The impact of training strategies on the stability of autonomous forecasting in a reservoir computing-based digital twin of the Lorenz system

Sofia Tkach
Research Institute of Applied Artificial Intelligence and Digital Solutions
Plekhanov Russian University of Economics
Moscow, Russia
sofata31@gmail.com

Andrey Andreev
Research Institute of Applied Artificial Intelligence and Digital Solutions
Plekhanov Russian University of Economics
Moscow, Russia
andreevandrei1993@gmail.com

2026 10th Scientific School Dynamics of Complex Networks and their Applications (DCNA) | 979-8-3195-2599-4/26/$31.00 ©2026 IEEE | DOI: 10.1109/DCNA72004.2026.11733456

***Abstract*—Reservoir computing efficiently learns chaotic dynamics, making it a well-suited candidate for constructing the predictive core of digital twins. A key challenge is the rapid divergence of the autonomously generated forecast from the true trajectory: small one-step errors are fed back as inputs and grow exponentially due to the intrinsic instability of chaotic dynamics. One way to mitigate this effect is to train the model under conditions that resemble autonomous generation, which can improve its resistance to self-generated errors. This paper compares three training strategies: classical, noise-additive, and phase-alternating (teacher-forcing and self-generated phases) training, applied to the Lorenz system. We evaluate these approaches on 1000 trajectories generated from random initial conditions and demonstrate that both modified strategies improve trajectory retention over the classical baseline. Phase-alternating training performs best, reducing the NRMSE by 4.8% and the mean error by 6.6%, and increasing the valid prediction time by 3.8% at a 750-step horizon. Noise-additive training extends the valid prediction time by 2.4% at no additional training cost ($0.13\,\mathrm{s}$ versus $0.35\,\mathrm{s}$ for phase-alternating training).**

***Index Terms*—reservoir computing, digital twin, Lorenz system, training approaches, noise-additive training, phase-alternating training**

I. Introduction

A digital twin is a virtual counterpart of a physical system that is continuously synchronized with it through data and is used to monitor, forecast and control its behavior [1]–[3]. The central element of any predictive digital twin is a dynamical model able to propagate the current state of the system forward in time [2]. For many real systems, first-principles models are unavailable, incomplete, or too expensive to evaluate in real time, which motivates the use of data-driven surrogates, including models that clone the dynamics of systems with hidden variables and stochastic components [4]. When the underlying dynamics are nonlinear and chaotic, the surrogate must not only fit the observed data but also remain stable when it is run forward on its own.

Reservoir computing (RC) [5], [6] and, in particular, echo state networks (ESNs) have proven to be highly effective for learning chaotic dynamics from time series [7]–[9]. In an ESN, a large, fixed, randomly connected recurrent network (the reservoir) nonlinearly expands the input signal, and only a linear readout is trained, typically by ridge regression [10]. Training therefore reduces to a single linear least-squares problem, which is much faster than training by backpropagation through time and makes RC attractive for digital twins that must be retrained or adapted on the fly [11]. RC models have been shown to reproduce the climate of chaotic attractors and estimate their Lyapunov spectra [8], to infer unmeasured variables [12], and to forecast large spatiotemporal systems [9]; related variants include clique-based reservoir topologies [13] and next-generation RC based on nonlinear vector autoregression [14]–[16]. Beyond benchmark systems, RC has been used to recover hidden variables and reconstruct experimental brain signals [12], [15]–[17], to classify regular and irregular time series [18], and to model and predict stochastic neuron dynamics and noise-induced phenomena such as coherence resonance [19]–[21], while small spiking neural networks have been applied to parameter estimation in radiophysical experiments [22].

To be used as a digital twin, an ESN is operated in closed loop: after a synchronization phase driven by measured data, its own prediction is fed back as the next input, and the model evolves autonomously. In this autonomous generation mode, the model faces a mismatch between training and inference. During classical training (teacher forcing [23]), the reservoir is driven exclusively by the true trajectory and never observes its own errors. During inference, however, every small one-step error is fed back as input and, owing to the positive largest Lyapunov exponent of the system, grows exponentially [24], [25]. This discrepancy between training and inference input distributions, known in the sequence-modeling literature as exposure bias, leads to a rapid divergence from the true trajectory

This work was prepared as part of research project No. FSSW-2026-0008 "Development of fundamental principles for constructing digital twins of infrastructure systems in the data economy" using funds from a subsidy from the federal budget within the framework of the state assignment of the Ministry of Education and Science of Russia.

Authorized licensed use limited to: Kant Baltic Federal Univ. Downloaded on October 23,2026 at 15:27:27 UTC from IEEE Xplore. Restrictions apply.

and, in some cases, to a collapse of the generated dynamics onto non-physical attractors [26], [27]. For stochastic systems, the quality of RC predictions has also been shown to exhibit intermittent behavior [28].

Several remedies have been proposed. Injecting small noise into the inputs during training is a common means of stabilizing the generative operation of ESNs [10], since it acts as a regularizer; recent work has proposed noise-inspired regularizers that directly target the stability of closed-loop RC forecasts [27]. For gradient-trained recurrent networks, scheduled sampling [29] reduces exposure bias by exposing the network to its own predictions during training. Other approaches improve forecasts by incorporating prior information about the data into the reservoir [30]. Alternatively, the state of an RC-based digital twin can be corrected during operation by periodically injecting measured data [31]; this, however, requires regular access to measurements, whereas the present study aims to improve the stability of the model itself at the training stage. However, scheduled sampling is formulated for multi-epoch gradient descent and cannot be applied directly to an ESN whose readout is obtained in a single regression step. Moreover, comparisons of training strategies are often performed on a few trajectories, whereas forecast quality for chaotic systems varies strongly with the initial condition [26].

In this paper, we systematically compare three training strategies for an ESN-based predictive core of a digital twin of the Lorenz system: (i) classical teacher forcing, (ii) noise-additive training, in which Gaussian noise is added to the inputs, and (iii) phase-alternating training, a single-shot analogue of scheduled sampling in which fixed-length teacher-forcing and self-generated phases alternate periodically. The rest of the paper is organized as follows. Section II describes the Lorenz system, the ESN architecture, the training strategies, and the evaluation protocol. Section III presents the results, and Section IV concludes the paper and outlines directions for future work.

II. Methods

A. The Lorenz system

The methods are evaluated on the Lorenz system [24], a canonical benchmark for chaotic dynamics, defined by

$$\dot{x} = \sigma(y - x), \tag{1}$$

$$\dot{y} = x(\rho - z) - y, \tag{2}$$

$$\dot{z} = xy - \beta z, \tag{3}$$

with the set of standard parameters $\sigma = 10$, $\rho = 28$, $\beta = 8/3$. Trajectories are generated by 4th-order Runge–Kutta integration with a fixed step $\Delta t = 0.01$.

B. Reservoir computing

As the predictive core of the digital twin, we adopt the Echo State Network (ESN) architecture [5], [10], which uses a randomly generated reservoir and trainable linear readout for forecasting the system dynamics. The reservoir state $\mathbf{r}(t) \in \mathbb{R}^N$ is updated according to the equation:

$$\mathbf{r}(t+\Delta t) = (1-\alpha)\mathbf{r}(t) + \alpha \tanh\big(W\mathbf{r}(t) + W_{in}\mathbf{u}(t) + \mathbf{b}\big), \tag{4}$$

where $\mathbf{u}(t) \in \mathbb{R}^3$ is the input state vector, $W \in \mathbb{R}^{N\times N}$ is the recurrent weight matrix, $W_{\text{in}} \in \mathbb{R}^{N\times 3}$ is the input weight matrix, $\mathbf{b} \in \mathbb{R}^N$ is a bias vector, and α is the leak rate. The recurrent matrix W is independently initialized from the standard normal distribution $\mathcal{N}(0,1)$, scaled with spectral radius ρ_W and sparsified according to the connectivity p_{res}. W_{in} has a modified block-structured architecture grouped by input coordinates. Each group is strongly connected to its own coordinate (connectivity density p_{in}, amplitude σ_{in}) and weakly connected to the others ($p_{cross} = 0.15\,p_{in}$, $\sigma_{cross} = 0.15\,\sigma_{in}$), forcing the blocks to partially specialize in individual components of the dynamics. This modification was selected because, in autonomous generation, it outperforms the classical architecture ($\mathrm{NRMSE}_{classic} = 0.1629$ vs $\mathrm{NRMSE}_{separated} = 0.1567$ under identical conditions) while using half as many units ($\mathrm{N}_{classic} = 1000$, $\mathrm{N}_{separated} = 500$) under optimized hyperparameters. The key hyperparameters of the model are summarized in Table I.

TABLE I
Reservoir hyperparameters

Parameter	Value
Number of reservoir units (N)	500
Leak rate (α)	0.25
Spectral radius (ρ_W)	0.60
Input scaling (σ_{in})	0.44
Reservoir connectivity (p_{res})	0.02
Input connectivity (p_{in})	0.21
Ridge regularization (λ)	10^{-4}

The output weights $W_{\text{out}} \in \mathbb{R}^{3\times N}$ are the only trainable parameters of the network. The readout is trained to predict the state one step ahead, $\hat{\mathbf{u}}(t+\Delta t) = W_{\text{out}}\,\mathbf{r}(t+\Delta t)$, where $\mathbf{r}(t+\Delta t)$ is obtained from (4) with input $\mathbf{u}(t)$. The weights are found by ridge regression (Tikhonov regularization) [10], minimizing

$$\mathcal{L}(W_{\text{out}}) = \sum_t \big\| W_{\text{out}}\,\mathbf{r}(t+\Delta t) - \mathbf{u}_{\text{true}}(t+\Delta t)\big\|_2^2 + \lambda \|W_{\text{out}}\|_F^2, \tag{5}$$

where $\lambda = 10^{-4}$ is the regularization parameter and $\|\cdot\|_F$ is the Frobenius norm.

C. Training strategies

Classical training. This strategy uses standard teacher forcing, where the reservoir is driven by the true state $\mathbf{u}_{\text{true}}(t)$, and the readout is trained to predict the next true state $\mathbf{u}_{\text{true}}(t+\Delta t)$ according to (4) and (5)zero-mean Gaussian noise.

Noise-additive training. During training, we add zero-mean Gaussian noise to the input data, while the target variable remains unchanged:

$$\tilde{\mathbf{u}}(t) = \mathbf{u}_{\text{true}}(t) + \boldsymbol{\eta}(t), \quad \boldsymbol{\eta}(t) \sim \mathcal{N}(\mathbf{0}, \sigma_\eta^2 I), \tag{6}$$

Authorized licensed use limited to: Kant Baltic Federal Univ. Downloaded on October 23,2026 at 15:27:27 UTC from IEEE Xplore. Restrictions apply.

where σ_η is the standard deviation of the noise, tuned as a hyperparameter.

Phase-alternating training. This strategy emulates the conditions of autonomous generation by periodically feeding the reservoir with the model's own predictions instead of the true state. Unlike classical scheduled sampling, where the proportion of self-generated values continuously increases across epochs, we apply a fixed-length periodic cycle defined as

$$L = T_{\mathrm{tf}} + T_{\mathrm{sg}}, \tag{7}$$

where T_{tf} and T_{sg} are the numbers of teacher-forcing and self-generated steps, respectively, with $T_{\mathrm{tf}} > T_{\mathrm{sg}}$. This design choice is motivated by the fact that the readout layer is trained in a single step rather than via iterative gradient descent over multiple epochs. To generate training data, we first train a readout under standard teacher forcing. This preliminary readout is used to produce predictions during the self-generated phases of the cycle. Finally, the readout is subsequently trained on the resulting mixed trajectory.

D. Testing

For each training strategy, the model is first trained on a single continuous trajectory of 12000 steps, which is isolated from the testing dataset. The model is then evaluated on an ensemble of $N_{\mathrm{traj}} = 1000$ independent trajectories starting from random initial conditions on the attractor, sampled after discarding a 2000-step transient. All evaluation metrics are calculated across generation horizons ranging from $N_{steps} = 400$ to $N_{steps} = 750$ steps.

E. Evaluation Metrics

For each method, we compute the Normalized Root-Mean-Square Error (NRMSE), the mean Euclidean error ($\mathrm{Err}_{\mathrm{mean}}$), and the valid prediction time (t_V).

The instantaneous prediction error at time step t is defined as the Euclidean distance between the true state $\mathbf{u}(t)$ and the predicted state $\hat{\mathbf{u}}(t)$:

$$e(t) = \|\mathbf{u}(t) - \hat{\mathbf{u}}(t)\|_2. \tag{8}$$

The mean Euclidean error over the evaluation horizon N_{steps} is then given by:

$$\mathrm{Err}_{\mathrm{mean}} = \frac{1}{N_{steps}} \sum_{t=1}^{N_{steps}} e(t). \tag{9}$$

To evaluate the relative trajectory deviation normalized by the full span of the true trajectory, we define the normalized root-mean-square error (NRMSE) as:

$$\mathrm{NRMSE} = \frac{\sqrt{\frac{1}{N_{steps}} \sum_{t=1}^{N_{steps}} e(t)^2}}{\max_t \|\mathbf{u}(t)\|_2 - \min_t \|\mathbf{u}(t)\|_2}. \tag{10}$$

The valid prediction time t_V captures the short-term prediction quality by quantifying the model's capability to closely track the ground-truth trajectory. The normalized prediction error $\delta_V(t)$ is defined as:

$$\delta_V(t) = \frac{\|\mathbf{u}(t) - \hat{\mathbf{u}}(t)\|_2^2}{\langle \|\mathbf{u}(t) - \langle \mathbf{u}(t) \rangle\|_2^2 \rangle}, \tag{11}$$

where $\langle \cdot \rangle$ denotes the time average of the argument up to time t. The valid prediction time is the first time at which $\delta_V(t)$ exceeds the threshold 0.4 [30], expressed in Lyapunov times:

$$t_V = \max\{t \mid \delta_V(t) < 0.4\}. \tag{12}$$

F. Hyperparameters optimization

The noise level σ_η (Eq.6) and T_{tf}, T_{sg} (Eq.7) were optimized to balance prediction accuracy (NRMSE) and the valid predictive time. Models were trained on 12000 time steps and evaluated on 80 validation trajectories generated with four random seeds (20 trajectories per seed) and disjoint from the test ensemble. The selected values are $\sigma_\eta = 0.0037$, $T_{\mathrm{tf}} = 16$, and $T_{\mathrm{sg}} = 10$.

III. Results

The three training strategies: classical, noise-additive, and phase-alternating training were compared on $N_{\mathrm{traj}} = 1000$ independent test trajectories starting from random initial conditions on the Lorenz attractor. Table II reports the mean and standard deviation of each metric across the test trajectories at a 750-step generation horizon.

TABLE II
COMPARISON OF TRAINING STRATEGIES AT 750 STEPS GENERATION HORIZON (MEAN ± STANDARD DEVIATION OVER 1000 TEST TRAJECTORIES; t_V IN LYAPUNOV TIMES).

Strategy	NRMSE	$\mathrm{Err}_{\mathrm{mean}}$	t_V
Classical	0.0943 ± 0.0395	6.04 ± 3.30	4.20 ± 1.39
Noise-additive	0.0914 ± 0.0398	5.80 ± 3.25	4.30 ± 1.39
Phase-alternating	0.0898 ± 0.0398	5.64 ± 3.14	4.36 ± 1.38

Both modified strategies outperform classical training on all three metrics. Relative to the classical baseline, phase-alternating training reduces the NRMSE by approximately 4.8% and $\mathrm{Err}_{\mathrm{mean}}$ by 6.6%, and increases the average t_V by approximately 3.8% (from 4.20 to 4.36 Lyapunov times, i.e., by about 18 integration steps). Noise-additive training yields smaller gains: 3.1% lower NRMSE, 4.0% lower $\mathrm{Err}_{\mathrm{mean}}$, and 2.4% longer t_V. The advantage of phase-alternating over noise-additive training is smaller: 1.8% lower NRMSE and 1.4% higher t_V. Phase-alternating training also yields the smallest spread of $\mathrm{Err}_{\mathrm{mean}}$ (3.14 versus 3.25–3.30 for the other strategies).

Fig. 1 shows the NRMSE and $\mathrm{Err}_{\mathrm{mean}}$ as functions of the generation horizon. The ranking of the strategies (phase-alternating $<$ noise-additive $<$ classical) is preserved over the entire range of $N_{\mathrm{steps}} \in [400, 750]$. As the horizon increases, all errors grow because of the exponential divergence of chaotic trajectories. The absolute gap between the strategies grows slightly with the horizon, whereas the relative gap decreases: for example, the relative reduction in $\mathrm{Err}_{\mathrm{mean}}$

Authorized licensed use limited to: Kant Baltic Federal Univ. Downloaded on October 23,2026 at 15:27:27 UTC from IEEE Xplore. Restrictions apply.

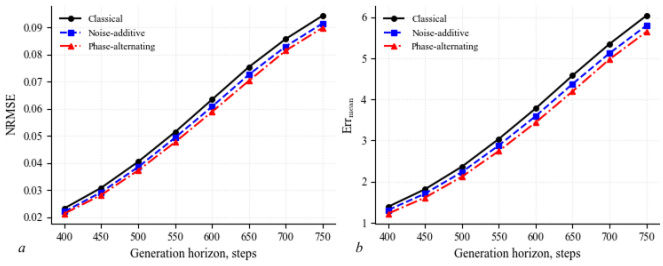


Fig. 1. Forecast error metrics as a function of the generation horizon $N_{steps} \in [400, 750]$ steps for classical, noise-additive, and phase-alternating training strategies: (a) normalized Root-Mean-Square Error (NRMSE); (b) mean Euclidean error ($\mathrm{Err}_{\mathrm{mean}}$).

achieved by phase-alternating training decreases from about 14% at 400 steps to 6.6% at 750 steps. This is expected, since once the forecast has decorrelated from the true trajectory, the error saturates at a level determined by the size of the attractor, regardless of the training strategy.

IV. Conclusion

This paper examined whether the training strategy of an echo state network affects the stability of autonomous forecasting of chaotic dynamics, in terms of both forecast accuracy (NRMSE, $\mathrm{Err}_{\mathrm{mean}}$) and the valid prediction time. Three strategies, namely classical teacher forcing, noise-additive training, and phase-alternating training, were compared on the Lorenz system. Each model was trained once on a 12000-step trajectory and tested on 1000 independent trajectories with random initial conditions on the attractor, for generation horizons from 400 to 750 steps

In terms of steady-state accuracy (NRMSE, Err_{mean}), the differences between the three evaluated strategies are minimal. However, visible differences emerge in the duration the model can sustain trajectory generation within acceptable limits before stable divergence from ground truth. In this regard, phase-alternating training consistently outperforms classical training achieving a lower value of NRMSE and Err_{mean} and the highest valid predictive time among the three strategies, with an advantage of roughly 3.8% over classical training and a smaller, but consistent, advantage over noise-based training across the tested horizon.

Therefore, these training modifications do not compromise accuracy and provide a consistent advantage in trajectory retention time in average, hence they can be used as a practical strategy to enhance the autonomous generation stability of ESN-based digital twins for chaotic systems. It is also worth noting the computational training overhead. Classical and noise-additive strategies require approximately $0.13\,\mathrm{s}$ to fit on 12000 steps, whereas phase-alternating training requires about $0.35\,\mathrm{s}$ due to the fitting of both preliminary and final readout layers combined with the iterative state generation loop. Consequently, noise injection serves as a computationally lightweight optimization of the classical method, providing a 2.4% extension in t_V with virtually zero added training time. Meanwhile, phase-alternating training offers the maximum gains (3.8% in t_V and 4.8% in NRMSE) at a training overhead of 2.7 times.

Acknowledgment

This work was prepared as part of research project No. FSSW-2026-0008 “Development of fundamental principles for constructing digital twins of infrastructure systems in the data economy” using funds from a subsidy from the federal budget within the framework of the state assignment of the Ministry of Education and Science of Russia.

References

[1] F. Tao, H. Zhang, A. Liu, and A. Y. C. Nee, “Digital twin in industry: State-of-the-art,” *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 2405–2415, 2019.

[2] A. V. Andreev, L. Sh. Daraseliya, V. S. Dozhdev, A. V. Shenderyuk-Zhidkov, V. V. Shpak, and A. E. Hramov, “Digital twins: Synthesis of complexity theory and artificial intelligence,” *Izvestiya VUZ. Applied Nonlinear Dynamics*, vol. 34, no. 3, pp. 371–419, 2026.

[3] A. Rasheed, O. San, and T. Kvamsdal, “Digital twin: Values, challenges and enablers from a modeling perspective,” *IEEE Access*, vol. 8, pp. 21 980–22 012, 2020.

[4] A. V. Andreev, A. N. Pisarchik, N. Kulagin, R. Jaimes-Reátegui, G. Huerta-Cuellar, A. A. Badarin, and A. E. Hramov, “Stochastic cloning of dynamical systems with hidden variables,” *Physical Review E*, vol. 112, no. 1, p. 015303, 2025.

[5] H. Jaeger, “The “echo state” approach to analysing and training recurrent neural networks,” German National Research Center for Information Technology, Bonn, Germany, GMD Report 148, 2001.

[6] M. Lukoševičius and H. Jaeger, “Reservoir computing approaches to recurrent neural network training,” *Computer Science Review*, vol. 3, no. 3, pp. 127–149, 2009.

[7] H. Jaeger and H. Haas, “Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication,” *Science*, vol. 304, no. 5667, pp. 78–80, 2004.

[8] 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*, vol. 27, no. 12, p. 121102, 2017.

[9] J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott, “Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach,” *Physical Review Letters*, vol. 120, no. 2, p. 024102, 2018.

[10] M. Lukoševičius, “A practical guide to applying echo state networks,” in *Neural Networks: Tricks of the Trade*, 2nd ed., ser. Lecture Notes in Computer Science, G. Montavon, G. B. Orr, and K.-R. Müller, Eds. Berlin, Germany: Springer, 2012, vol. 7700, pp. 659–686.

[11] L.-W. Kong, Y. Weng, B. Glaz, M. Haile, and Y.-C. Lai, “Reservoir computing as digital twins for nonlinear dynamical systems,” *Chaos*, vol. 33, no. 3, p. 033111, 2023.

[12] A. Badarin, A. Andreev, V. Klinshov, V. Antipov, and A. E. Hramov, “Hidden data recovery using reservoir computing: Adaptive network model and experimental brain signals,” *Chaos*, vol. 34, no. 10, p. 103121, 2024.

[13] A. E. Hramov, A. V. Andreev, N. D. Kulagin, A. A. Badarin, S. A. Kurkin, H. Gao, L. Minati, and S. Boccaletti, “Clique-based topology enhances reservoir computing performance,” *Chaos*, vol. 36, no. 7, p. 073141, 2026.

[14] D. J. Gauthier, E. Bollt, A. Griffith, and W. A. S. Barbosa, “Next generation reservoir computing,” *Nature Communications*, vol. 12, p. 5564, 2021.

[15] A. Badarin, A. Andreev, and A. Hramov, “Hidden data recovery and forecasting via next-generation reservoir computing with multiscale delay selection,” *IEEE Transactions on Neural Networks and Learning Systems*, 2026, early access, doi: 10.1109/TNNLS.2026.3706608.

[16] A. Badarin, F. Ratnikov, and A. Andreev, “Reconstruction of EEG signals using next-generation reservoir computing,” *The European Physical Journal Special Topics*, 2026, online first, doi: 10.1140/epjs/s11734-026-02557-8.

Authorized licensed use limited to: Kant Baltic Federal Univ. Downloaded on October 23,2026 at 15:27:27 UTC from IEEE Xplore. Restrictions apply.

[17] A. Badarin, V. Klinshov, P. Smelov, and A. Andreev, "Reservoir computing reconstructs blood-oxygen-level-dependent signals: Whole-brain modeling study," *The European Physical Journal Special Topics*, vol. 234, pp. 4497–4504, 2025.
[18] Z. Li, A. Andreev, A. Hramov, O. Blyuss, and A. Zaikin, "Novel efficient reservoir computing methodologies for regular and irregular time series classification," *Nonlinear Dynamics*, vol. 113, no. 5, pp. 4045–4062, 2025.
[19] A. E. Hramov, N. Kulagin, A. V. Andreev, and A. N. Pisarchik, "Forecasting coherence resonance in a stochastic FitzHugh–Nagumo neuron model using reservoir computing," *Chaos, Solitons & Fractals*, vol. 178, p. 114354, 2024.
[20] A. E. Hramov, N. Kulagin, A. N. Pisarchik, and A. V. Andreev, "Strong and weak prediction of stochastic dynamics using reservoir computing," *Chaos*, vol. 35, no. 3, p. 033140, 2025.
[21] N. D. Kulagin, A. V. Andreev, and A. E. Hramov, "Analysis of the effect of a recurrent neural network size on modeling and prediction accuracy of a stochastic FitzHugh–Nagumo neuron," *Bulletin of the Russian Academy of Sciences: Physics*, vol. 89, no. 3, pp. 471–475, 2025.
[22] V. I. Ponomarenko, D. D. Kulminskiy, A. V. Andreev, and M. D. Prokhorov, "Assessment of an external periodic force amplitude using a small spike neuron network in a radiophysical experiment," *Technical Physics Letters*, vol. 47, no. 2, pp. 162–165, 2021.
[23] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Computation*, vol. 1, no. 2, pp. 270–280, 1989.
[24] E. N. Lorenz, "Deterministic nonperiodic flow," *Journal of the Atmospheric Sciences*, vol. 20, no. 2, pp. 130–141, 1963.
[25] J. C. Sprott, *Chaos and Time-Series Analysis*. Oxford, U.K.: Oxford University Press, 2003.
[26] A. Haluszczynski and C. Räth, "Good and bad predictions: Assessing and improving the replication of chaotic attractors by means of reservoir computing," *Chaos*, vol. 29, no. 10, p. 103143, 2019.
[27] A. Wikner, J. Harvey, M. Girvan, B. R. Hunt, A. Pomerance, T. Antonsen, and E. Ott, "Stabilizing machine learning prediction of dynamics: Novel noise-inspired regularization tested with reservoir computing," *Neural Networks*, vol. 170, pp. 94–110, 2024.
[28] N. Kulagin, A. Andreev, A. A. Koronovskii, O. I. Moskalenko, A. P. Sergeev, A. Badarin, and A. E. Hramov, "Intermittency in predicting the behavior of stochastic systems using reservoir computing," *Physical Review E*, vol. 111, no. 2, p. 024209, 2025.
[29] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 28, 2015, pp. 1171–1179.
[30] F. Köster, D. Patel, A. Wikner, L. Jaurigue, and K. Lüdge, "Data-informed reservoir computing for efficient time-series prediction," *Chaos*, vol. 33, no. 7, p. 073109, 2023.
[31] S. Tkach and A. Andreev, "State correction via data injection for a reservoir computing-based digital twin of the Lorenz system," *The European Physical Journal Special Topics*, 2026, online first, doi: 10.1140/epjs/s11734-026-02537-y.

Authorized licensed use limited to: Kant Baltic Federal Univ. Downloaded on October 23,2026 at 15:27:27 UTC from IEEE Xplore. Restrictions apply.