0.012042	2.371786	0.013619	1.104351	0.005587	0.275596

consistent even when the behavior of the system is affected due to changes in the parameter values. Moreover, examining at the results in Tables V and VI it can be seen that the performance of the model is not affected compared to the reference values. In the case where the ρ values increased by 20% in the testing data, the model outperformed the reference model. In contrast, the standard LSTM model performance decreased in both these cases, this was confirmed by all utilized metrics, not only compared to the LSTMTF model but also the reference LSTM model.

Moving forward, Tables VII and VIII show the remainder of the results. The numbers show the limits of the LSTMTF model in terms of the values of the ρ parameters. For example, when training $\rho = 25$, all three metrics show a steady increase with the values of ρ in testing. Compared to the other cases, for $\rho = 25$ the model exhibits the highest sensitivity to changes in the test data. The models trained with $\rho = 140$ and $\rho = 200$ obtained a lower sensitivity. Another interesting observation is that when the model is trained with data generated using a wider range of ρ values, the model performs better, even compared to the reference model. This, again, is confirmed by the metrics utilized. This could be explained by the fact that in this scenario the training data size increased from 15,000 to 49,000 to encompass all ρ values.

Two points of discussion can be initiated from these results. First, if the scope of the deployed models is to detect changes in the functionality of the system, namely to increase the sensitivity to changes, training with less parameter values can be chosen. Second, if steady performance is desired over a larger set of parameter values, without disruptions in functionality, training the models with multiple parameter values is a possible approach. A possible scenario here is the decrease of false alerts in a monitoring system due to expected changes in parameters during normal conditions. Moreover, results confirm that training the model with multiple parameter values can increase the prediction performance. However, as the results showed, in the case of the Lorenz system, the selection of ρ values for data generation is crucial for the performance of the models.

Non-stationarity Analysis Results

Figure 6 illustrates the observed and predicted values of the LSTMTF model over 250 and 6000 time steps. The observations were generated using time-dependent ρ values as described by Equation 17. Notably, the system behavior changes compared to the case with fixed ρ values, as shown in Figure 4. In Figure 6, an observable upward trend is present, caused by the dynamic ρ values. However, as shown in the figure, the model prediction remains close to the observed values.

The results are further presented in Tables IX and X. In Table IX, the model was trained with data generated using various fixed ρ values and tested using time-dependent ρ values. As observed, when ρ was set to 25, the model yielded the worst performance with MSE values of 0.203703, SKL values of 0.121628 and R^2 values of 0.834381. In particular, when the ρ values increased for training, the performance of the model improved considerably. For example, for $\rho = 140$, the MSE values decreased to 0.000268, the SKL values decreased to 0.014870 and the R^2 values increased by more than 15% reaching 0.991480. In general, for both scenarios, the second best performance was obtained when the models were

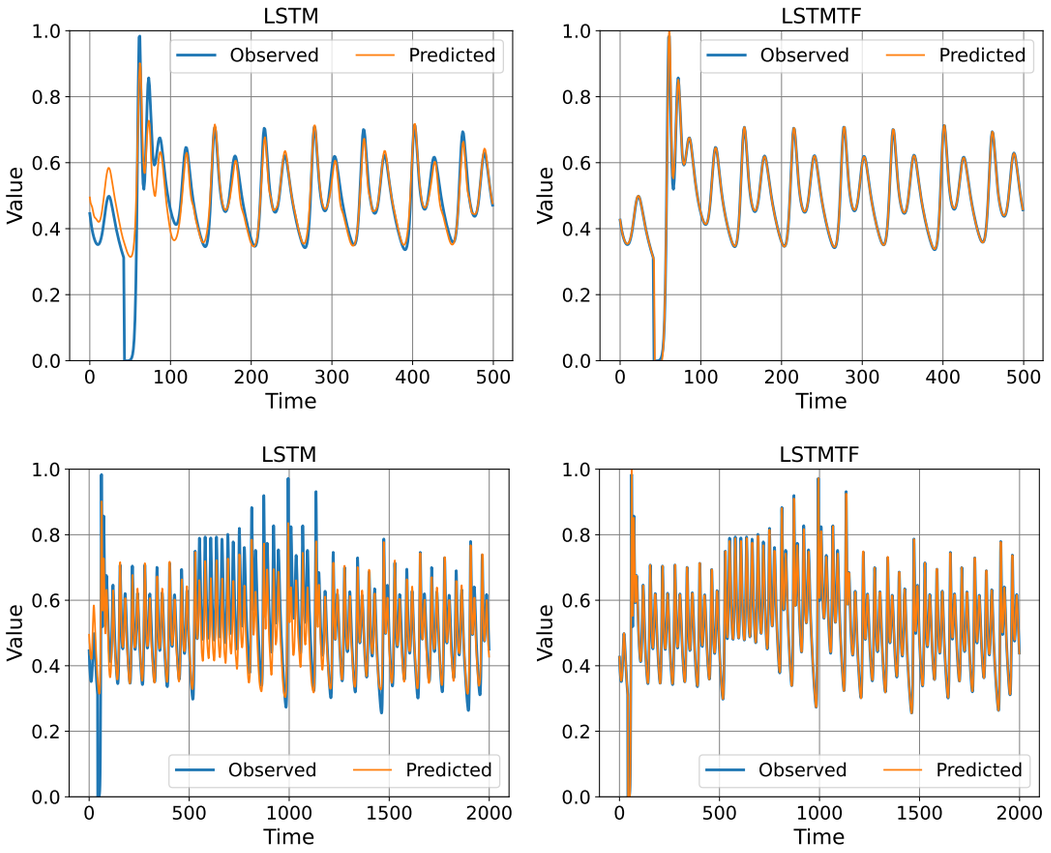

FIG. 5: Prediction performance illustration of the LSTM and LSTMTF models with dynamic $\rho = 140 \pm 20\%$ over 500 and 2000 time steps.

TABLE V: MSE, SKL and R^2 values for LSTM, LSTMTF with dynamic ρ values. The models are trained wih 15,000 data points generated with ρ = 140 and tested with $\rho \pm 10\%$ values in and outside of the chaotic region ρ = [126, 150].

Step $\rho = 140 \pm 10\%$	**MSE**	**SKL**	R^2
LSTM	0.004398	0.458290	0.858050
LSTMTF	0.000079	0.001102	0.997437

trained with ρ values of 140, 200 and using a wider range of values between 25 and 200. The best performances were achieved by models trained with data generated using multiple ρ values. Despite being trained on only four ρ values, these models performed as well as, or even outperformed, the reference models when handling unseen non-stationary data. These results are confirmed using all the metrics and in both testing scenarios. Table X shows the prediction performance of the model on data generated using time-dependent ρ values, as described in Equation 17, with fixed ρ_0 values of 50, 100, and 150. Although the values of ρ are very different depending on ρ_0 the model performance is largely unaffected by the different values of ρ_0, with little or no change observed in all three cases for all metrics (see Table X).

F. LSTMTF Gate Analysis Results

The parameters used for the generation of the data sets are as follows: $\rho = 28$, $\sigma = 10$, and $\beta = 8/3$.

LSTM Models Gate Activations Analysis Results

The results of this experiment are visually illustrated in Figures 7 and 8. Here, the gate values alongside the long- and short-term memory values are shown over 3000 data points from the testing set for four LSTM and LSTMTF cells.

As observed in the two figures, the range of gate values is shorter for the LSTMTF model, ranging from 0.17 to 0.82. In contrast, for the standard LSTM model, the gate values range from 0.2 to 1. For example, the activations for the input gates are often times saturated at 1. The same behavior is observed for the long- and short-term memory values, e.g., the hidden

TABLE VI: MSE, SKL and R^2 and values for LSTM, LSTMTF with dynamic ρ values. The models are trained with 15,000 data points generated with $\rho = 140$ and tested with $\rho \pm 20\%$ values inside and outside of the chaotic region and chaotic transitions $\rho = [114, 166]$, as originally identified in[16].

Step $\rho = 140 \pm 20\%$	**MSE**	**SKL**	R^2
LSTM	0.003474	0.214538	0.805274
LSTMTF	0.000046	0.002678	0.997394

TABLE VII: LSTMTF prediction MSE values when trained and tested with observations generated with various values for ρ. The testing **[25:200]** includes the values of ρ with increments of 25 in the $[25, 200]$ range. The model was trained with 15,000 data points for $\rho = (25, 90, 140, 200)$. For ρ in the $[25, 200]$ range, the model was trained with 49,000 data points. The model was tested with 990,000 data points for each ρ.

Train / Test ρ	**25**	**50**	**140**	**200**	**[25:200]**
25	0.000039	0.001763	0.477529	3.300690	0.565132
50	0.000016	0.000044	0.016334	0.096852	0.024754
140	0.000080	0.000111	0.000036	0.001187	0.000387
200	0.000115	0.000213	0.000215	0.000035	0.000392
[25:200]	0.000011	0.000016	0.000040	0.000059	0.000023

TABLE VIII: LSTMTF prediction SKL and R^2 values when trained and tested with observations generated with various values for ρ. The testing **[25:200]** includes the values of ρ with increments of 25 in the $[25, 200]$ range. The model was trained with 15,000 data points for $\rho = (25, 90, 140, 200)$. For ρ in the $[25, 200]$ range, the model was trained with 49,000 data points (1 simulation worth of data per ρ step of 40). The model was tested with 990,000 data points for each ρ value.

Train / Test ρ	**25**		**50**		**140**		**200**		**[25:200]**	
	SKL	R^2	**SKL**	R^2	**SKL**	R^2	**SKL**	R^2	**SKL**	R^2
25	0.001598	0.998747	0.030168	0.979292	1.743404	0.097873	3.867824	1.597241	0.784773	0.713122
50	0.005817	0.997570	0.002969	0.997607	0.232316	0.860230	0.964586	0.504762	0.142665	0.943082
140	0.127895	0.900972	0.062946	0.949082	0.002697	0.997343	0.156222	0.947693	0.027210	0.992328
200	0.196773	0.702921	0.102884	0.796366	0.087516	0.966984	0.003938	0.996723	0.040996	0.983817
[25:200]	0.003929	0.996322	0.004193	0.993764	0.009293	0.993721	0.004164	0.994526	0.000805	0.998805

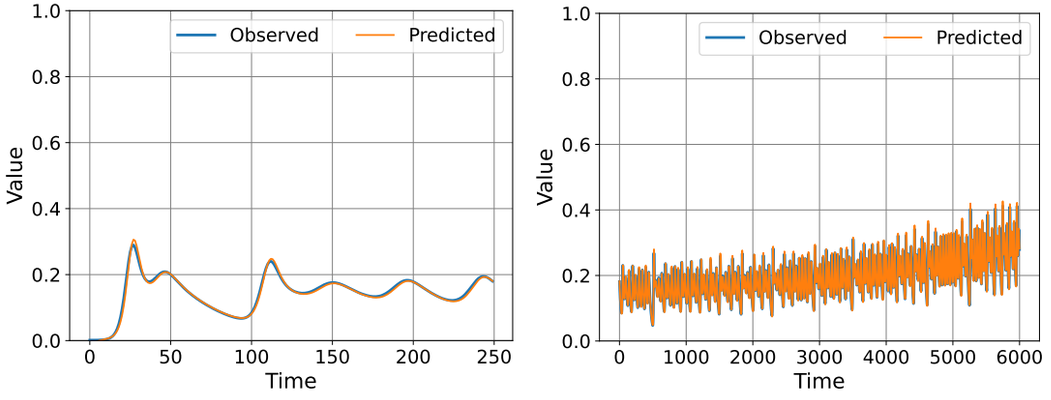

FIG. 6: Prediction performance illustration of the LSTMTF model with dynamic ρ values over 250 and 6000 time steps.

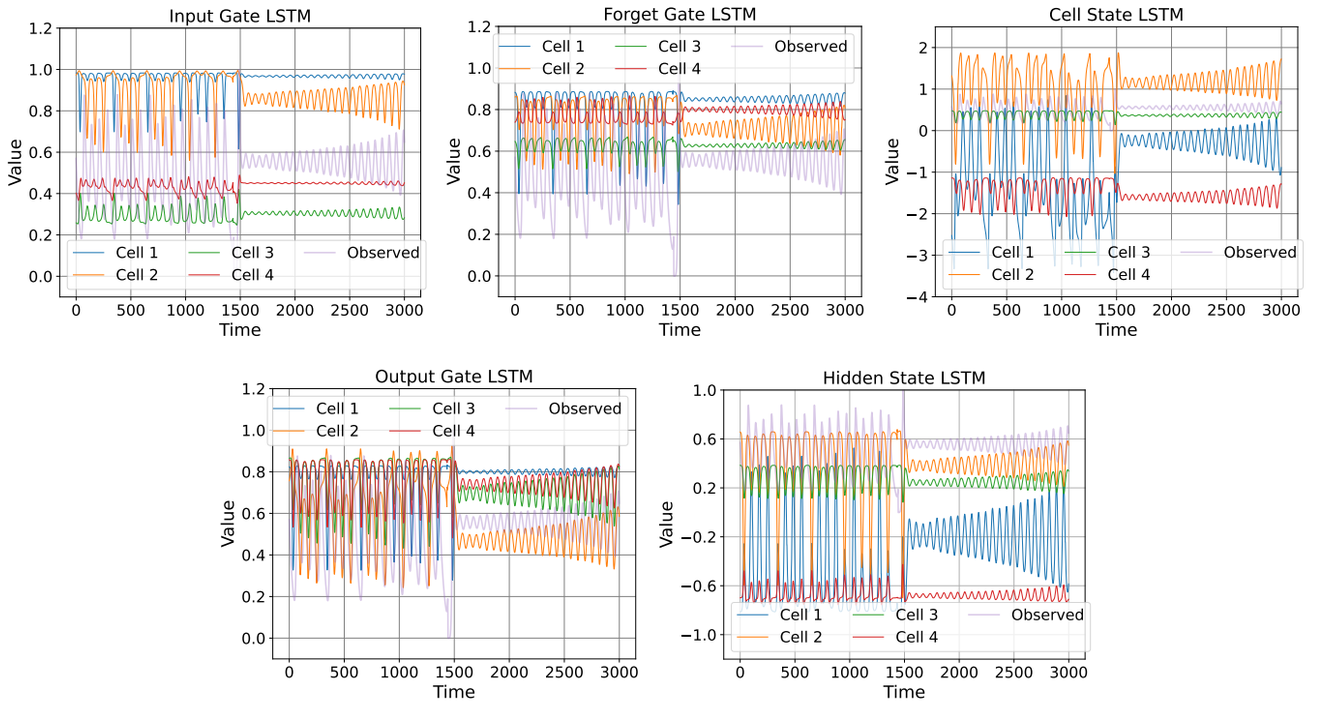

FIG. 7: Illustration of the standard LSTM gate activations and hidden states over 500 data points from the testing set.

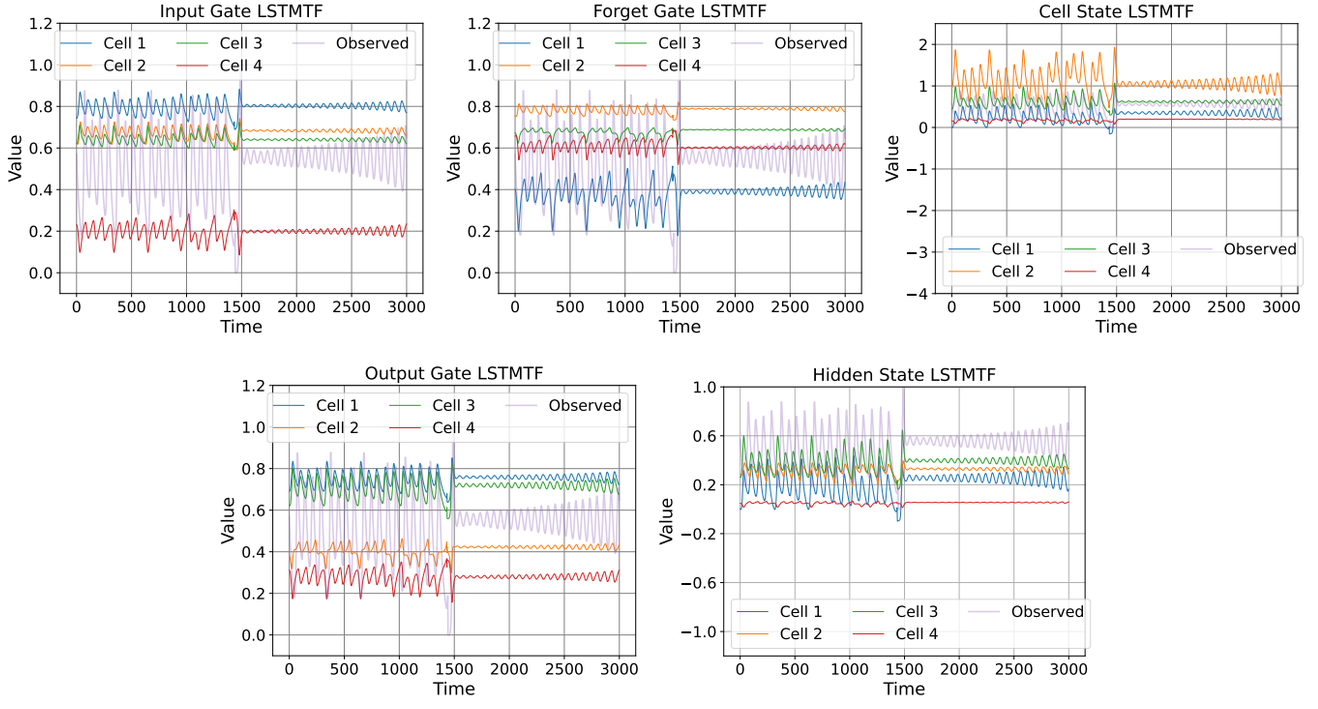

FIG. 8: Illustration of the LSTMTF gate activations and hidden states over 3000 data points from the testing set.

TABLE IX: LSTMTF prediction MSE, SKL and R^2 values when trained with observations generated with various values for ρ. The **[25:200]** includes the values of ρ with increments of 40 in the $[25, 200]$ range. The model was tested against data generated using the time-dependent ρ parameter values as shown in Equations 16 and 17.

Training ρ	**Testing ρ using Equation 16**			**Testing ρ using Equation 17**		
	MSE	**SKL**	R^2	**MSE**	**SKL**	R^2
25	0.203703	0.121628	0.834381	0.013371	0.201872	0.909006
50	0.011142	0.054469	0.958965	0.000277	0.010149	0.991459
140	0.000268	0.014470	0.991480	0.000259	0.066388	0.931208
200	0.000358	0.023590	0.976382	0.000199	0.124269	0.889596
[25:200]	0.000079	0.000802	0.995577	0.000021	0.002315	0.990015

TABLE X: LSTMTF prediction SKL and R^2 values when trained and tested with observations generated with time-dependent ρ using the sin() function. The model was trained with 49,000 data points for $\rho = 25, 65, 105, 145, 185$ and tested with 990,100 data points for each experiment with ρ_0 values of 50, 100 and 150.

Train / Test ρ	$\rho_0 = 50$			$\rho_0 = 100$			$\rho_0 = 150$		
	MSE	**SKL**	R^2	**MSE**	**SKL**	R^2	**MSE**	**SKL**	R^2
[25:200] / 40	0.000023	0.002731	0.989036	0.000023	0.002908	0.989072	0.000023	0.002722	0.989023

and cell lstate. For the standard LSTM model the values range from -3 to 2 while for the LSTMTF model, from 0 to 1.8. Additionally, the shape of the hidden state signal, which is the cell output, closely resembles the actual observed output in the case of the LSTMTF model.

A closer analysis of Figures 7 and 8 reveals that, depending on the gates, for the LSTMTF model, the signal values are in antiphase with the observed signal. This is clearly visible at the 1500th time step in Figure 8. For the input gate, three cells follow the same shape, while one cell is in antiphase. Similarly, in the case of the forget gate, three cells are in antiphase. In the case of the output gate, the signal values in three cells follow the same shape as the observed values. Another aspect worth noting is that, more visible in the case of the LSTMTF model, the cell activation values for the input, forget, and output gates tend to be either above or below the actual values, averaging to the observed signal values. An interesting observation is that, in the standard LSTM model, the gate activation values appear to be independent of each other, while in the LSTMTF model, all cells seem to exhibit similar behaviors, sometimes in antiphase. However, it is particularly intriguing to observe and analyze the behavior of these black-box models at the gate level.

LSTMTF Gate Modifications Analysis Results

The results of the second set of experiments are numerically shown in Table XI, where the previously observed output value is fed as an additional input only to certain LSTMTF gates. In Table XI, this was indicated with "gates affected" as the gates are not disabled, but rather a column of the weight matrices is removed. For this experiment, noise-free variants of the datasets were used for both training and testing. The training dataset includes 15,000 observations, while the testing set consists of 100 simulations, totaling 990,900 observations.

A first observation that can be drawn from analyzing the results table is that the additional input must be fed to at least the cell candidate gate. The worst results are obtained when this value is removed from this gate, or in combinations containing this gates. For example, the SKL values were higher for the subsequent combination of gate input-cell, forget-cell, cell-output, input-forget-cell, input-cell-output, and forget-cell-output gates.

Conversely, considering both metrics, MSE and SKL, the performance of the model remains closely similar when the previously observed output value is not fed into the input, forget, output, and the combination of input-forget-output gates. For example, in terms of MSE, when the input gate was affected, the model performed better than the reference model, with values of 0.000036 compared to the reference value of 0.000057. Similar behavior is observed for the combination input-forget-output. Interestingly, when the input gate is affected, an increase in performance is observed when measuring SKL and R^2 with a value of 0.001683 and 0.998509 compared to the reference value of 0.002072 and 0.998056.

In realistic scenarios, not feeding the previously observed output value to certain gates has the advantage of reducing the complexity. For each LSTM cell, for each gate, this translates to reducing each weight matrix by one column. As observed in the previous tables, for each cell, there are cases where the additional input is only required in one gate. Furthermore, if model training is performed on devices with an increased computational capability, reducing the complexity of the models during inference can be advantageous if such models are deployed on embedded devices.

Overall, the best performing architectures considering all the metrics are the models in which the input, forget, out-

TABLE XI: LSTMTF testing MSE, SKL, and R^2 values with the gates affected and disabled. Metrics are computed over 990,000 data points (100 simulations) from the testing set.

Gates	Gate Affected			Gate Disabled		
	MSE	SKL	R^2	MSE	SKL	R^2
Ref LSTMTF	0.000057	0.002072	0.998056	0.000057	0.002072	0.998056
Ref LSTM	0.001726	0.201782	0.941223	0.001726	0.201782	0.941223
Input	0.000036	0.001683	0.998509	0.000037	0.000801	0.998734
Forget	0.000057	0.001994	0.998048	0.000293	0.001754	0.990021
Cell	0.000047	0.001942	0.998373	0.000289	0.002481	0.990146
Output	0.000066	0.003040	0.997723	0.000051	0.001720	0.998241
Input, Forget	0.000043	0.001518	0.998511	0.014440	0.525202	0.508293
Input, Cell	0.000069	0.003947	0.997647	0.029368	inf	-1.633756
Input, Output	0.000051	0.001833	0.998248	0.000084	0.004254	0.997129
Forget, Cell	0.000079	0.003505	0.997297	0.000304	0.001204	0.989643
Forget, Output	0.000047	0.001520	0.998398	0.256552	4.868235	-7.735846
Cell, Output	0.000055	0.003639	0.998093	0.000447	0.010622	0.984761
Input, Forget, Cell	0.000115	0.006507	0.996060	0.029368	inf	-1.633756
Input, Forget, Output	0.000052	0.002085	0.998225	0.074341	0.029408	57.24442
Input, Cell, Output	0.000169	0.012539	0.994219	0.029368	inf	-1.633756
Forget, Cell, Output	0.000068	0.009677	0.997669	0.029485	inf	-0.004001

put, and input-forget-output gates were affected. These architectures are selected as candidates for the next experiments, where the performance will be tested under various conditions.

LSTMTF Gate Disabling Analysis Results

In this scenario, different combinations of cell gates were disabled. The results in terms of MSE, SKL, and R^2 are shown in Table XI. Furthermore, Figure 9 illustrates the observed and predicted values of the LSTMTF model with the disabled gates over unseen 500 data points. For this experiment, noise-free variants of the datasets were used for both training and testing. The training dataset includes 15,000 observations, while the testing set consists of 100 simulations, totaling 990,900 observations.

An important observation can be drawn from Table XI, specifically the anomalous results in various combinations involving the cell candidate gate. In nearly all experiments where the cell candidate gate was disabled, the predicted values remained constant, leading to similar MSE values and anomalous results across other metrics. A similar pattern was also observed in the previous set of experiments, indicating that the cell candidate gate appears to be the most critical element for now.

The results show that the gates can be individually disabled without significant variations in performance. For example, disabling the input gate yielded better MSE values, decreasing from 0.000057 to 0.000037 while the SKL values decreased from 0.002072 to 0.000901. Similarly, the R^2 values improved from 0.998056 to 0.998734. The only gate that yielded worst SKL values when disabled was the cell candidate gate.

Mowing forward, the best combination of two disabled gates was the input and output. In this scenario, the MSE values increased from 0.000057 to 0.00084, while the SKL values increased from 0.002072 to 0.004254. In terms of R^2 values, this combination of gates yielded a small decrease from the reference value of 0.998056 to 0.997129. Although this study analyzed every possible scenario, including cases where three of the four gates were disabled, as shown in Table XI, most of the results were anomalous. Moreover, in a realistic scenario, the disablement of three gates effectively reduces the LSTM to a simple recurrent model, breaking the purpose of the gating mechanisms.

Considering all the results from this set of experiments, across all metrics and scenarios, the following candidates have been selected for further experimental assessment: models with the input, forget, output, and input-output gates, disabled.

V. REDUCED LSTMTF ARCHITECTURE RESULTS AND DISCUSSIONS

In the final set of experiments, a new potential LSTMTF architecture is tested and validated using the results from subsection III E as input. To validate the new architecture, the experiments of subsections III C and III D are applied to these new configurations.

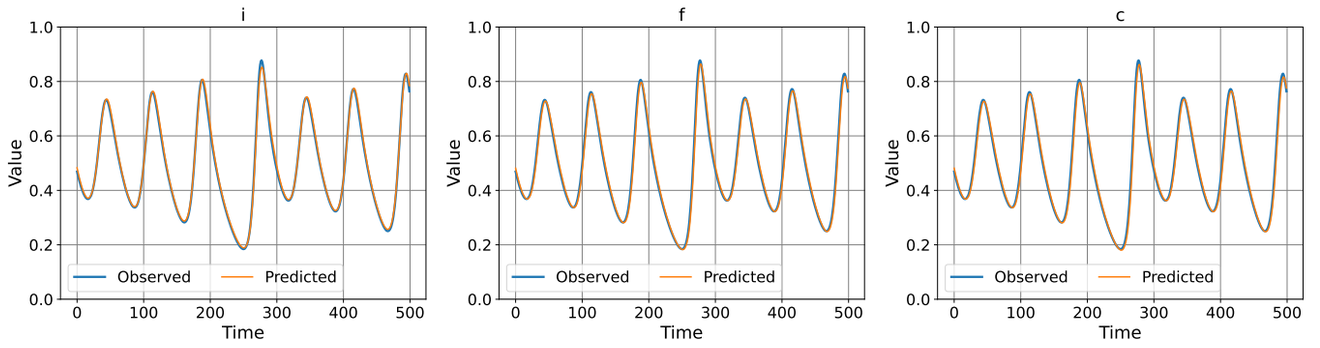

FIG. 9: Prediction performance illustration of the LSTMTF model over 500 samples from the unseen testing set with gates disabled during training and testing. In the figures i = input gate, f = forget gate, c = cell candidate gate.

A. Results

The candidates for this set of experiments are the models that achieved the best results in the gate analysis section, as illustrated in Table XI. The list of candidates includes the following:

i) Models with the following gates affected (e.g., the previous output feedback disabled in the gate): Input (I), Forget (F), Output (O), and a combination of Input-Forget-Output (IFO).

ii) Models with the following gates disabled: Input (I), Output (O), Forget (F), and Input-Output (IO).

Table XII shows the results for the previous set of experiments performed on the candidates described above and the reference model. In terms of affecting gates, specifically feeding the previously observed output value to only certain gates, the results illustrate that most of the architectures obtained similar results to the reference model. When this previously observed output value was fed only to the cell gate, there was an increase in MSE and SKL for the parameter influence experiments (column E4). In the scenario where the system parameters were not affected (column E1), all models obtained similar results, showing that under normal operating conditions, the complexity of the LSTMTF model can be reduced without sacrificing performance. Similar results were obtained for the forecasting, sample rate, and non-stationarity experiments. For the forecasting experiment, the worst results in terms of SKL were obtained when the output gate was affected, with an increase from 0.515222 to 0.796256. In addition, when the sampling frequency was changed (column E3), the same behavior was observed for all models, without significant performance variations. Overall, the results reveal that with the forget gate affected, the results remained close to the reference model.

In the scenario where the LSTMTF gates were disabled, similar results were obtained for the noise effects set of experiments. Even with two disabled gates, Input and Output, the models outperformed the reference model. As shown in Table XII, for the forecasting, sample rate, parameter influence, and non-stationarity experiments a small decrease in average performance was observed mainly in SKL values. For the forecasting, sample rate, and parameter influence experiments (columns E2, E3 and E4) the model with the forget gate disabled obtained the worst SKL values. In contrast, the model with the input and output gates disabled obtained the best SKL scores and similar MSE values for the first scenario, as shown in column E1 of the same table. Overall, the results indicate that deactivating the input gate had less impact on model performance compared to other gates. In fact, without the input gate, the model performed better, as demonstrated in columns E1 and E2 for both the noise and forecasting experiments, outperforming even the reference model and the models with other gates disabled.

An interesting result concerns the disabled forget gate, where in some experiments, the performance of the models was severely impacted (columns E2, E3 and E4). Greff et al.[19] obtained similar results for the LSTM classification of speech and handwriting recognition, and music modeling, confirming the necessity of the forget gate in multiple scenarios, including chaotic system modeling.

Taking into account the results of the previous sections, the results of Table XII, and the recommendations from the results sections, the following new LSTMTF architecture is proposed and evaluated. An LSTMTF with the input gates completely disabled and forget gate affected (without feedback). A visualization of the cell architecture is shown in Figure 10. To evaluate and validate this new model, the previous experiments are repeated, as illustrated in Table XII and presented in the previous sections.

The results for this new architecture are shown in Table XIII. As seen in the table, this new proposed architecture obtained similar results to the reference model for almost all experiments. The only experiment where this architecture performed worse than the reference model is the parameter influence scenario in terms of MSE and SKL values. For the non-stationarity set of experiments, the model yielded results similar to the reference model and outperformed the other architectures analyzed (see column E5 in Table XII). Noise effects analysis revealed that this new architecture performed similarly to the reference model, with slightly better SKL scores

TABLE XII: The LSTMTF prediction MSE and SKL mean values for the best-performing candidates selected from previous experiments. Affected = The extra input disabled in that specific gate. Disabled = The entire gate is disabled. E1 = Noise experiment, E2 = Forecasting experiment, E3 = Sample rate experiment, E4 = Parameter influence experiment, E5 = Non-stationarity experiment.

	Candidates	E1		E2		E3		E4		E5	
		MSE	SKL	MSE	SKL	MSE	SKL	MSE	SKL	MSE	SKL
Gates Affected	Ref.	0.000047	0.001722	0.017127	0.515222	0.024503	0.167421	0.000029	0.004476	0.000050	0.001558
	I	0.000048	0.001593	0.032902	0.645875	0.025343	0.167891	0.000077	0.005481	0.000051	0.001914
	F	0.000050	0.001900	0.017205	0.641606	0.025541	0.171307	0.000073	0.004903	0.000055	0.002091
	O	0.000053	0.002122	0.019896	0.796256	0.025524	0.188275	0.000085	0.005325	0.000055	0.002103
	IFO	0.000048	0.001824	0.030023	0.657557	0.024939	0.175496	0.000076	0.007075	0.000054	0.001919
	Candidates	E1		E2		E3		E4		E5	
		MSE	SKL	MSE	SKL	MSE	SKL	MSE	SKL	MSE	SKL
Gates Disabled	Ref.	0.000047	0.001722	0.017127	0.515222	0.024503	0.167421	0.000029	0.004476	0.000050	0.001558
	I	0.000032	0.000987	0.011473	0.745007	0.025302	0.228768	0.000092	0.018285	0.000059	0.002424
	F	0.000032	0.001023	0.023976	1.390861	0.052182	1.042909	0.041156	14.54777	0.000106	0.003008
	O	0.000032	0.001076	0.250687	2.750450	0.034132	0.423908	0.000076	0.027906	0.000060	0.002562
	IO	0.000033	0.000970	0.015692	0.981150	0.050588	0.942535	0.000291	0.101289	0.000116	0.003870

TABLE XIII: The LSTMTF prediction MSE and SKL mean values for the proposed LSTMTF architecture with the input gate disabled and the forget gate affected. Affected = The extra input disabled in that specific gate. Disabled = The entire gate is disabled. E1 = Noise experiment, E2 = Forecasting experiment, E3 = Sample rate experiment, E4 = Parameter influence experiment, E5 = Non-stationarity experiment.

Model	E1		E2		E3		E4		E5	
	MSE	SKL	MSE	SKL	MSE	SKL	MSE	SKL	MSE	SKL
Ref.	0.000047	0.001722	0.017127	0.515222	0.024503	0.167421	0.000029	0.004476	0.000050	0.001558
New LSTMTF	0.000051	0.001063	0.017167	0.676256	0.026140	0.238532	0.000089	0.013964	0.000052	0.002181

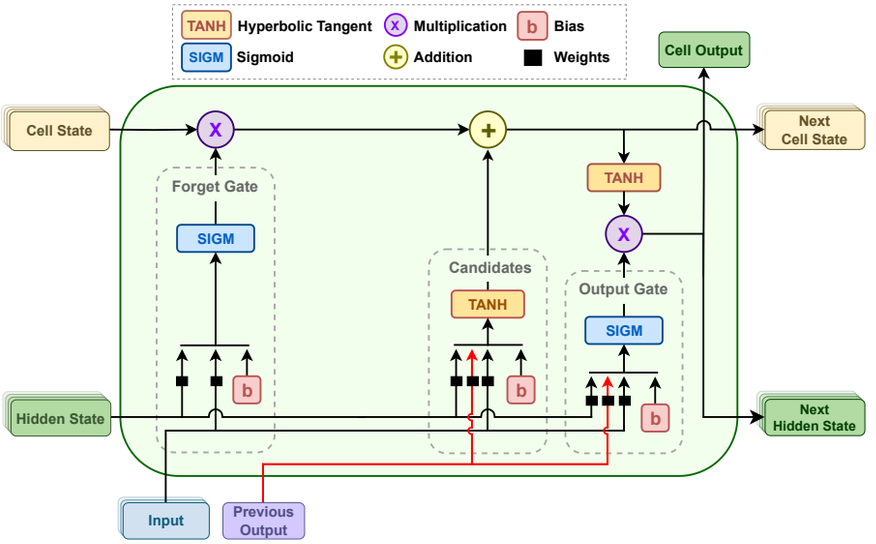

FIG. 10: Illustration of the new proposed architecture, with the input gate disabled and the previously observed output value fed only to the cell candidate and output gates.

and similar MSE values. In general, this new LSTMTF cell architecture outperformed the other models, as shown in Table XII and performed equally well to the reference LSTMTF model, with all four gates active, and full feedback from the previous observed output value. This final set of evaluations demonstrated the capabilities of the proposed model, illustrating that the LSTMTF architecture can be reduced whilst maintaining performance for MISO non-linear systems modeling.

B. Extended Evaluation Results

For additional validation, the standard LSTM, LSTMTF, and reduced LSTMTF model were also tested on a dataset generated using the Rössler system[38]. The Rössler system (or Rössler attractor) is defined by a set of three differential equations with three parameters, as shown in Equation 25, that describe a continuous-time system with low-dimensional chaos and unbounded dynamics.

$$\begin{cases} \dfrac{dx}{dt} = -y - x \\ \dfrac{dy}{dt} = x + ay \\ \dfrac{dz}{dt} = b + z(x - c). \end{cases} \tag{25}$$

This dataset was generated using the same methodology described in Section III B, similar to the approach used for the Lorenz case. As shown by Cetin et al.[39], with the parameters $a = 0.2$, $b = 0.2$, and c values above 4.2, the Rössler system transitions to chaotic dynamics. Consequently, for this set of experiments, the Rössler system was simulated using c values over 5 and in the range $[5, 45]$, as done by others[40–42]. The training dataset consists of observations generated with c values incremented by a step of 20 within this range, while the testing dataset is created using c values from the same range but with a step size of 5. Similarly to the Lorenz system, x and y were utilized as inputs, while z was used as output. Moreover, the standard LSTM, LSTMTF, and reduced LSTMTF hyperparameters used for the Lorenz system were similarly applied here.

TABLE XIV: LSTM, LSTMTF and reduced LSTMTF performance comparison on the Rössler system dataset.

reduced LSTMTF		**LSTMTF**		**LSTM**	
MSE	**SKL**	**MSE**	**SKL**	**MSE**	**SKL**
0.000002	0.040502	0.000002	0.039804	0.000846	0.949058

The results, as shown in Table XIV, illustrate the performance of the three models, standard LSTM, LSTMTF, and the reduced LSTMTF on the Rössler system testing dataset. As observed, the reduced LSTMTF model yields promising results further highlighting its effectiveness in various settings, not just on the Lorenz system.

C. Complexity Analysis

According to the original LSTM paper and other related studies[22,43], the computational complexity of the LSTM model per time step, per weight, is $\mathcal{O}(1)$, indicating that this model is local in space and time. Consequently, the overall complexity per time step is $\mathcal{O}(w)$, where w is the number of model parameters, including weights and biases. When we extend the complexity with the number of observations N and the number of epochs κ, the complexity becomes $\mathcal{O}(w \cdot N \cdot \kappa)$.

Consider a system with m measured signals, $m-1$ signals are used as inputs and 1 signal as output. For the standard LSTM model, the number of parameters w_{LSTM} is computed as the product of the number of gates and the sum inputs $m-1$, multiplied by the number of hidden units H, plus the square of the hidden units H^2 (e.g., hidden to hidden connections), and the bias term for the gate b_{gate}. Additionally, the number of hidden units that are connected to the output and the bias term for the output neuron are added to the product. The computation of w_{LSTM} is shown in Equation 26.

$$\begin{aligned} w_{\text{LSTM}} = \text{Gates} \times \left[(m-1) \times H + H^2 + b_{\text{gate}}\right] \\ + H_{\text{out}} + b_{\text{out}}. \end{aligned} \tag{26}$$

For the LSTMTF model, the number of parameters w_{LSTMTF} is computed as follows:

$$\begin{aligned} w_{\text{LSTMTF}} = \text{Gates} \times \left(m \times H + H^2 + b_{\text{gate}}\right) \\ + H_{\text{out}} + b_{\text{out}}. \end{aligned} \tag{27}$$

In Equations 26 and 27 the number of gates is 4. Next, for the reduced version of the LSTMTF model, by eliminating the input gate and removing the feedback to the forget gate, the number of parameters $w_{\text{reduced LSTMTF}}$ is computed as:

$$\begin{aligned} w_{\text{reduced LSTMTF}} = (\text{Gates - 2}) \times \left[m \times H + H^2 + b_{\text{gate}}\right] \\ + \left[(m-1) \times H + H^2 + b_{\text{gate}}\right] \\ + H_{\text{out}} + b_{\text{out}}. \end{aligned} \tag{28}$$

Consider the architecture of the models utilized in this paper: LSTM, LSTMTF, and reduced LSTMTF. The models feature 32 hidden units with 2 or 3 inputs and a single output. Utilizing the computation methods presented in Equations 26 - 28 results in 4513 parameters for the standard LSTM, 4641 parameters for LSTMTF, and, finally, 3457 parameters for the reduced LSTMTF. Compared to the standard LSTM, the computational complexity of the LSTMTF model increases by 2.84%. In contrast, the reduced LSTMTF model achieves an 30.54% reduction in computational complexity relative to the standard LSTM and 34.24% relative to LSTMTF. Nevertheless, as shown in Tables XI and XIV, both the LSTMTF and reduced LSTMTF models demonstrate up to $\sim 99.76\%$ error reduction compared to the standard LSTM in terms of both MSE and SKL.

Regarding spatial complexity, the LSTM model is characterized by the number of parameters and initial states. This can be expressed as a spatial complexity of $\mathcal{O}(w + |\mathbf{s}|)$, where

s represents the vectors of initial states, which remain constant regardless of the LSTM variant used. Typically, $|\mathbf{s}|$ is equal to 2H (for the cell and hidden states). Based on these considerations, it becomes obvious that the reduced LSTMTF model offers an advantage in terms of spatial complexity, in addition to the previously demonstrated performance improvements.

D. Discussions

The results, as shown in Table XIII, illustrate the utility and efficiency of the models with a reduced architecture as follows. If such models are employed for anomaly detection or monitoring tasks, the overall increase in the prediction residuals, as shown when measured with both metrics (e.g., MSE and SKL), can be useful. As the parameters of the system change and, consequently, the behavior of the system, the prediction residuals also increase. This is useful when designing a detection system that monitors any changes or deviations from the normal operating conditions, as the ones proposed in[9,44]. This can be observed in Table XIII under columns E4 and E5. Conversely, if parameter changes are expected or occur naturally, retraining might be necessary, or the initial training may need to include a larger data pool that encompasses multiple parameter values, see Tables VII, VIII and IX.

Alternatively, as shown in column E1 in Table XIII, even with the addition of noise, which can occur naturally with real-life measurements, the performance of the model is not negatively affected. Regarding the longer-term forecasting or sample rates change during inference, the performance is only slightly affected, as illustrated in XIII, in columns E2 and E3.

In general, the final experimental assessment demonstrated that the LSTMTF architecture can be simplified without sacrificing performance. This reduction in complexity naturally decreases the time and space requirements of the models, which is particularly beneficial for the implementation of devices with limited computational resources, such as the approach from[9], or when used in ensembles that encompass a large number of models. It also highlighted some limitations and the need for additional retraining, or more inclusive training datasets. It is worth mentioning that this study analyzed only regression approaches for nonlinear chaotic systems, and these results might not highlight the performance of such models for other tasks, including classification or natural language processing.

The main scope of this article was to explore LSTM variants for chaotic system modeling, with the main focus on LSTM architectures. Nevertheless, the authors acknowledge the existence of other similar gated architectures in the literature e.g., the Gated Recurrent Unit (GRU)[45]. Since GRUs were originally proposed, a multitude of studies focused on performance comparisons between LSTMs and GRUs in a wide variety of domains[46–49]. Some of these studies showed slight performance improvements that favor both architectures in various contexts. However, Bengio Yosua, one of the coauthors of the original GRU paper, investigated the performance of LSTMs and GRUs in[50]. The results of this study illustrate the superior performance of gated architectures compared to simple RNNs, but no definitive conclusions were reached regarding which model is better. For now, this remains an open question and an interesting possibility for future exploration.

Today, we live in the age of Transformers[51], where Transformer-based approaches and pre-trained models are considered the state-of-the-art. Although transformers dominate the fields of natural language processing, speech analysis[52,53] and text classification[54], various studies reveal that simpler models, including LSTMs, obtain better results[55–57]. For example, in[57] the authors introduce simple one-layer linear models and demonstrate how these approaches surprisingly outperform existing sophisticated Transformer-based models, often by a large margin. This was also observed when Transformers were utilized for fault and anomaly detection[58]. In this study, the authors tested various supervised attention-based architectures and found that Transformers generally underperformed compared to LSTMs in most experiments.

VI. CONCLUSIONS

This paper analyzed the modeling performance of an enhanced LSTM-based architecture which utilizes the previously observed output value as an additional input, for MISO chaotic systems. In this direction, an extensive experimental assessment was performed, including LSTM cell gate analysis, noise effects, sample rate change, parameter value drifts, and long-term forecasting capabilities. The experimental assessment mainly focused on the Lorenz system while for additional validation, the evaluation was extended to the Rössler attractor. The enhanced LSTMTF variant yielded promising results, surpassing the capabilities of a standard LSTM model, even with a simple single hidden layered architecture. Additionally, a new reduced LSTMTF architecture is presented, which includes only three gates, with the input gate removed and output feedback not applied to the forget gate. The valuable results of this study can be utilized to deploy simple and efficient models with promising capabilities, including low sensitivity to noise. The results also illustrate the model's robustness to system parameter changes and non-stationarity. Additionally, as revealed by an extensive analysis, the proposed enhanced LSTMTF variant can be successfully utilized for both short- and long-term forecasting, even in the presence of missing values. In a nutshell, this paper remains a large-scale study on LSTMs utilized for chaotic systems modeling and closes an existing gap in the direction of in-depth LSTM-based performance and architectural analysis. As revealed by the experimental evaluation, a simplified, less complex, architecture can be successfully utilized without always requiring the use of complex deep learning methodologies. To enhance the experimental evaluation, computational and spacial complexity analysis were computed for three models i.e., standard LSTM, LSTMTF, and the reduced LSTMTF. Future work directions include the analysis of various other applications, including dynamical chaotic systems, and the development of independent and ensemble-based anomaly detection techniques specifically designed for these types of systems.

DATA AVAILABILITY STATEMENT

The data that supports the findings of this study are openly available at Harvard Dataverse: https://doi.org/10.7910/DVN/BKNNTK

CONFLICT OF INTEREST STATEMENT

The authors have no conflicts to disclose.

REFERENCES

[1]S. Yu, W. Chen, and H. V. Poor, "Real-time monitoring of chaotic systems with known dynamical equations," IEEE Transactions on Signal Processing **72**, 1251–1268 (2024).
[2]H. Lin, C. Wang, F. Yu, J. Sun, S. Du, Z. Deng, and Q. Deng, "A review of chaotic systems based on memristive hopfield neural networks," Mathematics **11**, 1369 (2023).
[3]A. E. Giakoumis, C. K. Volos, I. N. Stouboulos, I. M. Kyprianidis, H. E. Nistazakis, and G. S. Tombras, "Implementation of a laboratory-based educational tool for teaching nonlinear circuits and chaos," in *Advances and Applications in Chaotic Systems*, edited by S. Vaidyanathan and C. Volos (Springer International Publishing, Cham, 2016) pp. 379–407.
[4]V. Gupta, "Application of chaos theory for arrhythmia detection in pathological databases," International Journal of Medical Engineering and Informatics **15**, 191–202 (2023).
[5]F. Yu, W. Zhang, X. Xiao, W. Yao, S. Cai, J. Zhang, C. Wang, and Y. Li, "Dynamic analysis and fpga implementation of a new, simple 5d memristive hyperchaotic sprott-c system," Mathematics **11** (2023).
[6]S.-K. Yang, C.-L. Chen, and H.-T. Yau, "Control of chaos in lorenz system," Chaos, Solitons and Fractals **13**, 767–780 (2002).
[7]J. Schoukens and L. Ljung, "Nonlinear system identification: A user-oriented road map," IEEE Control Systems Magazine **39**, 28–99 (2019).
[8]R. Bolboacă and P. Haller, "Performance analysis of long short-term memory predictive neural networks on time series data," Mathematics **11** (2023).
[9]R. Bolboacă, "Adaptive ensemble methods for tampering detection in automotive aftertreatment systems," IEEE Access **10**, 105497–105517 (2022).
[10]M. Han, J. Xi, S. Xu, and F.-L. Yin, "Prediction of chaotic time series based on the recurrent predictor neural network," IEEE transactions on signal processing **52**, 3409–3416 (2004).
[11]Q. Li and R.-C. Lin, "A new approach for chaotic time series prediction using recurrent neural network," Mathematical Problems in Engineering **2016**, 3542898 (2016).
[12]S. Shahi, F. H. Fenton, and E. M. Cherry, "Prediction of chaotic time series using recurrent neural networks and reservoir computing techniques: A comparative study," Machine learning with applications **8**, 100300 (2022).
[13]J.-S. Zhang and X.-C. Xiao, "Predicting chaotic time series using recurrent neural network," Chinese Physics Letters **17**, 88 (2000).
[14]M. Liu-Schiaffini, C. E. Singer, N. Kovachki, T. Schneider, K. Azizzadenesheli, and A. Anandkumar, "Tipping point forecasting in non-stationary dynamics on function spaces," (2023), arXiv:2308.08794.
[15]D. Patel and E. Ott, "Using machine learning to anticipate tipping points and extrapolate to post-tipping dynamics of non-stationary dynamical systems," Chaos: An Interdisciplinary Journal of Nonlinear Science **33**, 023143 (2023).
[16]D. Patel, D. Canaday, M. Girvan, A. Pomerance, and E. Ott, "Using machine learning to predict statistical properties of non-stationary dynamical processes: System climate,regime transitions, and the effect of stochasticity," Chaos: An Interdisciplinary Journal of Nonlinear Science **31**, 033149 (2021).
[17]P. Dubois, T. Gomez, L. Planckaert, and L. Perret, "Data-driven predictions of the lorenz system," Physica D: Nonlinear Phenomena **408**, 132495 (2020).
[18]T. M. Breuel, "Benchmarking of lstm networks," arXiv preprint arXiv:1508.02774 (2015).
[19]K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," IEEE transactions on neural networks and learning systems **28**, 2222–2232 (2016).
[20]W. A. Nassan, T. Bonny, K. Obaideen, and A. A. Hammal, "An lstm model-based prediction of chaotic system: Analyzing the impact of training dataset precision on the performance," in *2022 International Conference on Electrical and Computing Technologies and Applications (ICECTA)* (2022) pp. 337–342.
[21]A. Farzad, H. Mashayekhi, and H. Hassanpour, "A comparative performance analysis of different activation functions in lstm networks for classification," Neural Computing and Applications **31**, 2507–2521 (2019).
[22]S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural computation **9**, 1735–1780 (1997).
[23]R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," Neural computation **1**, 270–280 (1989).
[24]P. J. Werbos, "Backpropagation through time: what it does and how to do it," Proceedings of the IEEE **78**, 1550–1560 (1990).
[25]R. C. Staudemeyer and E. R. Morris, "Understanding lstm–a tutorial into long short-term memory recurrent neural networks," arXiv preprint arXiv:1909.09586 (2019).
[26]A. Sherstinsky, "Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network," Physica D: Nonlinear Phenomena **404**, 132306 (2020).
[27]E. N. Lorenz, EN"Deterministic Nonperiodic Flow," Journal of the Atmospheric Sciences **20**, 130–141 (1963), publisher: American Meteorological Society Section: Journal of the Atmospheric Sciences.
[28]Z.-M. Chen and W. Price, "On the relation between rayleigh–bénard convection and lorenz system," Chaos, Solitons and Fractals **28**, 571–578 (2006).
[29]N. A. F. Senan, "A brief introduction to using ode45 in matlab," University of California at Berkeley, USA (2007).
[30]F. Chollet *et al.*, "Keras," `https://keras.io` (2015).
[31]A. Martín *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," (2015), software available from tensorflow.org.
[32]S. Kullback and R. A. Leibler, "On information and sufficiency," The annals of mathematical statistics **22**, 79–86 (1951).
[33]H. Jeffreys, "An invariant form for the prior probability in estimation problems," Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences **186**, 453–461 (1946).
[34]S.-H. Cha, "Comprehensive survey on distance/similarity measures between probability density functions," City **1**, 1 (2007).
[35]X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research, Vol. 9, edited by Y. W. Teh and M. Titterington (PMLR, Chia Laguna Resort, Sardinia, Italy, 2010) pp. 249–256.
[36]A.-S. Roman, "Evaluating the privacy and utility of time-series data perturbation algorithms," Mathematics **11** (2023).
[37]A.-S. Roman, B. Genge, A.-V. Duka, and P. Haller, "Privacy-preserving tampering detection in automotive systems," Electronics **10** (2021).
[38]O. E. Rössler, "An equation for continuous chaos," Physics Letters A **57**, 397–398 (1976).
[39]K. Cetin, O. Afsar, and U. Tirnakli, "Generalized pesin-like identity and scaling relations at the chaos threshold of the rössler system," Entropy **20** (2018), 10.3390/e20040216.
[40]K. M. Ibrahim, R. K. Jamal, and F. H. Ali, "Chaotic behaviour of the rossler model and its analysis by using bifurcations of limit cycles and chaotic attractors," Journal of Physics: Conference Series **1003**, 012099 (2018).
[41]J. Sunny, J. Schmitz, and L. Zhang, "Artificial neural network modelling of rossler's and chua's chaotic systems," in *2018 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)* (2018) pp. 1–4.
[42]R. Barrio, F. Blesa, and S. Serrano, "Unbounded dynamics in dissipative flows: Rössler model," Chaos: An Interdisciplinary Journal of Nonlinear Science **24**, 024407 (2014), https://pubs.aip.org/aip/cha/article-pdf/doi/10.1063/1.4871712/14607145/024407_1_online.pdf.
[43]U. Javed, K. Ijaz, M. Jawad, I. Khosa, E. Ahmad Ansari, K. Shabih Zaidi, M. Nadeem Rafiq, and N. Shabbir, "A novel short receptive field based dilated causal convolutional network integrated with bidirectional

lstm for short-term load forecasting," Expert Systems with Applications **205**, 117689 (2022).

[44]R. Bolboacă and B. Genge, "Unsupervised outlier detection in continuous nonlinear systems: Hybrid approaches with autoencoders and one-class svms," in *The 17th International Conference Interdisciplinarity in Engineering*, edited by L. Moldovan and A. Gligor (Springer Nature Switzerland, Cham, 2024) pp. 376–398.

[45]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).

[46]M. Pirani, P. Thakkar, P. Jivrani, M. H. Bohara, and D. Garg, "A comparative analysis of arima, gru, lstm and bilstm on financial time series forecasting," in *2022 IEEE International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE)* (IEEE, 2022) pp. 1–6.

[47]S. Yang, X. Yu, and Y. Zhou, "Lstm and gru neural network performance comparison study: Taking yelp review dataset as an example," in *2020 International workshop on electronic communication and artificial intelligence (IWECAI)* (IEEE, 2020) pp. 98–101.

[48]N. Gruber and A. Jockisch, "Are gru cells more specific and lstm cells more sensitive in motive classification of text?" Frontiers in artificial intelligence **3**, 40 (2020).

[49]A. Shewalkar, D. Nyavanandi, and S. A. Ludwig, "Performance evaluation of deep neural networks applied to speech recognition: Rnn, lstm and gru," Journal of Artificial Intelligence and Soft Computing Research **9**, 235–245 (2019).

[50]J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," arXiv preprint arXiv:1412.3555 (2014).

[51]A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," Advances in neural information processing systems **30** (2017).

[52]S. Karita, N. Chen, T. Hayashi, T. Hori, H. Inaguma, Z. Jiang, M. Someki, N. E. Y. Soplin, R. Yamamoto, X. Wang, S. Watanabe, T. Yoshimura, and W. Zhang, "A comparative study on transformer vs rnn in speech applications," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, Vol. 1 (2019) pp. 449–456.

[53]L. Tunstall, L. Von Werra, and T. Wolf, *Natural language processing with transformers* (O'Reilly Media, Inc., Sebastopol, California, 2022).

[54]A. Murtadha, S. Pan, W. Bo, J. Su, X. Cao, W. Zhang, and Y. Liu, "Rank-aware negative training for semi-supervised text classification," Transactions of the Association for Computational Linguistics **11**, 771–786 (2023).

[55]P.-A. Buestán-Andrade, M. Santos, J.-E. Sierra-García, and J.-P. Pazmiño-Piedra, "Comparison of lstm, gru and transformer neural network architecture for prediction of wind turbine variables," in *International Conference on Soft Computing Models in Industrial and Environmental Applications* (Springer, 2023) pp. 334–343.

[56]Ezen-Can, Aysu, "A comparison of lstm and bert for small corpus," arXiv preprint arXiv:2009.05451 (2020), https://doi.org/10.48550/arXiv.2009.05451.

[57]A. Zeng, M. Chen, L. Zhang, and Q. Xu, "Are transformers effective for time series forecasting?" in *Proceedings of the AAAI conference on artificial intelligence*, Vol. 37 (2023) pp. 11121–11128.

[58]I. Lomov, M. Lyubimov, I. Makarov, and L. E. Zhukov, "Fault detection in tennessee eastman process with temporal deep learning models," Journal of Industrial Information Integration **23**, 100216 (2021).