IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received March 22, 2022, accepted April 13, 2022, date of publication April 25, 2022, date of current version May 4, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3170042

Activation Function Modulation in Generative Triangular Recurrent Neural Networks

SESHADRI SIVAKUMAR[1], (Senior Member, IEEE), AND SHYAMALA SIVAKUMAR[2], (Senior Member, IEEE)
[1]Pasumai Energytech LLC, Bradenton, FL 34211, USA
[2]Computing Information Systems, Saint Mary's University, Halifax, NS B3H 3C3, Canada
Corresponding author: Seshadri Sivakumar (seshadri.sivakumar@pasumaienergytech.com)

ABSTRACT Autonomous generation of time series is challenging because the network must capture short-term features while tracking long-term time dependencies. This paper introduces the modulation of the activation function slopes of the upper-lower triangular recurrent neural networks (ULTRNNs) for dynamic variation of memory through a secondary recurrent network with its own independent states. A zigzag propagation algorithm for weight updates is proposed that accounts for the dynamic interaction of the states between the ULTRNN and the secondary network. A novel training method is proposed that distributes the eigenvalues of the closed-loop system around the unit circle in the complex z-plane to ensure that the network behaves as a nonlinear oscillator with an output that neither collapses nor saturates but continues to emulate the target. Examples encompassing the Lorenz series, Santa Fe laser data, *kolam* patterns, electrocardiogram (ECG) signals, stock pricing data, and smart grid data are presented to demonstrate that the proposed approach is highly effective in the generative modeling of complex periodic, chaotic, and nonstationary time series. The qualitative and quantitative performance of the ULTRNN obtained with the proposed activation-function modulation technique is comparable to that of state-of-the-art techniques including feedforward networks and generative adversarial networks, but with far fewer trainable parameters and shorter computation times.

INDEX TERMS Triangular RNN, activation function slope modulation, generative networks, closed-loop training.

I. INTRODUCTION

This paper introduces a novel sparse recurrent neural network (RNN) architecture for the generative modeling of time series. Generative modeling is challenging because the network must capture short-term features while tracking long-term time-dependencies. RNN-based networks are uniquely suited to such problems owing to their autoregression ability. Typically, the activation function slope used in such networks is fixed and determines their ability to model long-term time-dependencies at the cost of short-term features or vice versa. We propose dynamically varying the activation function slope to control the memory retention and forgetting. In addition, unconstrained RNN weight placements may result in unstable networks whose outputs saturate or highly stable networks whose outputs collapse. Therefore, we propose a novel training method to constrain the eigenstructure of the closed-loop system for marginal stability to ensure a stable emulation of the target data with an output that neither collapses nor saturates. We apply activation function modulation, together with constraining the closed-loop eigenvalue distribution, to synthetically generate chaotic, complex periodic and quasi-periodic waveforms, and model non-stationary processes such as stock data.

Sparse RNNs have the advantage of reduced computation and storage compared with fully recurrent networks. In addition, special recurrent structures such as block-diagonal (BDRNN) [1] and upper-lower triangular (ULTRNN) [2] inherently facilitate monitoring stability and ensure a robust learning ability. The eigenvalues of the 2×2 diagonal blocks of the BDRNN recurrent weight matrix were constrained during the training to lie on the unit circle of the complex z-plane. Such placement inherently mitigates the vanishing and exploding gradient issues normally associated with conventional RNN [3]. Reported applications of BDRNNs include speech recognition [1], lung-sound processing [4],

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Liu.

and telecom call volume prediction [5]. The ULTRNN [2] uses upper and lower triangular weight matrices, each with 2×2 diagonal blocks with the same constraints as the BDRNN. As a second constraint, the corresponding diagonal blocks in the two triangular matrices are set equal to reduce the chances of overfitting as the network learns to model the hidden oscillatory modes of the target trajectory. The ULTRNN is effective for modeling dynamic time series that exhibit chaotic, nonsymmetric, and long-term dependency behaviors [2].

Adaptation of the activation function slope has previously been considered a tool for improving learning performance. Reference [6] used this technique with a limited scope to improve the intrinsic stability of generic recurrent networks. Reference [7] used trainable activation functions in a deep feedforward neural network for an MNIST handwritten digit recognition task. In a previous study, the learning capability of a ULTRNN was enhanced by modulating the slope of the activation function associated with each state variable [8]. Modulation of the activation function incorporates variable memory with an objective similar to that of an LSTM network that employs multiplicative gates [9].

The slope of the activation function directly affects the time duration contribution of the corresponding state variable. A sharp activation function slope lengthens the time contribution and helps model and control long-term dependencies. Conversely, a slack slope shortens the time contribution and helps to model controlled ''forgetting''. A fixed activation slope favors the equal contribution of all states over time and the network learns long-term dependencies at the expense of its short-term performance or vice versa [8]. Varying the slope of the activation function with time can be used to model the dynamics of the system time constants that produce the target waveform. Reference [8] dynamically computed the slope of the (*tanh*) squashing function for each state variable using a secondary network. For this secondary network, a feedforward structure that uses the main ULTRNN states was chosen to simplify the training with a direct extension of the conventional backpropagation algorithm. However, such sharing of states may result in suboptimal mapping of the activation function slope space. To address this shortcoming, in this study, we propose using a recurrent architecture for a secondary network with its own independent and dimensionally unconstrained state variables. We call this secondary network an independent state activation function network (ISAFN).

The interaction between the state variables of the ULTRNN and ISAFN imposes additional constraints on the gradient computation process during training. Considering this interaction, we developed a novel zigzag propagation (z-prop) algorithm that uses a combination of forward and backward steps for exact gradient computation. A truncation technique that increases the computational speed by trading off the desired gradient accuracy level is presented. In addition, the activation function slope is constrained to a range that ensures the marginal stability of the ULTRNN.

The activation function modulation of the ULTRNN can be applied to a wide range of time-series modeling and prediction tasks. This study focuses on the unique objective of autonomous generation or replication of time series with behavioral characteristics close to those of the parent system as envisioned in [10], [16]. From a practical perspective, such generative networks can be used to clone new data when collecting real data is difficult or when its availability is sparse. We consider one-step prediction as the primary mechanism for training, such that the trained network autonomously generates outputs by feeding back only its own output from the previous time step. Because the stable performance of the trained network depends on the eigenstructure of the closed-loop system, we introduce an eigenvalue-based heuristic cost function. Using this approach, the closed-loop network behaves as a nonlinear oscillator emulating the target with an output that neither collapses nor saturates. We demonstrate the performance of ISAFN-ULTRNN with the autonomous generation of synthetic examples of Lorenz limit cycles [11], Santa Fe laser data [12], one-stroke *kolam* patterns [13], ECG signals [14], [15], Google stock data [16], and smart grid data [17], [18].

The main contributions of this work include: (a) the controlled modulation of the activation function slopes of the ULTRNN which facilitates the dynamic variation of memory retention. A secondary recurrent network whose states are independent of the ULTRNN states is employed for the dynamic computation of the activation function slopes. (b) A zigzag propagation algorithm that uses a combination of forward and backward steps for weight updates to account for the dynamic interaction of the states between the ULTRNN and secondary network. (c) A novel training method distributes the eigenvalues of the linearized closed-loop system around the unit circle in the complex z-plane to ensure that the trained network behaves as a nonlinear oscillator with an output that neither collapses nor saturates but continues to emulate the target.

The remainder of this paper is organized as follows: In Section II, the structure of the ULTRNN with ISAFN is developed, and its key features are compared with the common-state AFN (CSAFN) presented in [8]. In Section III, we develop a learning algorithm for the ISAFN with a specific focus on the z-prop algorithm for gradient computation, which considers the interaction between the state variables of the ULTRNN and ISAFN. In this section, we also discuss the computational burden of the z-prop algorithm and suggest techniques for increasing computational speed through truncation. Section IV describes the development of an eigenvalue-based cost function that ensures the marginal stability of the linearized closed-loop system. Section V presents illustrative examples to demonstrate and contrast the generative performance of the ISAFN-ULTRNN with state-of-the-art networks. Section VI summarizes the key contributions of this study.

II. DYNAMIC ACTIVATION FUNCTION MODULATION

A. ACTIVATION FUNCTION MODULATED ULTRNN

The ULTRNN architecture uses twin triangular state-feedback weight matrices [2] as shown in Figure 1. The system equations, with sampling instant k, are given by:

$$\left.\begin{aligned} \boldsymbol{s}_{\boldsymbol{x}}^{\varsigma}(k+1) &= \boldsymbol{W}_{x}^{\varsigma}\boldsymbol{x}^{\zeta}(k) + \boldsymbol{B}_{x}^{\varsigma}\boldsymbol{u}(k) \\ \boldsymbol{x}^{\varsigma}(k) &= \boldsymbol{f}_{\boldsymbol{x}}(\boldsymbol{a}_{x}^{\varsigma}(k), \boldsymbol{s}_{\boldsymbol{x}}^{\varsigma}(k)), \varsigma = \boldsymbol{U}, \boldsymbol{L} \end{aligned}\right\} \quad (1)$$

$$\left.\begin{aligned} \boldsymbol{s}_{\boldsymbol{y}}(k) &= \boldsymbol{C}_{\boldsymbol{x}}^{\boldsymbol{U}}\boldsymbol{x}^{\boldsymbol{U}}(k) + \boldsymbol{C}_{x}^{\boldsymbol{L}}\boldsymbol{x}^{\boldsymbol{L}}(k) \\ \boldsymbol{y}(k) &= \boldsymbol{h}_{\boldsymbol{y}}(\boldsymbol{s}_{\boldsymbol{y}}(k)), \varsigma = \boldsymbol{U}, \boldsymbol{L} \end{aligned}\right\} \quad (2)$$

where, with $\varsigma = U, L$ representing the upper and lower networks, respectively,

$\boldsymbol{W}_{x}^{\varsigma} = \{w_{x_{i,j}}^{\varsigma}, i, j = 1, 2, ..N_x\}$ are the triangular matrices,

$\boldsymbol{B}_{x}^{\varsigma} = \{b_{x_{i,j}}^{\varsigma}, i = 1, 2, \ldots N_x, j = 1, 2, \ldots N_i\}$ are the input matrices,

$\boldsymbol{C}_{\boldsymbol{x}}^{\varsigma} = \{c_{x_{j,i}}^{\varsigma}, i = 1, 2, \ldots N_x, j = 1, 2, \ldots N_o\}$ are the output matrices.

With $i = 1, 2, \ldots N_x$,

$\boldsymbol{x}^{\varsigma}(k) = \{x_i^{\varsigma}(k)\}^T$ as the state vector,

$\boldsymbol{s}_{\boldsymbol{x}}^{\varsigma}(k) = \{s_{x_i}^{\varsigma}(k)\}^T$ is the intermediate linear state vector,

$\boldsymbol{a}_{x}^{\varsigma}(k) = \{a_{x_i}^{\varsigma}(k)\}^T$ is the activation function slope vector, and

$\boldsymbol{f}_{\boldsymbol{x}}(\boldsymbol{a}, \mathbf{s}) = \{f_x(a_i, s_i)\}^T$ is the state activation function vector.

$\boldsymbol{u}(k) = \{u_i(k), i = 1, 2, \ldots N_i\}^T$ is the network input vector.

With $i = 1, 2, \ldots N_o$,

$\boldsymbol{s}_{\boldsymbol{y}}(k) = \{s_{y_i}(k)\}^T$ is the intermediate linear output vector,

$\boldsymbol{y}(k) = \{y_i(k)\}^T$ is the network output vector, and

$\boldsymbol{h}_{\boldsymbol{y}}(\boldsymbol{s}) = \{h_y(s_i)\}^T$ is the output activation function vector.

The state activation function is:

$$f_x(a_i, s_i) = \frac{(1 - e^{-a_i s_i})}{(1 + e^{-a_i s_i})}, \quad a_i \geq 0 \quad (3)$$

Note a_i varies dynamically. With a fixed slope $\alpha > 0$, the output activation function is:

$$h_y(s_i) = f_x(\alpha, s_i) \quad (4)$$

Note that the i^{th} 2×2 block diagonal submatrices of $\boldsymbol{W}_{x}^{\varsigma}$, $\varsigma = \boldsymbol{U}, \boldsymbol{L}$ are constrained to be equal and formulated using angular θ_i variables as follows:

$$\begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i) \end{bmatrix}, \quad i = 1, 2, \ldots N_x/2 \quad (5)$$

A key motivation for using (5) is that it is an effective mechanism for modeling the underlying oscillatory modes [1], [2]. A sufficient condition for network and training stability is [1],

$$a_{x_i}^{\varsigma}(k) \leq 2, \quad i = 1, 2, \ldots N_x. \quad (6)$$

Eqn. (5) with (6) constrains each eigenvalue pair of $\boldsymbol{W}_{x}^{\varsigma}$ to lie on the unit circle of the complex z-plane.

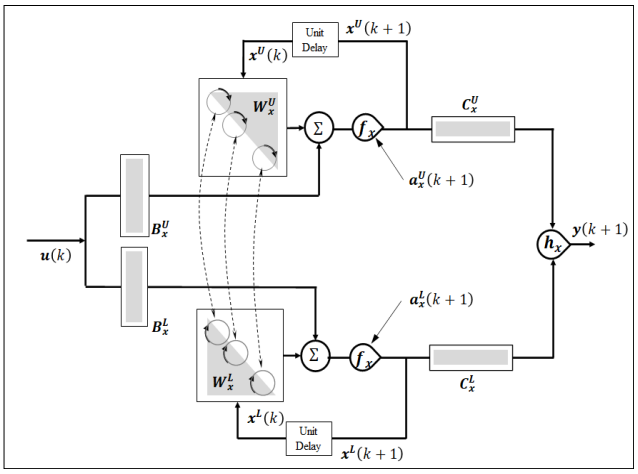

FIGURE 1. ULTNN architecture with activation function slope modulation.

This ensures both network and learning stability without any need for online monitoring. Such eigenvalue placement eliminates the exploding and vanishing gradients issue associated with RNN. Constraining the respective 2×2 block diagonals of $\boldsymbol{W}_{x}^{\boldsymbol{U}}$ and $\boldsymbol{W}_{x}^{\boldsymbol{L}}$ to be equal minimizes the possibility of overfitting while learning [2], [19].

Typically, in neural networks, the slope a of the activation function $f_x(a, s)$ is chosen *a priori* and is always fixed. In this study, we dynamically vary the activation function slopes to emphasize the contribution of some states over others and this varies with time. With each state variable of the main ULTRNN having a unique activation function, the variation in the activation function slopes over time enhances the dynamic performance of the ULTRNN.

We now describe two alternative architectures for a secondary network that output the variable slope activation functions required by the main network. The first (CSAFN), is a feedforward network that uses the state variables of the main network to compute the activation function slopes [8]. The second (ISAFN), is a novel architecture that employs a recurrent network with its own independent state variables to compute activation function slopes.

B. COMMON STATES ACTIVATION FUNCTION NETWORK (CSAFN)

The state variables are common to the ULTRNN and CSAFN as shown in Figure 2. The output of the CSAFN is the activation function slope $\boldsymbol{a}_{\boldsymbol{x}}^{\varsigma}(k)$, given by:

$$\left.\begin{aligned} \boldsymbol{s}_{\tilde{\boldsymbol{x}}}^{\varsigma}(k+1) &= \boldsymbol{W}_{\tilde{\boldsymbol{x}}}^{\varsigma}\boldsymbol{x}^{\varsigma}(k) + \boldsymbol{B}_{\tilde{\boldsymbol{x}}}^{\varsigma}\boldsymbol{u}(k) \\ \boldsymbol{a}_{\boldsymbol{x}}^{\varsigma}(k) &= \boldsymbol{g}_{\tilde{\boldsymbol{x}}}(\boldsymbol{s}_{\tilde{\boldsymbol{x}}}^{\varsigma}(k)), \varsigma = \boldsymbol{U}, \boldsymbol{L} \end{aligned}\right\} \quad (7)$$

where,

$\boldsymbol{W}_{\tilde{\boldsymbol{x}}}^{\varsigma} = \{w_{\tilde{x}_{i,j}}^{\varsigma}, i, j = 1, 2, \ldots N_x\}$ are the state weights,

$\boldsymbol{B}_{\tilde{\boldsymbol{x}}}^{\varsigma} = \{b_{\tilde{x}_{i,j}}^{\varsigma}, i = 1, 2, \ldots N_x, j = 1 \ldots N_i\}$ are the input matrices,

$\boldsymbol{s}_{\tilde{\boldsymbol{x}}}^{\varsigma}(k) = \{s_{\tilde{x}_i}^{\varsigma}(k), i = 1, 2, \ldots N_x\}^T$ are the intermediate linear state vectors, and

$\boldsymbol{g}_{\tilde{\boldsymbol{x}}}(\boldsymbol{s})$ represents the vector function $\{g_{\tilde{x}}(s_i)$, $i = 1, 2, \ldots N_x\}^T$ on vector $\boldsymbol{s} = \{s_i, i = 1, 2, \ldots N_x\}^T$,

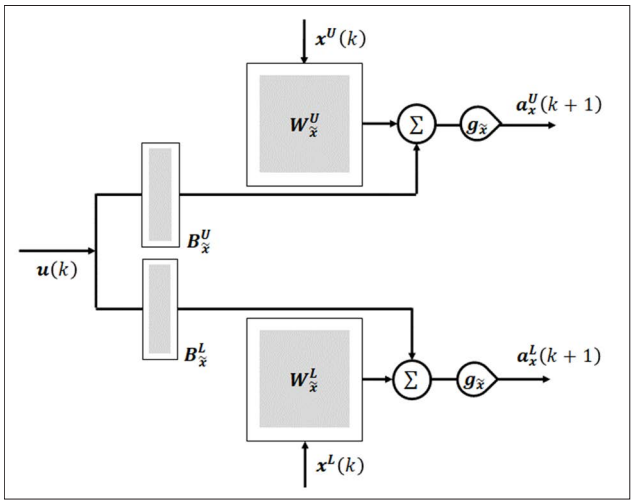

FIGURE 2. Common-state AFN architecture.

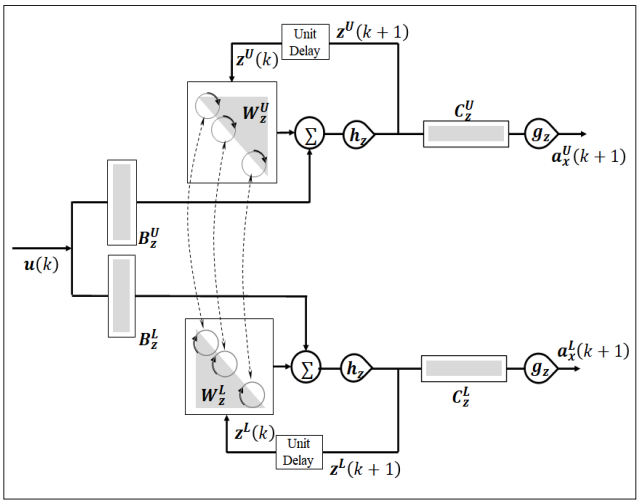

FIGURE 3. Independent-state AFN architecture.

where

$$g_{\tilde{x}}(s_i) = m_0 + \frac{2m_2}{\left(1 + e^{-m_3(s_i+m_1)}\right)} \tag{8}$$

where,

the parameters $m_i, i = 0, 1, 2, 3$ are the scaling factors that determine the vertical shift, span, slope, and horizontal shift, respectively.

During training, the CSAFN simultaneously learns to output $\boldsymbol{a}_x^{\varsigma}(k)$ as the ULTRNN learns to model dynamic processes. Note that in [8], $\boldsymbol{W}_{\tilde{x}}^{\varsigma}$ is chosen as a feedforward, triangular matrix with block diagonals of the form (5). However, such a constraint is not necessary because the CSAFN does not use recurrence.

C. INDEPENDENT STATES ACTIVATION FUNCTION NETWORK (ISAFN)

The CSAFN architecture may impact the learning performance by mapping the network state to a positive activation function slope. Such suboptimal mapping compromises network memory. This may impact the generalizability of the CSAFN, and thereby, the network's robust learning ability. With the primary motivation of decoupling the network states of the AFN from the main network and improving the memory retention control, we propose a novel AFN architecture that employs recurrent states that are completely independent of the main ULTRNN states as shown in Figure 3. The outputs of the independent states AFN (ISAFN) are the activation function slopes $\boldsymbol{a}_{\boldsymbol{x}}^{\varsigma}(k)$ used by the main ULTRNN, as shown in Figure 1. The state equations of the ISAFN are given by:

$$\left.\begin{aligned} \boldsymbol{s}_z^{\varsigma}(k+1) &= \boldsymbol{W}_z^{\varsigma}\boldsymbol{z}^{\varsigma}(k) + \boldsymbol{B}_z^{\varsigma}\boldsymbol{u}(k) \\ \boldsymbol{z}^{\varsigma}(k) &= \boldsymbol{h}_z\left(\boldsymbol{s}_z^{\varsigma}(k)\right), \varsigma = \boldsymbol{U}, \boldsymbol{L} \end{aligned}\right\} \tag{9}$$

where,

$\boldsymbol{z}^{\varsigma}(k) = \{z_i^{\varsigma}(k), i = 1, 2, \ldots N_z\}^T$ is the state vector,

$\boldsymbol{W}_z^{\varsigma} = \{w_{z_{i,j}}^{\varsigma}, i, j = 1, 2, \ldots N_z\}$, is the state weight matrix,

$\boldsymbol{B}_z^{\varsigma} = \{b_{z_{i,j}}^{\varsigma}, i = 1, 2, \ldots N_z, j = 1, 2, \ldots N_i\}$ is the input matrix,

$\boldsymbol{s}_z^{\varsigma}(k) = \{s_{z_i}^{\varsigma}(k), i = 1, 2, \ldots N_z\}^T$, is the intermediate linear state vector, and

$\boldsymbol{h}_z(\boldsymbol{s}) = \{h_z(s_i)\}^T, \{s_i, i = 1, 2, \ldots N_z\}^T$ is the activation function of the form (4).

Owing to the recurrence used in the ISAFN, the state matrices $\boldsymbol{W}_z^{\varsigma}$ are constrained to triangular matrices with block diagonal submatrices of form (5). The output of the ISAFN is:

$$\left.\begin{aligned} \boldsymbol{s}_{\boldsymbol{a}}^{\varsigma}(k) &= \boldsymbol{C}_z^{\varsigma}\boldsymbol{z}^{\varsigma}(k) \\ \boldsymbol{a}_{\boldsymbol{x}}^{\varsigma}(k) &= \boldsymbol{g}_z\left(\boldsymbol{s}_{\boldsymbol{a}}^{\varsigma}(k)\right), \varsigma = \boldsymbol{U}, \boldsymbol{L} \end{aligned}\right\} \tag{10}$$

where,

$\boldsymbol{C}_z^{\varsigma} = \{c_{z_{j,i}}^{\varsigma}, i = 1, 2, \ldots N_z, j = 1, 2, \ldots N_x\}$ is the output matrix, and $\boldsymbol{g}_z(\boldsymbol{s}) = \{g_z(s_i), i = 1, 2, \ldots N_x\}^T$ is the activation function of the same form as (8).

Note N_z does not need to be equal to N_x and can be selected as higher or lower depending on the task objectives.

The parameters of $g_{\tilde{x}}(\cdot)$ of the CSAFN and $g_z(\cdot)$ of the ISAFN, and how they impact the activation function $f_x(a, x)$ of the ULTRNN, are shown in Figure 4. The vertical-shift factor m_0 determines the minimum limit of the activation function slope; the span factor m_1 expands or contracts the span and determines the granularity of forgetting levels; and the slope factor m_2 impacts both the range and sensitivity of forgetting levels; the horizontal-shift factor m_3 impacts the desired value of the activation function slope for no external inputs. These parameters are empirically chosen such that (a) the marginal stability of the ULTRNN and its training are retained at all instants with or without external inputs, and (b) the slope assumes a range of values above a minimum threshold to represent various levels of forgetting.

III. TRAINING ALGORITHM

The combined CSAFN and ULTRNN are referred to as CULTN, and the combined ISAFN and ULTRNN are referred to as IULTN. The learning algorithms for the CULTN and IULTN are described in this section. The algorithms were

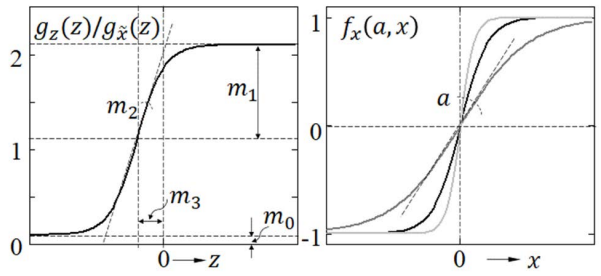

FIGURE 4. Activation functions of the AFN (left) and the main ULTRNN (right).

derived from the small perturbation theory to compute the exact gradients. The cost function for minimizing the output error during training is:

$$J_y = \sum_{k=1}^{N_q} \sum_{i=1}^{N_o} e_{y_i}(k)^2 \tag{11}$$

where $e_{y_i}(k)$ is the element of the error vector given by

$$\boldsymbol{e_y}(k) = \left\{e_{y_i}(k)\right\}^T = \boldsymbol{y^p}(k) - \boldsymbol{y}(k) \tag{12}$$

where $\boldsymbol{y^p}(k) = \left\{y_i^p(k), i = 1, 2, ..N_o\right\}^T$ is the target vector and, N_q is the number of training samples in an epoch.

A. TRAINING CULTN

The feedforward architecture of the CSAFN allows training to be performed with a simple extension of the conventional backpropagation algorithm. The following steps are performed for $k = N_q, ..1$, at each training cycle $t = 1, ..N_t$, with $\varsigma = \boldsymbol{U}, \boldsymbol{L}$

$$\boldsymbol{\varepsilon_{xo}^{\varsigma}} = \begin{cases} \boldsymbol{e_y}(k)^T \boldsymbol{h_y'}(k)\, \boldsymbol{C_x^{\varsigma}}, & k = N_q \\ [\boldsymbol{f_x^{s'^{\varsigma}}}(k)\, \boldsymbol{W_x^{\varsigma}} + \boldsymbol{f_x^{a'^{\varsigma}}}(k)\, \boldsymbol{g_{\tilde{x}}^{\prime\varsigma}}(k)\, \boldsymbol{W_{\tilde{x}}^{\varsigma}}]\boldsymbol{e_x^{\varsigma}}(k) \\ \quad + \boldsymbol{e_y}(k)^T \boldsymbol{h_y'}(k)\, \boldsymbol{C_x^{\varsigma}}, \\ \qquad\qquad k = N_q - 1, ..2, 1 \end{cases} \tag{13}$$

where,

$\boldsymbol{\varepsilon_{xo}^{\varsigma}}$ is an intermediate error function variable,

$\boldsymbol{f_x^{a'^{\varsigma}}}(k) = diag\{f_{x_i}^{a'^{\varsigma}}(k), i = 1, 2, .., N_x\}$, and

$\boldsymbol{f_x^{s'^{\varsigma}}}(k) = diag\{f_{x_i}^{s'^{\varsigma}}(k), i = 1, 2, .., N_x\}$

where,

$f_{x_i}^{a'^{\varsigma}}(k)$ and $f_{x_i}^{s'^{\varsigma}}(k)$ are the partial derivatives of $f_x(a_{x_i}^{\varsigma}(k), s_{x_i}^{\varsigma}(k))$ with respect to $a_{x_i}^{\varsigma}(\cdot)$ and $s_{x_i}^{\varsigma}(\cdot)$, respectively;

$\boldsymbol{g_{\tilde{x}}^{\prime\varsigma}}(k) = diag\{g_{\tilde{x}_i}^{\prime\varsigma}(k), i = 1, 2, .., N_x\}$,

where,

$g_{\tilde{x}_i}^{\prime\varsigma}(k)$ is the derivative of $g_{\tilde{x}}\left(s_{x_i}^{\varsigma}(k)\right)$, and

$\boldsymbol{h_y'}(k) = diag\{h_{y_i}'(k), i = 1, 2, .., N_o\}$,

where,

$h_{y_i}'(k)$ is the derivative of $h_y(s_i(k))$; and

$\boldsymbol{e_x^{\varsigma}}(k)$ is updated per (16).

Weight differentials for the ULTRNN are accumulated at each k iteration step as

$$\left.\begin{aligned} \boldsymbol{\Delta W_x^{\varsigma}} - &= \boldsymbol{\varepsilon_{xo}^{\varsigma}} \boldsymbol{f_x^{s'^{\varsigma}}}(k)\, \boldsymbol{x^{\varsigma}}(k-1) \\ \boldsymbol{\Delta B_x^{\varsigma}} - &= \boldsymbol{\varepsilon_{xo}^{\varsigma}} \boldsymbol{f_x^{s'^{\varsigma}}}(k)\, \boldsymbol{u^{\varsigma}}(k-1) \\ \boldsymbol{\Delta C_x^{\varsigma}} - &= \boldsymbol{e_y}(k)^T \boldsymbol{h_y'}(k)\, \boldsymbol{x^{\varsigma}}(k) \end{aligned}\right\} \tag{14}$$

Weight differentials for the CSAFN are accumulated at each k iteration step as

$$\left.\begin{aligned} \boldsymbol{\Delta W_{\tilde{x}}^{\varsigma}} - &= \boldsymbol{\varepsilon_{xo}^{\varsigma}} \boldsymbol{f_x^{a'^{\varsigma}}}(k)\, \boldsymbol{g_{\tilde{x}}^{\prime\varsigma}}(k)\, \boldsymbol{x^{\varsigma}}(k-1) \\ \boldsymbol{\Delta B_{\tilde{x}}^{\varsigma}} - &= \boldsymbol{\varepsilon_{xo}^{\varsigma}} \boldsymbol{f_x^{a'^{\varsigma}}}(k)\, \boldsymbol{g_{\tilde{x}}^{\prime\varsigma}}(k)\, \boldsymbol{u^{\varsigma}}(k-1) \end{aligned}\right\} \tag{15}$$

Note that the symbol $-\ =$ in (14) and (15) represents the negative accumulation of the weight differentials at the kth iteration step to the accumulated values from the previous steps N_q through $k+1$. Before the next iteration step, the following update is performed

$$\boldsymbol{e_x^{\varsigma}}(k) = \boldsymbol{\varepsilon_{xo}^{\varsigma}}. \tag{16}$$

After the accumulation of the differentials over all sample steps, the differential of the angular variable of each diagonal block of $\boldsymbol{W_x^{\varsigma}}$ is computed as [2]

$$\begin{aligned} &\Delta\theta_{\frac{i}{2}} \\ &= \cos\theta_{\frac{i}{2}} \left\{\Delta w_x^U{}_{i,i+1} - \Delta w_x^U{}_{i+1,i} + \Delta w_x^L{}_{i,i+1} - \Delta w_x^L{}_{i+1,i}\right\} \\ &\quad - \sin\theta_{\frac{i}{2}} \left\{\Delta w_x^U{}_{i,i} + \Delta w_x^U{}_{i+1,i+1} + \Delta w_x^L{}_{i,i} + \Delta w_x^L{}_{i+1,i+1}\right\} \\ &\quad i = 2, 4, \ldots N_x. \end{aligned} \tag{17}$$

(17) constrains the block diagonal elements of $\boldsymbol{W_x^{\varsigma}}$ to lie on the unit circle in the complex z-plane and helps maintain network stability and improve robust learning ability. The possibility of exploding and vanishing gradients is minimized while retaining the sensitivity required to model the oscillatory modes of the target.

B. TRAINING IULTN

The interaction of the recurrent states of the ISAFN with the recurrent states of the ULTRNN necessitates two parallel training processes, one for the ULTRNN and the other for the ISAFN.

1) BACKPROPAGATION ALGORITHM FOR THE ULTRNN

The following steps are performed for $k = N_q, ..2, 1$ at each training cycle $t = 1, 2, ..N_t$ with $\varsigma = \boldsymbol{U}, \boldsymbol{L}$

$$\boldsymbol{\varepsilon_{xo}^{\varsigma}} = \begin{cases} \boldsymbol{e_y}(k)^T \boldsymbol{h_y'}(k)\, \boldsymbol{C_x^{\varsigma}}, & k = N_q \\ \boldsymbol{f_x^{s'^{\varsigma}}}(k)\, \boldsymbol{W_x^{\varsigma}} \boldsymbol{\varepsilon_x^{\varsigma}}(k) + \boldsymbol{e_y}(k)^T \boldsymbol{h_y'}(k)\, \boldsymbol{C_x^{\varsigma}}, \\ \qquad\qquad k = N_q - 1, ..2, 1 \end{cases} \tag{18}$$

where $\boldsymbol{\varepsilon_{xo}^{\varsigma}}$ is the intermediate error function variable. Weight differentials are accumulated at each k^{th} iteration step as shown in (14) using $\boldsymbol{\varepsilon_{xo}^{\varsigma}}$ of (18). Before the next k^{th} iteration step, the following update is performed:

$$\boldsymbol{\varepsilon_x^{\varsigma}}(k) = \boldsymbol{\varepsilon_{xo}^{\varsigma}} \tag{19}$$

2) ZIGZAG PROPAGATION ALGORITHM FOR ISAFN

$\boldsymbol{x}^{\varsigma}(k)$ being a function of two variables is impacted by the ISAFN weights through two propagation paths. One is directly through $\boldsymbol{a}_x^{\varsigma}(k)$ of the ISAFN and the other is indirectly through $\boldsymbol{s}^{\varsigma}(k)$ of the ULTRNN, which is in turn impacted by $\boldsymbol{a}_x^{\varsigma}(k-1)$ and $\boldsymbol{W}_{\boldsymbol{x}}^{\varsigma}$. Hence, the ISAFN gradient terms are computed through a combination of forward and backward propagation steps as summarized below.

The following steps are performed for $k = 1, 2, \ldots N_q$ at each training cycle $t = 1, 2, ..N_t$ with $\boldsymbol{\varsigma} = \boldsymbol{U}, \boldsymbol{L}$

$$\left.\begin{aligned} \boldsymbol{\varepsilon}_z^{\varsigma}(k) &= \boldsymbol{f}_{\boldsymbol{x}}^{a'^{\varsigma}}(k)\, \boldsymbol{g}_z^{\prime\varsigma}(k) \\ \boldsymbol{E}_z^{\varsigma}(k) &= \boldsymbol{e}_{\boldsymbol{y}}(k)^{\boldsymbol{T}} \boldsymbol{h}_{\boldsymbol{y}}^{'}(k)\, \boldsymbol{C}_{\boldsymbol{x}}^{\varsigma} \boldsymbol{\varepsilon}_z^{\varsigma}(k)\, \boldsymbol{C}_z^{\varsigma} \boldsymbol{h}_z^{\prime\varsigma}(k) \end{aligned}\right\} \tag{20}$$

where,

$\boldsymbol{\varepsilon}_z^{\varsigma}(k)$ and $\boldsymbol{E}_z^{\varsigma}(k)$are the intermediate error function variables,
$\boldsymbol{g}_z^{\prime\varsigma}(k) = diag\{g_{z_i}^{\prime\varsigma}(k), i = 1, 2, .., N_x\}$, where $g_{z_i}^{\prime\varsigma}(k)$ is the derivative of $g_z\left(s_{z_i}^{\varsigma}\right)$; and
$\boldsymbol{h}_z^{\prime\varsigma}(k) = diag\{h_{z_i}^{\prime\varsigma}(k), i = 1, 2, ..N_z\}$, where $h_{z_i}^{\prime\varsigma}(k)$ is the derivative of $h_z(s_{z_i}^{\varsigma}(k))$.

Note that the interaction of the ISAFN and ULTRNN states is reflected in $\boldsymbol{\varepsilon}_z^{\varsigma}(k)$.

For each k-th iteration step above, the following steps are performed for $\kappa = k, k-1, ..2, 1$:

$$\boldsymbol{\varepsilon}_{zo}^{\varsigma} = \begin{cases} \boldsymbol{\varepsilon}_z^{\varsigma}(k), & \kappa = k \\ \boldsymbol{f}_{\boldsymbol{x}}^{s'^{\varsigma}}(k)\, \boldsymbol{W}_{\boldsymbol{x}}^{\varsigma} \boldsymbol{\varepsilon}_{z^{\varsigma}}(\kappa), & \kappa = k-1, ..2, 1 \end{cases} \tag{21}$$

$$\boldsymbol{E}_{zo}^{\varsigma} = \begin{cases} \boldsymbol{E}_z^{\varsigma}(k), & \kappa = k \\ [\boldsymbol{E}_z^{\varsigma}(k)\, \boldsymbol{W}_z^{\varsigma} + \boldsymbol{e}_{\boldsymbol{y}}(k)^{\boldsymbol{T}} \boldsymbol{h}_{\boldsymbol{y}}^{'}(k)\, \boldsymbol{C}_{\boldsymbol{x}}^{\varsigma} \boldsymbol{\varepsilon}_{zo}^{\varsigma} \boldsymbol{C}_z^{\varsigma}] \boldsymbol{h}_z^{\prime\varsigma}(\kappa), \\ \qquad \kappa = k-1, ..2, 1 \end{cases} \tag{22}$$

Weight differentials are accumulated at each κ iteration step as

$$\left.\begin{aligned} \boldsymbol{\Delta W}_z^{\varsigma} -{=} \boldsymbol{E}_{zo}^{\varsigma} z^{\varsigma}(\kappa-1) \\ \boldsymbol{\Delta B}_z^{\varsigma} -{=} \boldsymbol{E}_{zo}^{\varsigma} \boldsymbol{u}^{\varsigma}(\kappa-1) \\ \boldsymbol{\Delta C}_z^{\varsigma} -{=} \boldsymbol{e}_{\boldsymbol{y}}(k)^{\boldsymbol{T}} \boldsymbol{h}_{\boldsymbol{y}}^{'}(k)\, \boldsymbol{C}_{\boldsymbol{x}}^{\varsigma} \boldsymbol{\varepsilon}_{zo}^{\varsigma} z^{\varsigma}(\kappa) \end{aligned}\right\} \tag{23}$$

Before the next κ^{th} step, respectively, the following updates are performed:

$$\left.\begin{aligned} \boldsymbol{\varepsilon}_z^{\varsigma}(\kappa) &= \boldsymbol{\varepsilon}_{zo}^{\varsigma} \\ \boldsymbol{E}_z^{\varsigma}(k) &= \boldsymbol{E}_{zo}^{\varsigma} \end{aligned}\right\} \tag{24}$$

Given the backward steps for each forward iteration step, the algorithm is termed as zigzag propagation (z-prop). The algorithm is illustrated in Figure 5 for iterations $k-1$ and k. In the figure, the solid downward arrow indicates a conventional backward propagation step, and the dashed upward arrow indicates forward propagation. Each box shows an error function from which the gradient components are obtained by pre-multiplying with $\boldsymbol{e}_{\boldsymbol{y}}(\cdot)^{\boldsymbol{T}}$ and post-multiplying with the ISAFN state $\boldsymbol{z}^{\varsigma}(\cdot)$. The process begins with the initial error computed for each kth iteration step (shown in the top-left box). The error function at each lower box is obtained

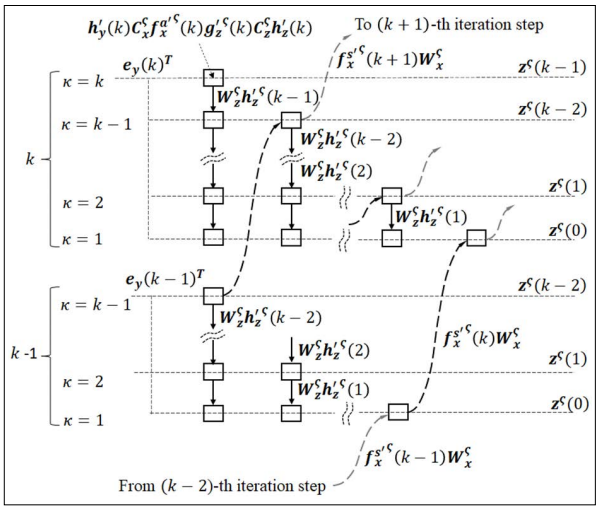

FIGURE 5. Zigzag propagation algorithm for ISAFN: representative iteration steps.

through post-multiplication of the term in the box above with the ISAFN backpropagation term $\boldsymbol{W}_z^{\varsigma}\boldsymbol{h}_z'^{\varsigma}(\cdot)$. The process continues in the next column with the error in the top box obtained by pre-multiplying the error in the previous step $(k-1)$ with the ULTRNN forward propagation term $\boldsymbol{f}_{\boldsymbol{x}}^{s'^{\varsigma}}(\cdot)\boldsymbol{W}_{\boldsymbol{x}^{\varsigma}}$. This process is repeated recursively for all columns in the kth iteration step, and for all $k = 1, 2, ..N_q$.

3) WEIGHT UPDATES

The CULTN or IULTN weights are updated at each training cycle consisting of an epoch of K_q time steps, using

$$\Phi(t+1) = \Phi(t) - \frac{\eta_w}{K_q} \Delta\Phi(t), \quad \mu > 0 \tag{25}$$

$\Phi(t)$ represents the weights in training cycle t, and η_w is the learning rate parameter. Note that updating $\boldsymbol{W}_{\boldsymbol{x}}^{\varsigma}$ and $\boldsymbol{W}_z^{\varsigma}$ is subject to the form of (17).

4) COMPUTATIONAL REQUIREMENTS

The number of trainable weights for the CULTN is given by:

$$N_w^C = N_x(3N_x + 4N_i + 2N_o) \tag{26}$$

and for the IULTN is given by

$$N_w^I = N_x\,(N_x + 2(N_i + N_o)) + N_z\,(N_z + 2(N_i + N_x)) \tag{27}$$

For a given network size, the z-prop used by the ISAFN requires $O(N_q^2/2)$ computations whereas the CSAFN requires only $O(N_q)$ computations. Hence, the ISAFN can be computationally expensive for a large N_q. Truncating the number of κ recursions for a large k to a fraction of N_q trades off the accuracy of the gradients with a reduction in computational speed. For example, κ iterations can be limited to a threshold κ_{th}, that can be selected empirically.

To empirically assess the impact of recursion truncation on the gradient accuracy and computation speed, a representative 2-output, 5/5-mode ($N_x = N_z = 10$) IULTN was considered.

The inputs and outputs were chosen as random values in the range of (-1:1). The gradients were computed first with the z-prop algorithm and then with the direct perturbation of the individual weight elements. Errors between the gradients were compiled for several truncation thresholds as a fraction of N_q in the range (0.1:1). The error comparison was repeated for several N_q values in the range (50:200). The normalized gradient accuracy based on the root-mean-squared (RMS) error versus the computational time profiles is summarized in Figure 6. The gradient error is close to zero with no truncation ($\kappa_{th} = N_q$), which confirms the correctness of the z-prop equations (20) – (24). The gradient errors exhibited a hyperbolic profile with a κ_{th} knee point between $0.2N_q$ and $0.35N_q$. The lower the epoch size N_q, the higher the increase in error with increased truncation. The computational time exponentially increased as the truncation decreased to zero, with a worse impact for the higher epoch size cases. A threshold chosen close to the knee point provides a reasonable trade-off between the gradient accuracy and computation time.

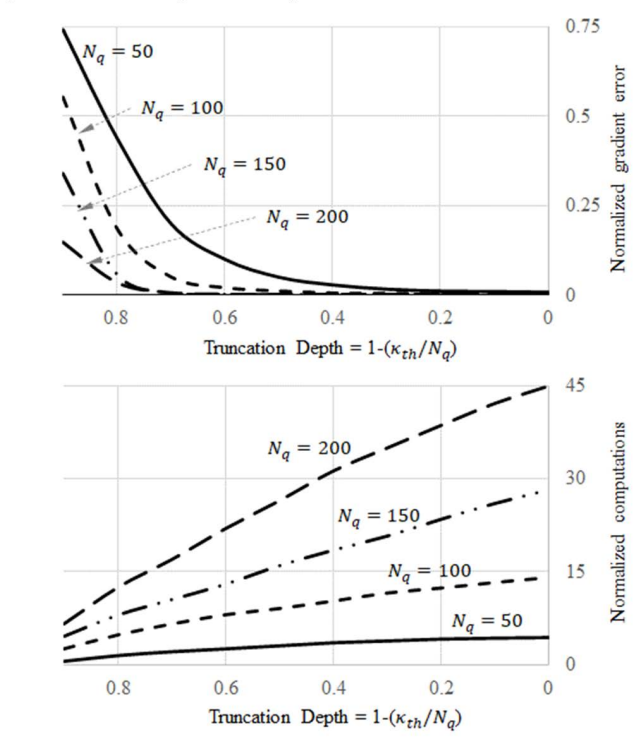

FIGURE 6. Impact of z-prop truncation depth on error and computation burden.

IV. CLOSED-LOOP GENERATIVE TRAINING

For autonomous generation using only the network's predicted outputs, the question of how to ensure closed-loop stability of the network needs to be considered. A tractable metric of the closed-loop system performance would be ideal to steer the training, such that the trained network behaves as a nonlinear oscillator with an output that neither collapses nor saturates but continues to emulate the target. However, owing to the nonlinearity of the activation functions and the variation in their slopes, such an ideal metric for guaranteed closed-loop performance is not computationally feasible. Instead, we seek a simple metric based on the linear control system theory that can potentially improve closed-loop performance. A heuristic metric based on the sufficiency condition for the local marginal stability of the linearized closed-loop system is derived. The closed-loop state transition equation of the ULTRNN with *one-step prediction feedback* for *small perturbations* is:

$$\begin{bmatrix} \Delta \boldsymbol{x}^U(k+1) \\ \Delta \boldsymbol{x}^L(k+1) \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}_{\boldsymbol{x}}^{s'^U}(k) & 0 \\ 0 & \boldsymbol{f}_{\boldsymbol{x}}^{s'^L}(k) \end{bmatrix} \times \boldsymbol{W}_{x_{CL}}^{UL}(k) \begin{bmatrix} \Delta \boldsymbol{x}^U(k) \\ \Delta \boldsymbol{x}^L(k) \end{bmatrix} \tag{28}$$

where,

$$\boldsymbol{W}_{x_{CL}}^{UL}(k) = \begin{bmatrix} \boldsymbol{W}_x^U + \boldsymbol{B}_x^U \boldsymbol{h}_{\boldsymbol{y}}'(k) \boldsymbol{C}_x^U & \boldsymbol{B}_x^U \boldsymbol{h}_{\boldsymbol{y}}'(k) \boldsymbol{C}_x^L \\ \boldsymbol{B}_x^L \boldsymbol{h}_{\boldsymbol{y}}'(k) \boldsymbol{C}_x^U & \boldsymbol{W}_x^L + \boldsymbol{B}_x^L \boldsymbol{h}_{\boldsymbol{y}}'(k) \boldsymbol{C}_x^L \end{bmatrix} \tag{29}$$

In (4), with slope $\alpha = 2$ for the output activation function $\boldsymbol{h}_{\boldsymbol{y}}(\cdot)$, derivative $\boldsymbol{h}_{\boldsymbol{y}}'(k)$ is a vector of positive values with a maximum of unity. In (29), we set $\boldsymbol{h}_{\boldsymbol{y}}'(k)$ to unity to obtain a linearized closed-loop matrix (LCLM)

$$\boldsymbol{W}_{x_{CL}}^{UL^*} = \begin{bmatrix} \boldsymbol{W}_x^U + \boldsymbol{B}_x^U \boldsymbol{C}_x^U & \boldsymbol{B}_x^U \boldsymbol{C}_x^L \\ \boldsymbol{B}_x^L \boldsymbol{C}_x^U & \boldsymbol{W}_x^L + \boldsymbol{B}_x^L \boldsymbol{C}_x^L \end{bmatrix} \tag{30}$$

and set $\boldsymbol{h}_{\boldsymbol{y}}'(k)$ to zero to obtain the open-loop representation

$$\boldsymbol{W}_{x_{OL}}^{UL^*} = \begin{bmatrix} \boldsymbol{W}_x^U & 0 \\ 0 & \boldsymbol{W}_x^L \end{bmatrix} \tag{31}$$

Ensuring the marginal stability of the closed-loop system for small perturbations requires constraining the eigenvalues of $\boldsymbol{W}_{x_{CL}}^{UL}(k)$ to within an annular ring around unity in the complex z-plane. Because the elements of $\boldsymbol{h}_{\boldsymbol{y}}'(k)$ vary from zero to unity, $\boldsymbol{W}_{x_{CL}}^{UL}(k)$ transitions within the ranges $\boldsymbol{W}_{x_{CL}}^{UL^*}$ and $\boldsymbol{W}_{x_{OL}}^{UL^*}$. Given that $\boldsymbol{W}_{x_{OL}}^{UL^*}$ is marginally stable due to constraint (5) imposed on its block diagonal elements, it is heuristically evident that imposing marginal stability on $\boldsymbol{W}_{x_{CL}}^{UL^*}$, in turn, ensures the marginal stability of $\boldsymbol{W}_{x_{CL}}^{UL}(k)$ for all values of $\boldsymbol{h}_{\boldsymbol{y}}'(k)$ **.** Note that in (28) as k varies, modulated by the AFN, $\boldsymbol{f}_{\boldsymbol{x}}^{s'^{\varsigma}}(k)$, $\varsigma = \boldsymbol{U}, \boldsymbol{L}$, shapes the closed-loop system performance by dynamically adjusting the annular ring that constrains the eigenvalues.

The cost function (11) used for training is modified as

$$J_{y\lambda} = J_y + J_\lambda \tag{32}$$

to include the cost term J_λ which represents the effective distance of the eigenvalues of $\boldsymbol{W}_{x_{CL}}^{UL^*}$ from unity.

$$J_\lambda = \frac{1}{2N_x} \sum_{i=1}^{2N_x} \left\| |\lambda_i \left(\boldsymbol{W}_{x_{CL}}^{UL^*} \right)| - 1 \right\| \tag{33}$$

where, $\lambda_i(\boldsymbol{W})$ is the eigenvalue of $\boldsymbol{W}$, and $\|\cdot\|$ represents any scalar norm. The gradients of J_λ with reference to the individual elements of $\boldsymbol{W}_{xCL}^{UL^*}$ are computed through direct perturbation at each weight update. Note that echo-state networks [20] employ eigenvalues in the training process to eliminate exploding gradients. The gradients of J_λ require $O(N_x^4)$ computations and can be computationally expensive for large networks. The computational speed can be increased at the cost of accuracy using an alternative cost function based on a rough estimate of the spectral radius, that is, the largest eigenvalue, of $\boldsymbol{W}_{xCL}^{UL^*}$ can be used [19]. Note that (29) applies only to equal input and output dimensions, and if other additional inputs or outputs are present, they should be separated from $\boldsymbol{B}_x^{\varsigma}, \boldsymbol{C}_x^{\varsigma}, \varsigma = \boldsymbol{U}, \boldsymbol{L}$.

After training, the marginal stability of the closed loop system can be assessed through an evaluation criterion based on several eigenvalue measures including J_λ. Example of other measures include the ones based only on the maximum and minimum amplitude of the eigenvalues. A successful training will result in the closed-loop eigenvalues distributed close to the unit circle in the complex z-plane. If the network with the chosen dimensions is inadequate to model the target time series, J_λ may not reduce sufficiently during learning. In such a scenario, the network dimensions need to be increased and retrained.

V. ILLUSTRATIVE EXAMPLES

Several illustrative examples are presented to demonstrate and compare the effectiveness of the IULTN architecture with state-of-the-art networks for generative tasks. The first two examples pertain to the autonomous generation of chaotic waveforms with characteristics close to those of Lorenz limit cycles [11], and Santa Fe laser data [12]. The third example deals with the autonomous generation of a complex periodic waveform based on the South Indian kolam patterns [13]. The fourth example autonomously generates synthetic electrocardiogram (ECG) data with characteristics similar to real patient data [14], [15]. The fifth example models a nonstationary process using the Google stock data studied in [16]. The final example generates hourly solar generation and energy consumption data studied in [18] as part of smart grid data synthesis. In each example, the network was trained for one-step prediction using the cost function (32) and tested in a closed loop with the output from the current time step used to predict the next output. Techniques adopted to improve the learning robustness include [8]: (a) input noise injection to improve generalizability [7], [21], (b) use of additional pilot inputs to embed periodicity, (c) use of a composite input consisting of the target and the network outputs for guided training, and (d) combined use of n-step prediction outputs for improved noise rejection during generation.

For the chaotic time series considered in the first two examples, the Kullback-Leibler (KL) divergence [22] was used for performance comparison. The KL divergence is not a measure of the global closeness of the network output to that of the target. However, at the local level, it is useful to quantitatively compare the probability distribution of outputs from different networks. We use the following formulation to facilitate a performance comparison of CULTN and IULTN: A covariance phase plot for each of the target and network outputs is created with coordinates $\{y_i^p(k), y_i^p(k+N_c)\}$ and $\{y_i(k), y_i(k+N_c)\}$, $i = 1, 2, ..N_o$, $k = 1, 2, \ldots N_d$, respectively, where N_d is the data length for an arbitrarily chosen covariance time step N_c. Each point on the plot is quantized into $\{y_m^*, y_n^*\}$, $m, n = 1, 2, \ldots N_r$ using a quantization level N_r. The probability distribution of the target outputs $P_i^p(y_m^*, y_n^*)$, and the network outputs $P_i(y_m^*, y_n^*)$, $i = 1, 2, ..N_o, m, n = 1, 2, \ldots N_r$, are computed as the joint probability of the occurrence of y_m^* and y_n^* over all time instances $k = 1, 2, ..N_d$. The symmetric KL divergence is computed as

$$D_i = 0.5 \left\{ \begin{array}{c} \sum_{m=1}^{N_r} \sum_{n=1}^{N_r} P_i(y_m^*, y_n^*) \ln\left(\dfrac{P_i^p(y_m^*, y_n^*)}{P_i(y_m^*, y_n^*)}\right) \\ + P_i^p(y_m^*, y_n^*) \ln\left(\dfrac{P_i(y_m^*, y_n^*)}{P_i^p(y_m^*, y_n^*)}\right) \end{array} \right\} \tag{34}$$

The parameters for the comparative analysis were the covariance time step N_c and quantization range N_r.

For all the examples, minimizing the cost function (32) was used to ensure closed-loop stability and robust learning. Typically, training is conducted in two phases to improve learning robustness. The first phase uses only the target waveform as the input, and the second combines the target waveform $y_i^p(k)$ and predicted network output $y_i(k)$:

$$\begin{aligned} u_i(k+1) &= \gamma y_i^p(k) + (1-\gamma) y_i(k), \\ & i = 1, 2, \ldots N_o, \quad 0 \le \gamma \le 1 \end{aligned} \tag{35}$$

where γ is chosen empirically. The training transition from the first to the second phase occurs when the eigenvalue metric J_λ decreases below a threshold. During the second training phase, the weight update parameter η_w was progressively reduced to ensure J_λ was maintained at or below the threshold.

To generate complex periodic waveforms, the waveform period was set to the epoch length N_q. To reduce the initial transient errors, the initial values of the network state variables in each training cycle τ are updated as

$$\chi_i(1)|_{\tau+1} = \left(\beta \chi_i(1) + (1-\beta)\chi_i(N_q)\right)|_\tau, \quad 0 \le \beta \le 1 \tag{36}$$

with β chosen empirically. Note, χ_i represents the network's state variables.

For all networks used in the examples, based on several trial-and-error runs and evaluating for consistent stability and learning performance, the parameters m_0, m_1, m_2, and m_3 of $g_{\tilde{x}}(\cdot)$ used in the CSAFN and $g_z(\cdot)$ used in the ISAFN per (8) are empirically chosen as 0.1, 2.1, 2.1, and 1.5, respectively. This selection limits the activation function slope to a maximum of 2.2 aimed at sustained long-term memory,

and a minimum of 0.1 heuristically chosen to regulate the forgetting level. The initial angular variables θ_i of the block diagonals of the ULTRNN and ISAFN were assigned random values that were uniformly distributed in the range $(0{:}2\pi)$. The elements of all other matrices were randomly distributed in the range $(-1{:}1)$. For input-output compatibility, before training, the inputs and targets were amplitude-normalized by scaling to span the range (-0.85:0.85). The initial value of the weight update parameter η_w was set to 0.1.

Tables 1 and 2 summarize the network parameters for the CULTN and IULTN, respectively, and Table 3 lists the key training and testing parameters used in the examples.

TABLE 1. CULTN network parameters.

Example	Mode	N_x	N_i	N_o	N_w^C
Lorenz Limit Cycles	6	12	3	3	648
	8	16	3	3	1056
Santa Fe Laser Data	4	8	1	2	256
	8	16	1	2	896
Kolam (Case 1)	6	12	2	2	576
ECG (Case 1)	4	8	2	2	288

TABLE 2. IULTN network parameters.

Example	Mode	N_x	N_z	N_i	N_o	N_w^I
Lorenz Limit Cycles	4/4	8	8	3	3	400
	6/6	12	12	3	3	792
Santa Fe Laser Data	4/4	8	8	1	2	320
Kolam (Case 1)	5/5	10	10	2	2	520
Kolam (Case 2)	6/6	12	12	2	2	720
ECG (Case 1)	2/2	4	4	2	2	112
	3/3	6	6	2	2	216
	3/4	6	8	2	2	276
ECG (Case 2)	12/12	24	24	3	2	2688
Stock Data	10/10	20	20	7	6	2400
Smart Grid data	15/15	30	30	23	22	7200

TABLE 3. Training and testing parameters.

Example	Training		Testing
	N_d	N_q	N_d
Lorenz Limit Cycles	10000	100	10000
Santa Fe Laser Data	1500	100	8500
Kolam (Case 1)	200	50	500
Kolam (Case 2)	9 x 200	50	9 x 500
ECG (Case 1)	200	50	500
ECG (Case 2)	50 x 200	50	50 x 500
Stock Data	3636	48	909
Smart Grid data	18 x 90	96	18 x 90

A. LORENZ LIMIT CYCLE

We consider the autonomous generation of time-series with characteristics close to the Lorenz limit cycles [11]. The target outputs are produced by solving the ordinary differential equation (ODE):

$$\frac{d}{dt}\begin{bmatrix} y_1^p \\ y_2^p \\ y_3^p \end{bmatrix} = \begin{bmatrix} \sigma_p\left(y_2^p - y_1^p\right) \\ \rho_p y_1^p - y_1^p y_3^p - y_2^p \\ y_1^p y_2^p - \beta_p y_3^p \end{bmatrix} \tag{37}$$

where $\sigma_p = 10$, $\rho_p = 28$, and $\beta_p = 8/3$ [11]. The ODE is solved with initial values set at $y_1^p = -10$, $y_2^p = -8$, $y_3^p = 30$, using a fourth-order Runge Kutta with a step size of 0.001 to generate the targets y_i^p, $i = 1, 2, 3$. The trajectories after amplitude normalization are shown in Figure 7.

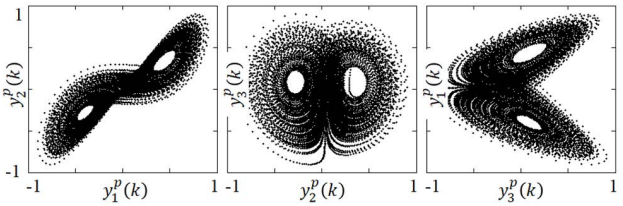

FIGURE 7. Lorenz limit cycles: target trajectories.

With 3-inputs and 3-outputs, a 6-mode ($N_x = 12$) CULTN, 4/4-mode ($N_x = N_z = 8$) IULTN, and 6/6-mode ($N_x = N_z = 12$) IULTN were considered. The networks were trained with 10000 points of the data generated by (37), with an epoch size $N_q = 100$. White noise with a variance of 0.016 was injected into the input. The network was first trained in an open loop and then in a closed loop with γ in (35) reduced from 1 to 0.5 to 0.25 such that the network was forced to rely more on its prediction to improve the orbit switching accuracy. The generative performance of the network was assessed by replacing the targets $y_i^p(k)$ with the corresponding network outputs $y_i(k)$, $i = 1, 2, 3$. The performance of the 6-mode CULTN closely matched that of the 4/4-mode IULTN but was inferior to that of the 6/6-mode IULTN. Hence, an 8-mode ($N_x = 16$) CULTN was considered, and its performance was comparable to that of 6/6-mode IULTN. From Figure 8, it can be observed that the LCLM eigenvalues for the two networks are distributed closely around the unit circle in the complex z-plane.

Figure 9 shows the autonomously generated outputs of the 8-mode CULTN and the 6/6-mode IULTN. Qualitatively it can be seen that both networks can generate and sustain chaotic waveforms with attractor contours similar to that of the target. The two basins of attraction are distinguishable, and orbit switching between them is retained Quantitative performance was assessed using symmetric KL divergences (33) for multiple ensembles of autonomous generation. The ensembles are computed for quantization levels N_r in the range (4:44), by setting (a) the time instant for transfer from open-loop to closed-loop in the range (500:2500) and (b) the covariance parameters N_c in the range (20:200). Figure 10 shows the ensemble average and standard deviation of the KL divergences for the 8-mode CULTN and 4/4 mode, and 6/6-mode IULTN. The average KL divergence of the 6/6-mode IULTN is consistently lower than that of the 8-mode CULTN with a lower standard deviation. Even the 4/4-mode IULTN exhibited marginally better performance than the 8-mode CULTN.

In summary, IULTN with fewer weights ($N_w^I = 792$) outperformed CULTN ($N_w^C = 1056$) both qualitatively and quantitatively in the generative modeling of the Lorenz limit

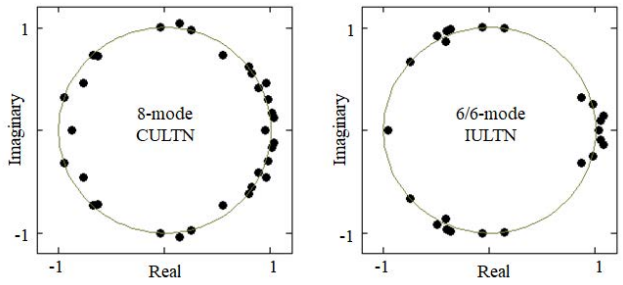

FIGURE 8. Lorenz limit cycles: LCLM eigenvalue placement in the complex z-plane.

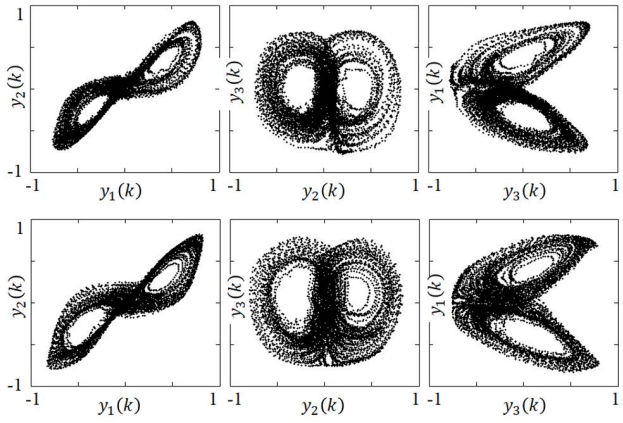

FIGURE 9. Lorenz limit cycles: autonomously generated trajectories with 8-mode CULTN (top) and 6/6 mode IULTN (bottom).

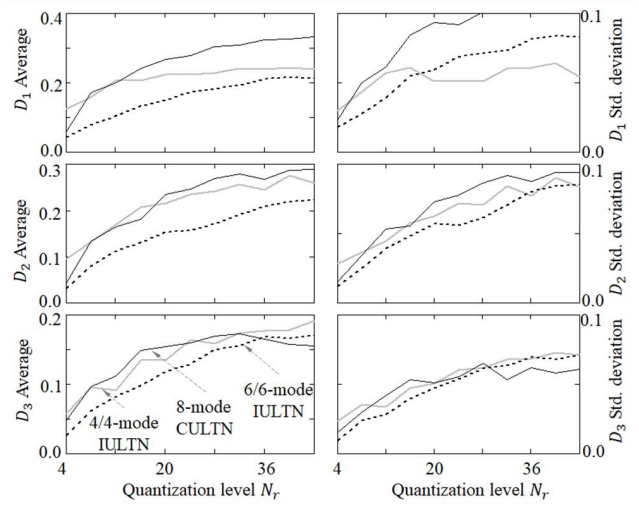

FIGURE 10. Generating lorenz limit cycles: ensemble average and standard deviation of KL divergences vs. quantization level (top: first output, middle: second output, bottom: third output); gray line: 4/4 mode IULTN, dash line: 6/6-mode IULTN, dark thin line: 8-mode CULTN.

cycles. A similar task considered in [23] used a reservoir computing network with an estimated 1800 trainable weights.

B. SANTA FE LASER DATA

We considered autonomously generating synthetic data similar to the Santa Fe laser dataset [12]. The original dataset consists of periodic to chaotic intensity pulsations of a far-infrared laser. The laser output exhibited significant dissymmetry around the mean value and mimicked a chaotic low-frequency waveform modulating a high-frequency carrier as shown in Figure 11.

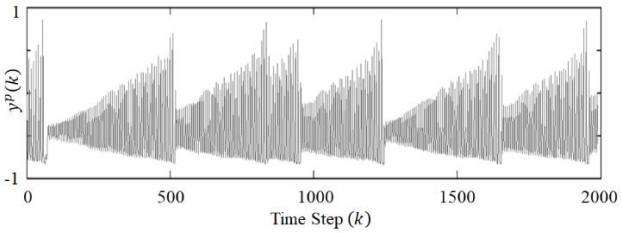

FIGURE 11. Santa Fe laser data: target waveform.

First, a 4-mode ($N_x = 8$) CULTN and a 4/4-mode ($N_x = N_z = 8$) IULTN were considered. With the laser data as the target input, a 1-input 2-output network was used with the outputs being the 1-step and 4-step prediction values. To improve the learning robustness, a second output is included to enhance the network's ability to capture the structure of the laser data better, for example, its rise and collapse. In addition, white noise with a variance of 0.01 is added to the input. The networks were trained with the first 1500 data points of the original data series "A.Cont," with an epoch size of $N_q = 100$. The networks were first trained in an open loop and then in a closed loop with γ in (35) reduced from 1 to 0.5 such that the network relied more on its prediction. Figure 12 shows that the LCLM eigenvalues of the two networks were distributed within an annular ring closely around the unit circle in the complex z-plane.

The generative performance of the trained networks was assessed by replacing the target $y^p(k)$ with the mean of the predicted network outputs $y(k)$ and $y(k-4)$. The performance of the 4-mode CULTN is significantly inferior to that of the 4/4-mode IULTN. Hence, an 8-mode CULTN was considered to have the performance comparable to that of a 4/4 mode IULTN. Figure 13 shows the CULTN and IULTN output time plots for 2000 sampling instants. The first 200 points show the network output with the target input, and the remaining autonomously generated. It is qualitatively evident that both networks can sustain high-frequency oscillations and chaotic low-frequency variations similar to the original laser data. However, the CULTN waveform has 'softer' valleys with higher values at which the envelope collapses compared to that of the IULTN. The IULTN output was more similar to the target (Figure 11) with a sharper rise and lower valleys of the envelope.

Figure 14 compares the representative covariance phase plots of the autonomously generated outputs with those of the target. The phase plots describe $y(k+50)$ vs. $y(k)$ where the time step 50 is arbitrarily chosen to represent the fractional period of the dominant low-frequency envelope of the target. It can be seen that the contour edges of the phase plots of

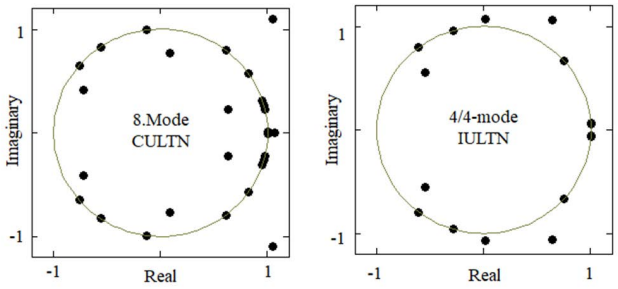

FIGURE 12. Santa Fe laser data: LCLM eigenvalue placement in the complex z-plane.

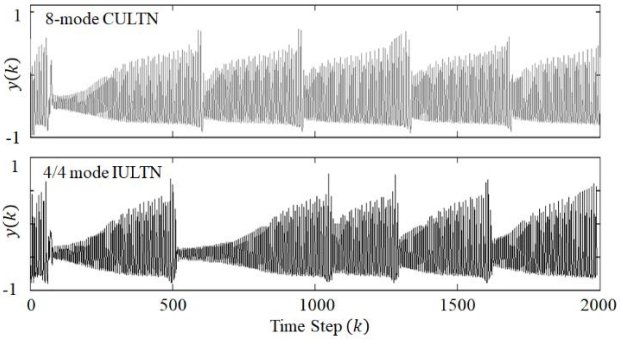

FIGURE 13. Santa Fe laser data: autonomously generated network outputs (top: 8-mode CULTN, bottom: 4/4 mode IULTN).

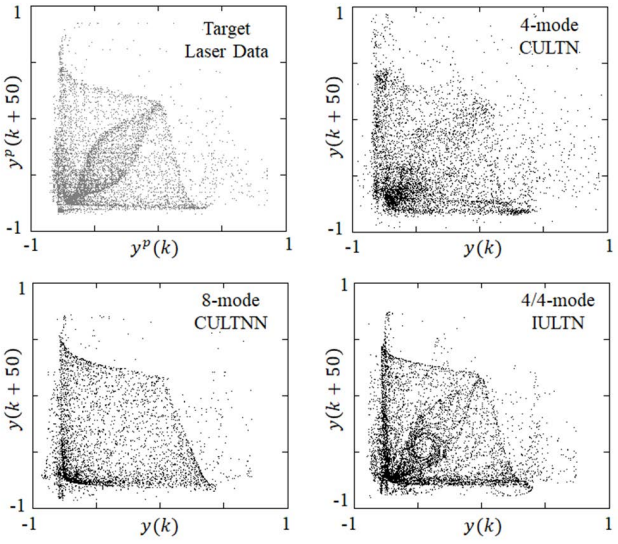

FIGURE 14. Santa Fe laser data: covariance phase plots of original laser data (top left) and autonomously generated outputs (top right: 4-mode CULTN, bottom left: 8-mode CULTN, bottom right: 4/4-mode IULTN.

the 4-mode CULTN are fuzzier than those of the target. The contour edges of the 8-mode CULTN phase plots are sharper and are similar to the target. The phase plots of the 4-mode IULTN are the closest to the target, including a distinct eye-like feature.

Quantitative performance was assessed using symmetric KL divergences (33) for multiple ensembles of autonomous generation. The ensembles were computed for quantization levels N_r in the range (4:44) and setting the covariance parameters N_c in the range (20:200). Figure 15 shows the ensemble average and standard deviation of the KL divergences as a function of the quantization level N_r. The average KL divergence of the 4/4-mode IULTN was consistently lower with a smaller standard deviation than the 4-mode CULTN and closer to that of the 8-mode CULTN.

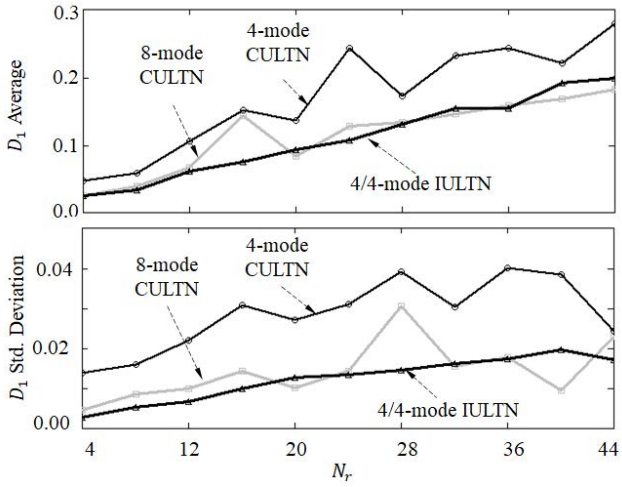

FIGURE 15. Santa Fe laser data: KL divergences of autonomously generated outputs.

In summary, IULTN with far fewer weights ($N_w^I = 320$) outperformed CULTN ($N_w^C = 896$) in the generative modeling of Santa Fe Laser data. A similar task considered in [10] with a multilayer perceptron network used an estimated 1300 trainable weights.

C. KOLAM PATTERN

Kolam is a floor art of complex and intricate patterns that is commonly practiced in South India. A kolam is drawn as a continuous line looping or joining straight and curved lines typically around symmetric dot patterns [13]. The mathematical properties implied in kolam design have been studied in [13], [24] and [25]. A single-stroke kolam variant can be represented as a trajectory plot with x and y coordinates. The modeling complexity of a single-stroke kolam lies in the rich harmonic content of the x- and y- variables, and their spatiotemporal synchronization.

Case 1 compares the performance of CULTN and IULTN in the autonomous generation of a single-stroke kolam with 4-4-4-4 dot placements, as shown in Figure 16. A 2-input, 2-output, 6-mode ($N_x = 12$) CULTN, and 5/5-mode ($N_x = N_z = 10$) IULTN were trained to perform 1-step prediction with the target being the Kolam's x and y coordinates. The kolam pattern was periodic normailized with $N_q = 100$. Given that the kolam pattern is periodic, updates to the initial values of the network states for each training epoch used

$\beta = 0.95$ in (36). The network was first trained in an open loop and then in a closed loop with γ in (35) reduced from 1 to 0.5, to improve the replication accuracy. The kolam plots autonomously generated by CULTN and IULTN are shown in Figure 16. Both CULTN and IULTN can stably and continuously generate a kolam pattern. However, the 5/5-mode IULTN, with fewer weights ($N_w^I = 520$), outperforms the 6-mode CULTN ($N_w^C = 576$) in terms of accuracy and symmetry around the dot pattern, as confirmed by the average and standard deviation of the errors. Errors were obtained by comparing the distances between the generated pattern and target to the nearest dot.

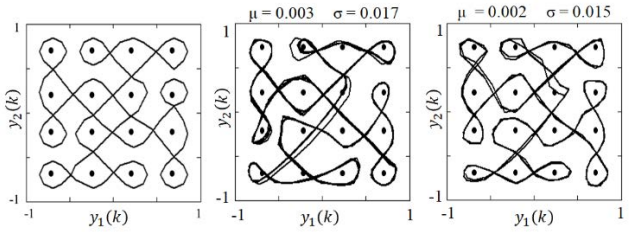

FIGURE 16. Kolam with 4-4-4-4 dot pattern: target waveform (left), autonomously generated output with 6-mode CULTN (middle), 5/5-mode IULTN (right). (μ: error average; σ: error standard deviation).

Case 2 shown in Figure 17 is the autonomously generated plot by a set of nine 6/6-mode ($N_x = N_z = 12$) IULTNs, each trained to reproduce one of the symmetric one-stroke kolams drawn around 1-3-5-3-1 dot placements studied in [13]. The low average errors and low standard deviations confirm the quantitative accuracy achieved by IULTN.

D. ELECTROCARDIOGRAM (ECG) WAVEFORM

Case 1 considers several 2-input, 2-output CULTN, and IULTNs to autonomously generate the ECG waveform synthesized using the model presented in [14]. As shown in Figure 18, the first input is the target, and the second is a pilot sine wave whose frequency is the same as the fundamental ECG waveform. To improve learning robustness, given that segments of the ECG waveform are nearly flat for a substantial amount of time, the pilot signal plays a key role in the distinct mapping of each ECG segment. The ECG waveform was period normalized with $N_q = 100$. The networks were first trained in an open loop and then in a closed loop with γ in (35) reduced to 0.5, and then to 0.25, such that the network relied more on its prediction as training progressed. To reduce the initial transient errors, the initial values are updated at each cycle with β in (36) set to 0.95.

The autonomously generated outputs, and LCLM eigenvalue distribution of a 4-mode CULTN ($N_x = 8$) are shown in Figure 19. Note that the pilot sine wave was also generated by the networks; hence the process was fully autonomous. The CULTN continuously reproduces the subperiod, shape, and peak values of the P, R, and T waves, and heartbeat with reasonable accuracy. However, it cannot accurately reproduce flat segments.

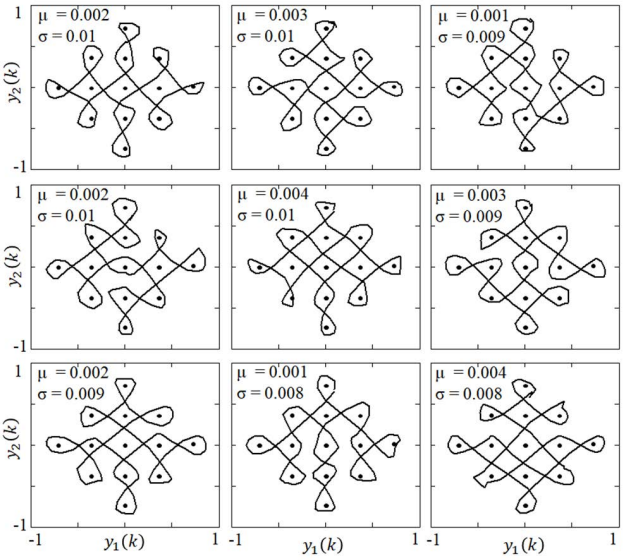

FIGURE 17. Kolam with 1-3-5-3-1 dot pattern: autonomously generated outputs with 6/6-mode IULTNs: KL divergences of autonomously generated outputs. (μ: error average; σ: error standard deviation).

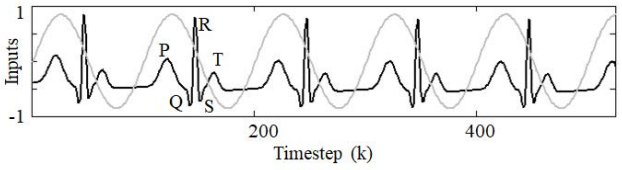

FIGURE 18. ECG: target (dark) and pilot (grey) waveforms.

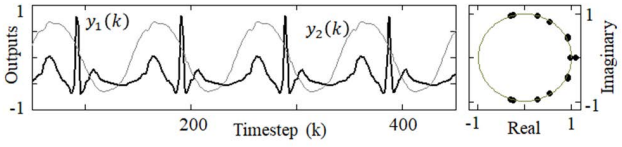

FIGURE 19. ECG: autonomously generated outputs (left) by 4-mode CULTN (ECG output (dark) and pilot (grey); LCLM eigenvalue placement in the complex z-plane (right).

The autonomously generated outputs, and the associated LCLM eigenvalue distributions, of a 2/2-mode ($N_x = N_z = 4$), 3/3 mode ($N_x = N_z = 6$), 3/4 mode ($N_x = 6, N_z = 8$) IULTNs are shown in Figure 20. It can be observed that the 2/2 IULTN, with fewer weights ($N_w^I = 112$), outperformed the 4/4 mode CULTN ($N_w^C = 288$), with P and T waves better reproduced. The reproduction of the flat segment between the T and P waves improved with the 3/3 mode IULTN. The ECG waveform reproduction was significantly improved with the 3/4 mode IULTN, and with approximately the same number of weights ($N_w^I = 276$), it significantly outperforms the 4-mode CULTN ($N_w^C = 288$). Note that the additional dimensional freedom of the 3/4 mode IULTN, with the 3-mode ULTRNN ($N_x = 6$) and 4-mode ISAFN ($N_z = 8$), results in a better reproduction of the valleys in the

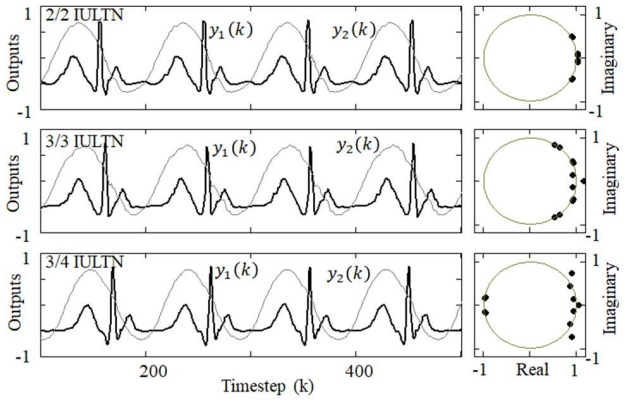

FIGURE 20. ECG: autonomously generated IULTN outputs (left) by 2/2-mode (top), 3/3-mode (middle), 3/4 -mode (bottom) - ECG output (dark) and pilot (grey), and LCLM eigenvalue placement in the complex z-plane (right).

Q and S waves. Relaxing the dimensional constraint may have resulted in a more optimal activation function slope dynamic, thus improving the network memory for modeling.

Case 2 considered the generation of clones with characteristics close to the ECG of a general population using a single network. Similar tasks using generative adversarial networks (GANs) were studied in [26], [27]. The synthesized generation of clone waveforms is useful for data augmentation in biosignal classification, clinical training, and data analysis tasks where real field data are scarce and difficult to collect. In addition, synthetic data are useful for anonymizing and alleviating patient privacy concerns [26]. In the dataset [15], it was noted that ECGs with elevated T waves are normal variants and form a small fraction (<5%) of over 4000 patients with no heart ailments. Such cases may not be well represented in data-modeling tasks, and their synthetic generation may be useful for increasing the data diversity. We consider a 3-input, 2-output, 12/12-mode ($N_x = N_z = 24$) IULTN for the autonomous generation of ECG clones with elevated T waves. We used the 50 samples drawn from [15] as the training set. Each training sample was period normalized with $N_q = 100$. While the first and second inputs are the target ECG and pilot sine wave, the third input is a random seed drawn from a Gaussian noise generator and is associated with each target ECG. The outputs were one-step predicted values of the target and the pilot sine wave. The network was first trained in the open loop, and then in the closed loop with γ gradually reduced to 0.5. During closed-loop training, only the first two inputs use output feedback, and the noise seed is retained as the external third input.

Figure 21 shows the representative target ECGs, along with the corresponding autonomously generated outputs when fed with the training seeds. In addition, the test outputs were autonomously generated using new Gaussian noise inputs that were different from the training seeds. The generative capability of IULTN is evident from cloned outputs that have structural similarity to the training set, while exhibiting

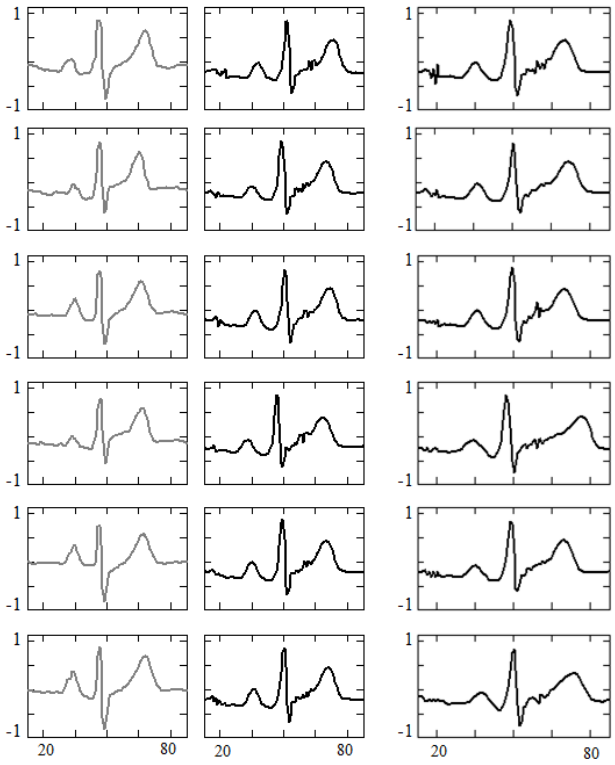

FIGURE 21. ECG with elevated-T segments: target waveforms (left), autonomously generated outputs by 12/12-mode IULTN with training seed (middle) and with random seed (right); x-axis: time steps; y-axis: target (left) and network outputs (middle and right).

natural diversity. This example demonstrates the ability of the IULTN to autonomously generate complex biosignals with characteristics similar to those of a parent population. Reference [26] considered an LSTM-based deep GAN with 200 hidden units and three layers for an estimated 800000 trainable weights for each of the generator and discriminator networks. Comparable results were obtained with IULTN, which uses only 2880 trainable weights, a minuscule fraction (0.2%) of LSTM-GAN [26].

E. STOCK DATA MODELING

We consider synthetically generating nonstationary time- series data with a probability distribution that best approximates that of the parent. We specifically modeled the distribution of Google's stock data previously studied in [16] which used TimeGAN. TimeGAN uses an embedding subnetwork and a recovery subnetwork to learn the underlying temporal dynamics, in addition to the discriminator and generator subnetworks common to a generic GAN. The training process has three stages: the first for the embedding subnetwork, the second for the full network, but with only supervised loss, and the third for a joint process considering all losses. TimeGAN exhibits the best performance among competing techniques for a range of time-series generative tasks [16].

The stock dataset consists of six variables: open, low, high, closing, and adjusted closing stock prices, and the stock trading volume for each trading day for the years 2004-2019. The objective of this exercise is to synthetically generate stock data over a 48-day period that mimics the Google stock data distribution. Stock data exhibit a high feature correlation among the five pricing variables. A 48-day period is considered here, as the temporal correlation for the 24-day periods studied in [16] is high, but significantly reduced for 48-day periods. We trained a 7-input 6-output 10/10 mode ($N_x = N_z = 10$) IULTN to perform generation through a one-step prediction. The first six inputs are the stock data variables. To improve learning robustness, the seventh is Gaussian noise vector that better maps the temporal variability in any period. The network was first trained in the open loop and then in the closed loop, with γ in (35) progressively reduced to 0.25. The network was tested for autonomous generation of stock data using Gaussian noise as the external input. TimeGANs with four, six, and eight gated recurrent units (GRUs) for each of the six stock variables, similar to the one studied in [16] were trained, and the one with the eight GRUs that had the best performance was chosen for comparison with the IULTN. The key comparative items of the network architecture, its dimensions, and the computational burden are listed in Table 4. As shown in the table, the number of trainable parameters for IULTN (2400) is a small fraction (∼1.4%) of that of TimeGAN (174727). This is reflected in the substantially low computation time per epoch iteration for IULTN training (∼15.3% of TimeGAN). The number of generative parameters required for the trained IULTN remains at 2400 and is still a small fraction (∼6.2%) of the trained TimeGAN (38400).

TABLE 4. Stock data modeling – TimeGAN vs IULTN.

Category	TimeGAN	IULTN
Training parameters	174727	2400
Generating parameters	38400	2400
The relative computational time for training	6.55	1.0 (Normalization base)

Figure 22 shows a qualitative comparison of the IULTN-generated data with that of TimeGAN. The visualization plots compare the principal component analysis (PCA) [28] and t-distributed stochastic neighbor embedding (t-SNE) [29] maps of the synthetically generated data with those of the original data. It is apparent that the distribution of the IULTN-generated data closely follows that of the original data and is comparable to that of the TimeGAN-generated data.

A quantitative comparison was performed using the train on the synthetic test on real (TSTR) technique using generic assessment networks based on GRU [16]. The first assessment network was trained to predict the closing stock price using training data extracted from the synthetic generation, with the trading volume data discarded. Next, the second assessment network with the same structure and dimensions

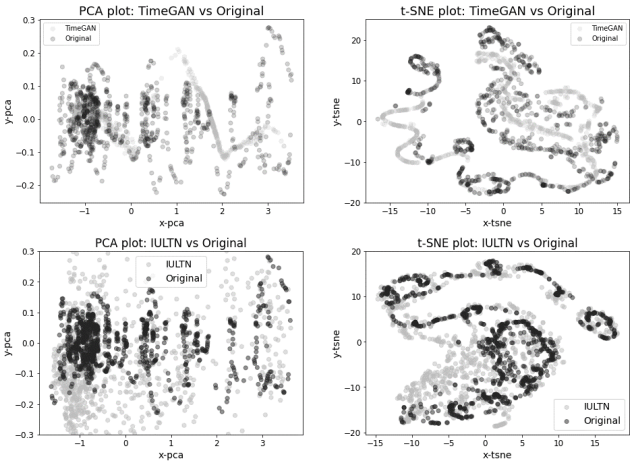

FIGURE 22. Stock Data: Comparative PCA (left) and t-SNE (right) plots of TimeGAN (top) and IULTN (bottom) generated outputs.

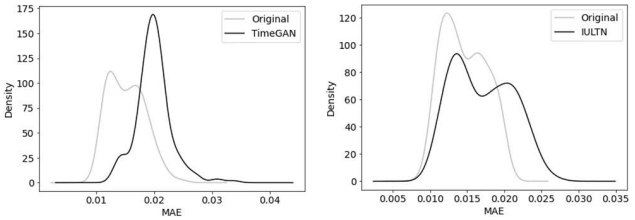

FIGURE 23. Stock Data: Comparative MAE density map of predictions with TimeGAN (left) and IULTN (right) generated outputs.

as the first was trained using the training data extracted from the original data. The prediction performance of both networks was tested using fresh test data extracted from the original data. The mean absolute errors (MAE) of the predictions obtained using the two assessment networks collected over 200 train and test runs were used as comparison metric. The prediction MAE densities obtained for the TimeGAN- and IULTN-generated data are shown in Figure 23. It is clear from the figure that the prediction MAE with IULTN-generated data is closer to that obtained with the original data than that of TimeGAN. Specifically, the peak MAE of the IULTN case (0.0135) occurred in close proximity to that corresponding to the original data (0.0125). However, an accurate objective comparison with TimeGAN is not possible because the network dimensions and training are not fully optimized in either case.

In summary, the generative performance of IULTN compares well with that of TimeGAN both qualitatively and quantitatively with far fewer weights, a simpler training process, and lower computational times.

F. SMART GRID DATA MODELING

The objective of this exercise, similar to the task addressed in [18], is to synthetically generate energy consumption and solar generation data with the same probabilistic distribution as that of the Pecan Street Dataset [17]. We observed that

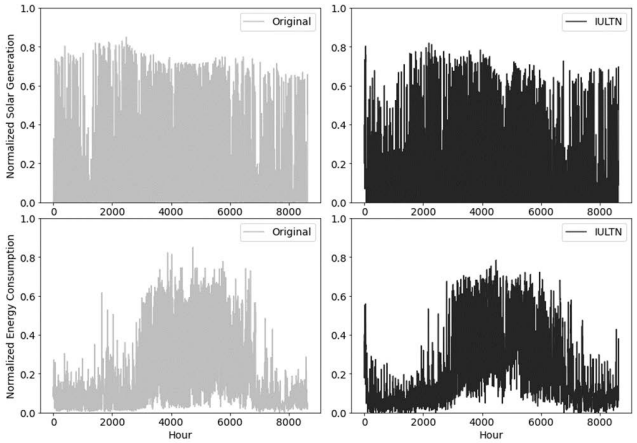

FIGURE 24. Smart grid data: Comparative solar generation (top) and energy consumption (bottom) data: original (left) and IULTN generated outputs (right).

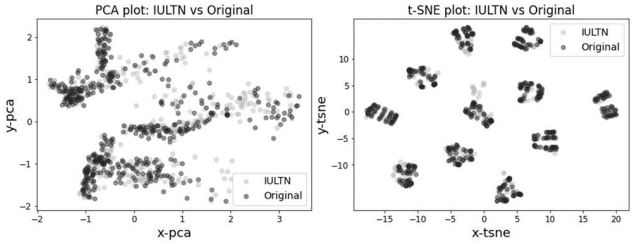

FIGURE 25. Smart grid data: Comparative PCA (left) and t-SNE (right) plots of IULTN generated outputs vs. that of the original data.

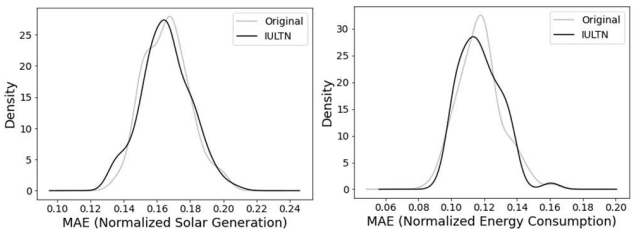

FIGURE 26. Smart grid data: Comparative MAE density map of predictions with IULTN (right) generated outputs vs. that of the original data.

the solar generation data for multiple users with installed PV panels exhibited repetitive patterns with variations owing to the onset of clouds and rain. Energy consumption data also exhibit similar trends but are accompanied by irregular spikes because of random consumption. The training data comprised hourly data of nine users in Austin collected over a year. We trained a 23-input 22-output 15/15 mode ($N_x = N_z = 15$) IULTN to perform autonomous generation through one-step prediction. The first 18 inputs and outputs correspond to the solar generation and energy consumption data of nine users. The 4 inputs/outputs were the pilot sine waves to represent the periodicity of the daily, weekly, and annual variations. A Gaussian noise seed is used as the final input. The autonomous generation capability of the network was tested using pilot waves and a Gaussian noise seed as the only external inputs. Figure 24 shows the IULTN-generated solar generation and energy consumption data over a one-year period for a representative user, compared with the corresponding original data.

Figure 25 shows a qualitative assessment of the IULTN-generated data using visualization plots based on PCA [28] and t-SNE [29] maps. It is apparent that the distribution of IULTN-generated data closely follows that of the original data.

A quantitative comparison was performed using the TSTR technique with assessment networks based on GRU [16], similar to the one considered in Section E for stock data modeling. The two assessment networks were trained for one-step prediction, the first with the original data and the second with the IULTN-generated data. The MAEs of the solar and energy data prediction for the nine users collected over 200 train and test runs are shown in Figure 26. It is clear from Figure 26 that the MAE density of the predictions with the IULTN-generated data closely matched that of the original data. An objective comparison with the results of [18] is not possible because of the sparsity of network information.

In summary, the autonomously generated IULTN data compares well with the original solar generation and energy consumption data both qualitatively and quantitatively.

VI. CONCLUSION

The main contributions of this study are as follows: (1) Activation function slope modulation with a recurrent ISAFN architecture where the states and dimensions are decoupled from the ULTRNN for better control of memory retention. (2) Zigzag propagation algorithm for the weight updates to account for the state interactions in the IULTN. A truncation technique is presented that significantly reduces the speed of gradient computation without significantly affecting accuracy. (3) Closed-loop training with an eigenvalue metric of the linearized closed-loop system to ensure sustained output generation emulating the target. The effectiveness of the proposed approach was demonstrated with applications in synthetic data generation encompassing chaotic systems, complex periodic patterns, biosignals, and nonstationary processes.

We developed a novel method to dynamically vary the activation function slopes of a ULTRNN by using a secondary network. Varying memory retention over time allows the network to capture short-term features while continuing to remember long-term dependencies, which is a key requirement for the generative modeling of time series. We compared two architectures for the secondary network, CSAFN and ISAFN. The CSAFN is a feedforward architecture that uses the main ULTRNN states to compute activation function slopes. This simplifies training by directly extending the backpropagation algorithm. The ISAFN uses a recurrent architecture with independent states to compute activation function slopes. The interaction between the ISAFN and

ULTRNN states makes the z-prop training process complex. The biggest advantage of the ISAFN over the CSAFN is its recurrent architecture which aids in controlling long-term memory. The ability of the CSAFN to build long-term memory is limited by its feedforward architecture. The independent recurrent states of the ISAFN are more effective in the dynamic shaping of the activation function slopes which results in enhanced control of memory retention. This is exemplified in the generative experimental results where the IULTN accurately captures nuanced features of the target data such as the sharp collapse of the Santa Fe laser data followed by a low frequency ramp rise, and the flat segments of the ECG waveform. Even with increased network dimensions, the CULTN is unable to match the performance of the IULTN.

The marginal stability of the triangular state-feedback submatrices of the ULTRNN ensures stable and robust training of the open-loop system, because the gradients are not allowed to either explode or vanish. In generative tasks, with the predicted output fed back as the input to generate the next prediction, it is essential to ensure marginal stability of the closed-loop system. An eigenvalue metric of the linearized closed-loop system was employed to ensure that the network behaved as a nonlinear oscillator whose output neither saturated nor collapsed. Maintaining the marginal stability of both open- and closed-loop systems inherently allows for the robust and stable learning of the target features. Robust learning was enhanced by noise injection, n-step prediction, and pilot waves.

Examples presented demonstrate that the autonomous waveforms generated by the IULTN have qualitative and quantitative features closely similar to the target, and that the IULTN outperforms state-of-the-art neural network architectures with a smaller number of trainable weights and lower training times, as shown in Table 5. Specifically, the IULTN uses only a very small fraction of the weights when compared with deep GAN networks. The better performance achieved by the IULTN is at the cost of increased gradient computations imposed by the zig-zag propagation algorithm and the eigenvalue based marginal stability computations required for stable learning. Future work will explore additional means to speed up gradient computations without impacting learning performance. It will also study the impact of hyper-parameter selection on controlling memory retention and its application to modeling complex multivariate time series tasks such as bio signals.

TABLE 5. Comparative architecture & weight profiles.

Task	Previous works			IULTN Trainable Weights
	Reference	Architecture	Estimated Trainable Weights	
Lorenz limit cycles	[23]	Reservoir Computing	~1800	792
Santa Fe laser	[10]	Multilayer feedforward	~1300	320
ECG (Case 2)	[26]	GAN (with LSTM)	~1.6M	2688
Stock Pricing	[16)	TimeGAN (with GRU)	~175K	2400

REFERENCES

[1] S. C. Sivakumar, W. Robertson, and W. J. Phillips, "Online stabilization of block-diagonal recurrent neural networks," *IEEE Trans. Neural Netw.*, vol. 10, no. 1, pp. 167–175, Jan. 1999.
[2] S. Sivakumar and S. Sivakumar, "Marginally stable triangular recurrent neural network architecture for time series prediction," *IEEE Trans. Cybern.*, vol. 48, no. 10, pp. 2836–2850, Oct. 2018.
[3] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.
[4] P. A. Mastorocostas and J. B. Theocharis, "A stable learning algorithm for block-diagonal recurrent neural networks: Application to the analysis of lung sounds," *IEEE Trans. Syst., Man, Cybern., B (Cybern.)*, vol. 36, no. 2, pp. 242–254, Apr. 2006.
[5] P. A. Mastorocostas, C. H. Hilas, D. Varsamis, and S. Dova, "A recurrent neural network-based forecasting system for telecommunications call volume," *Appl. Math. Inf. Sci.*, vol. 7, no. 5, pp. 1643–1650, May 2013.
[6] P. Campolucci, "A circuit theory approach to recurrent neural network architecture and learning methods," Ph.D. dissertation, Electrotech. Eng., Univ. Bologna, Bologna, Italy, 1998.
[7] O. Adigun and B. Kosko, "Using noise to speed up video classification with recurrent backpropagation," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, May 2017, pp. 108–115.
[8] S. Sivakumar and S. Sivakumar, "Modulation of activation function in triangular recurrent neural networks for time series modeling," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2019, pp. 1–8.
[9] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, May 2015.
[10] R. Bakker, J. C. Schouten, C. L. Giles, F. Takens, and C. M. V. D. Bleek, "Learning chaotic attractors by neural networks," *Neural Comput.*, vol. 12, no. 10, pp. 2355–2383, Oct. 2000.
[11] E. N. Lorenz, "Deterministic nonperiodic flow," *J. Atmos. Sci.*, vol. 20, no. 2, pp. 130–141, 1963.
[12] A. S. Weigend and N. A. Gershenfeld, *Time Series Prediction*. Reading, MA, USA: Addison-Wesley, 1994.
[13] K. Yanagisawa and S. Nagata, "Fundamental study on design system of kolam pattern," *Forma*, vol. 22, no. 1, pp. 31–46, 2007.
[14] P. E. McSharry, G. D. Clifford, L. Tarassenko, and L. A. Smith, "A dynamical model for generating synthetic electrocardiogram signals," *IEEE Trans. Biomed. Eng.*, vol. 50, no. 3, pp. 289–294, Mar. 2003.
[15] R. Bousseljot, D. Kreiseler, and A. Schnabel, "Nutzung der EKG-signaldatenbank CARDIODAT der PTB über das internet," *Biomedizinische Technik, ECG Heartbeat Categorization Dataset, Segmented Preprocessed ECG Signals Heartbeat Classification*, vol. 40, no. 5, p. 317, 1995.
[16] J. Yoon, D. Jarrett, and M. van der Schaar, "Time-series generative adversarial networks," in *Proc. 33rd Conf. Neural Inf. Process. Syst. (NeurIPS)*, Vancouver, BC, Canada, 2019.
[17] (2018). *Pecan Streen Dataset*. [Online]. Available: http://www.pecanstreet.org/category/dataport/
[18] C. Zhang, S. R. Kuppannagari, R. Kannan, and V. K. Prasanna, "Generative adversarial network for synthetic time series data generation in smart grids," in *Proc. IEEE Int. Conf. Commun., Control, Comput. Technol. Smart Grids (SmartGridComm)*, Oct. 2018, pp. 1–6.
[19] V. Kozyakin, "On accuracy of approximation of the spectral radius by the Gelfand formula," *Linear Algebra Appl.*, vol. 431, no. 11, pp. 2134–2141, Nov. 2009.
[20] H. Jaeger, "The 'echo state' approach to analysing and training recurrent neural networks-with an erratum note," Fraunhofer Inst. Auton. Intell. Syst., German Nat. Res. Center Inf. Technol., Tech. Rep. GMD148, 2001.
[21] K. Jim, C. Giles, and B. Horne, "An analysis of noise in recurrent neural networks: Convergence and generalization," *IEEE Trans. Neural Netw.*, vol. 7, no. 6, pp. 1424–1438, Apr. 1996.
[22] D. J. C. MacKay, *Information Theory, Inference, and Learning Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2003, p. 34.
[23] J. Pathak, Z. Lu, B. R. Hunt, M. Girvan, and E. Ott, "Using machine learning to replicate chaotic attractors and calculate Lyapunov exponents from data," *Chaos: Interdiscipl. J. Nonlinear Sci.*, vol. 27, no. 12, Dec. 2017, Art. no. 121102.

[24] G. Siromoney, R. Siromoney, and K. Krithivasan, "Array grammars and kolam," *Comput. Graph. Image Process.*, vol. 3, no. 1, pp. 63–82, Mar. 1974.
[25] M. Asher, "The kolam tradition: A tradition of figure-drawing in southern India expresses mathematical ideas and has attracted the attention of computer science," *Amer. Scientist*, vol. 90, no. 1, pp. 56–63, Jan./Feb. 2002.
[26] S. Haradal, H. Hayashi, and S. Uchida, "Biosignal data augmentation based on generative adversarial networks," in *Proc. 40th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*, Jul. 2018, pp. 368–371.
[27] T. G. Technion and K. Radinsky, "Pgans: Personalized generative adversarial networks for ecg synthesis to improve patient-specific deep ECG classification," in *Proc. AAAI Conf. Artif. Intell.*, 2019, vol. 33, no. 1, pp. 557–564.
[28] F. B. Bryant and P. R. Yarnold, "Principal-components analysis and exploratory and confirmatory factor analysis," in *Reading and Understanding Multivariate Statistics*. Washington, DC, USA: American Psychological Association, 1995, pp. 99–136.
[29] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2606, Nov. 2008.

SESHADRI SIVAKUMAR (Senior Member, IEEE) received the B.E. degree in electrical technology and electronics from the Indian Institute of Science, Bengaluru, India, in 1977, and the M.Sc.E. and Ph.D. degrees in electrical engineering from the University of New Brunswick, Fredericton, NB, Canada, in 1983 and 1987, respectively.

He is currently the Chief Consultant with Pasumai Energytech LLC, Bradenton, FL, USA, and also with Healy Wave Energy LLC, working on the assignment of developing smart control and power electronic systems for a wave energy converter. He has led engineering and research and development assignments on power electronic products and systems at Bharat Heavy Electricals Ltd., Bengaluru, from 1977 to 1981; Pivotal Power Inc., Bedford, NS, Canada, from 1987 to 2006; MKS Instruments Ltd., Rochester, NY, USA from 2007 to 2009; United Technologies Research Center, East Hartford, CT, USA, from 2009 to 2010; and SunPower Corporation, Richmond, CA, USA, from 2010 to 2016. He was an Adjunct Faculty in electrical engineering with Dalhousie University, Halifax, NS, Canada, from 2004 to 2012. His current research interests include machine learning algorithms, power electronic topologies, and control systems for alternative and distributed energy applications.

SHYAMALA SIVAKUMAR (Senior Member, IEEE) received the B.E. degree in electrical engineering from Bangalore University, India, in 1984, and the M.A.Sc. and Ph.D. degrees in electrical engineering from the Technical University of Nova Scotia (currently Dalhousie University), Halifax, NS, Canada, in 1992 and 1997, respectively.

From 1997 to 1999, she was a Postdoctoral Research Fellow with the Internetworking Program, Dalhousie University. She started her career as an Aeronautical Engineer with Hindustan Aeronautics Ltd., Bengaluru, India, from 1985 to 1989. Her academic career began as an Assistant Professor with the Internetworking Program, Dalhousie University, from 1999 to 2000. She moved to Saint Mary's University, Halifax, in 2000, where she is currently a Professor of computer and information systems with the Sobey School of Business. She has also been an Adjunct Faculty of engineering mathematics and internetworking with Dalhousie University, since 2000. Her research interests include machine learning architectures and algorithms and recurrent neural networks in time-series prediction tasks.