


Attention-enhanced reservoir computing

Felix Köster¹,* Kazutaka Kanno¹, Jun Ohkubo¹, and Atsushi Uchida¹†

Department of Information and Computer Sciences, Saitama University, 255 Shimo-Okubo, Sakura-ku, Saitama City, Saitama 338–8570, Japan

 (Received 27 December 2023; revised 1 May 2024; accepted 18 June 2024; published 16 July 2024)

Photonic reservoir computing has been successfully utilized in time-series prediction as the need for hardware implementations has increased. Prediction of chaotic time series remains a significant challenge, an area where the conventional reservoir computing framework encounters limitations of prediction accuracy. We introduce an attention mechanism to the reservoir computing model in the output stage. This attention layer is designed to prioritize distinct features and temporal sequences, thereby substantially enhancing the prediction accuracy. Our results show that a photonic reservoir computer enhanced with the attention mechanism exhibits improved prediction capabilities for smaller reservoirs. These advancements highlight the transformative possibilities of reservoir computing for practical applications where accurate prediction of chaotic time series is crucial.

DOI: [10.1103/PhysRevApplied.22.014039](https://doi.org/10.1103/PhysRevApplied.22.014039)

I. INTRODUCTION

The combination of photonics and machine learning has emerged as a promising field of information processing, utilizing the development of more rapid and efficient computational methodologies [1]. Central to this interdisciplinary area is the concept of photonic reservoir computing, a paradigm that utilizes optical capabilities to enhance reservoir computations [2–9]. A semiconductor laser with optical feedback is one example of a photonic reservoir, which is a prime example of the photonic delay-based reservoir computer model [4–8]. Numerous studies have underscored the efficacy of delay-based reservoir computing in tackling a variety of complex tasks, extending from bit error correction in optical communications [9] to prediction of spatiotemporal chaotic systems [10].

Chaotic time-series prediction is known to be challenging owing to its inherent sensitivity to accuracy and its pronounced nonlinearity [11]. A prime example of this challenge is weather forecasting, which is typically limited to short-term predictions due to the fundamentally chaotic nature of weather systems.

The framework of reservoir computing has been widely implemented using a variety of physical devices, such as electronic circuits [12], spintronic devices [13], soft robots [14], and nanostructure materials [15]. One of the advantages of reservoir computing is the easy implementation of the reservoir without high computational cost because input weights and network weights are randomly fixed, and

only the output weights are trained by a learning algorithm. Classical reservoir computing is often limited because of the inflexibility of the fixed weight structure when faced with the complexities inherent in such chaotic time series [16]. However, hardware technology advancements have opened pathways for accelerated prediction, giving access to real-time responses and improved computational efficiency [17]. This progress has enabled the integration of reservoir computing with other advanced machine learning methods, one example being deep learning and its most successful recent development, the attention mechanism [18].

To further enhance the predictive capabilities of reservoir computing, we propose integrating the attention mechanism within a reservoir computer. This mechanism has demonstrated superior effects in diverse areas of machine learning [18]. By incorporating an attention layer as the output layer, the system acquires the capacity to prioritize specific temporal characteristics, potentially yielding a more dynamical comprehension of the data [19].

This study introduces the attention mechanism to delay-based photonic reservoir computing using a semiconductor laser with optical feedback. We perform chaotic time-series prediction by deploying two benchmark tasks. The first is the unidirectionally coupled two-Lorenz systems, in which the reservoir computer is only exposed to the x , y , and z variables of the forced Lorenz system [20], while the driving system remains hidden, thus rendering the task a non-Markovian deterministic chaotic system with hidden states. The second is a time series consisting of alternating Lorenz and Rössler data sets [21], which we use to investigate how well the attention mechanism adapts to swiftly changing tasks.

*Contact author: felixk@mail.saitama-u.ac.jp

†Contact author: auchida@mail.saitama-u.ac.jp

II. ATTENTION-ENHANCED RESERVOIR COMPUTING

A. Reservoir computing

Reservoir computing (RC) is a paradigm in the field of recurrent neural networks that offers an efficient approach to tasks involving temporal dynamics and sequential patterns [17]. The delay-based RC is a prime example of RC, distinctive for its utilization of time-multiplexed masking for input representation [12].

Let us assume we have a data set $\mathbf{X} = \{(x_1, y_1), \dots, (x_l, y_l), \dots, (x_L, y_L)\}$ where we want to is y_l through x_l , where l is the indexing and L the number of data points in the set. We multiply every data point x_l with a periodic mask $m(t)$, which is essentially a high-dimensional random vector distributed over time. This mask expands and distributes the data point x_l to enrich the temporal response dynamics of the reservoir system. We, therefore, have the masked input $v_l(t)$, formally represented as $v_l(t) = x_l \cdot m(t)$, where $m(t)$ is the masking vector of length T . In this study, we use a mask $m(t)$ that is constant for an interval, where the values of the mask m_n are drawn randomly from a uniform distribution on the interval between 0 and 1. N is the number of reservoir node states within T , i.e., $T = N\theta$. After applying $v_l(t)$ to the reservoir, we sample the system's response at distances of θ yielding a high-dimensional representation, allowing for rich and diverse dynamics to be captured for downstream tasks.

As a reservoir, we use a delay-based photonic reservoir computer, which captures the dynamics of semiconductor lasers with weak optical feedback. The Lang-Kobayashi equations mathematically represent the systems as follows [4,5,22]:

$$\dot{E}(t) = (1 + i\tilde{\alpha})E(t) - |E(t)|^2 E(t) + \kappa E(t - \tau) \exp(i\phi), \quad (1)$$

$$\dot{n}(t) = p + \eta v_l(t) - (1 + |E(t)|^2)n(t), \quad (2)$$

where $E(t)$ denotes the complex electric field amplitude, $n(t)$ represents the carrier density, $\tilde{\alpha}$ is the linewidth enhancement factor, κ signifies the feedback strength, τ is the feedback delay time, ϕ is the phase shift, and p is the normalized injection current. The parameter values are summarized in Table I.

Within the context of delay-based RC, the input modulates the injection current p , thus affecting the system's dynamics. The modulated injection current is denoted as $\eta v_l(t)$, where η is the input strength and $v_l(t)$ is the input to the reservoir system. The reservoir responses $r(t)$ are derived from the intensity $r(t) = I(t) = |E(t)|^2$ of the electric field amplitude at discrete time points of distance θ , i.e., $r(\theta), r(2\theta), \dots, r(N\theta)$. To yield the response of one input data point x_l , we have

$$\mathbf{r}_l = (r(IT + \theta), r(IT + 2\theta) \dots, r(IT + N\theta))^T, \quad (3)$$

TABLE I. Summary of parameters and their numerical values.

Parameter	Numerical value
Mask period T	$N\theta = 5 \times 10^{-9}$ s
Linewidth enhancement factor $\tilde{\alpha}$	3
Phase shift ϕ	0.0
Feedback strength κ	10^8
Feedback delay time τ	$1.01T = 5.05 \times 10^{-9}$ s
Normalized injection current p	1.11
Input strength η	0.08
Number of nodes N	50
Interval length θ	10^{-10} s

with T denoting the transpose of the vector \mathbf{r}_l . Collecting all the responses for all the data points x_l , we arrive at the response matrix $\mathbf{R} \in \mathbb{R}^{L \times N}$ with

$$\mathbf{R} = \begin{pmatrix} \mathbf{r}_1^T \\ \vdots \\ \mathbf{r}_L^T \end{pmatrix}. \quad (4)$$

In the classic RC framework, only the readout weights are trained while the internal weights of the reservoir are left unchanged [16]. The output or readout weights are crucial in mapping the reservoir states to the desired outputs. Denoted typically as $\mathbf{w}_{\text{reg}} \in \mathbb{R}^{N \times 1}$, these weights are applied to the reservoir states \mathbf{R} to approximate the target $\mathbf{y} \in \mathbb{R}^{L \times 1}$. Training the output weights involves optimizing to minimize the distance between the approximation vector $\mathbf{d} = \mathbf{R}\mathbf{w}_{\text{reg}} \in \mathbb{R}^{L \times 1}$ and the true target vector \mathbf{y} . Commonly, a linear regression approach is employed [16], considering a regularization term to prevent overfitting. The training process can be mathematically represented as

$$\mathbf{w}_{\text{reg}} = \arg \min_{\mathbf{w}} (||\mathbf{y} - \mathbf{R}\mathbf{w}||^2 + \lambda ||\mathbf{w}||^2), \quad (5)$$

where \mathbf{R} denotes the collection of reservoir states, \mathbf{y} represents the targets, and λ is the regularization coefficient.

The analytical solution to the ridge-regression problem can be derived by setting the gradient of Eq. (5) with respect to \mathbf{w} to zero. The resulting equation can be expressed as

$$\mathbf{w}_{\text{reg}} = (\mathbf{R}^T \mathbf{R} + \lambda \mathbf{I})^{-1} \mathbf{R}^T \mathbf{y}, \quad (6)$$

where $\mathbf{I} \in \mathbb{R}^{N \times N}$ denotes the identity matrix. This solution provides a direct way to compute the optimal \mathbf{w}_{reg} without iterative optimization. The addition of the regularization term $\lambda \mathbf{I}$ ensures that the matrix $\mathbf{R}^T \mathbf{R} + \lambda \mathbf{I}$ is invertible, thus guaranteeing a unique solution and penalizing overfitting.

B. Attention mechanism

In deep learning, particularly when dealing with sequential data, the attention mechanism has emerged as a

groundbreaking concept [18]. This mechanism enables models to selectively focus on relevant portions of the input sequence during output generation, mirroring how humans selectively prioritize information during decision making or data processing. Central to this mechanism is the calculation of attention weights, which subsequently guide a weighted combination of input values. These weights effectively signify the level of “attention” or significance each part of the input should receive.

To clarify the method of attention, we present an example inspired by Ref. [23]. Imagine a database containing daily temperature records over several years, illustrated as tuples such as { (“2024-01-01,” “−2 °C”), (“2024-01-02,” “0 °C”), (“2024-01-03,” “3 °C”)}, where the dates act as keys (k) and temperatures as values (v). Querying this database (q) with the date “2024-01-02,” would fetch the exact temperature “0 °C.” When an exact match for the queried date is absent and approximate matching is allowed, the system might return approximate temperatures close to neighboring dates. In this example, the query (q) asks for the attention considering all values in the sequence (here being three temperatures at different time steps). The keys (k) enable to focus on specific values (v) of the input sequence by comparing the query (q) to all the keys (k) of the input sequence.

Expanding on this illustration, we compute a distance function between the query (q) and the keys (k) within the entire input sequence $\mathbf{U} = \{(k_1, v_1), \dots, (k_m, v_m)\}$, providing a weighting (or attention) to the associated values (v). This relationship is captured as

$$\text{attention}(\mathbf{q}, \mathbf{K}) = \sum_{i=1}^m \alpha(\mathbf{q}, \mathbf{k}_i) v_i, \quad (7)$$

where $\text{attention}(\mathbf{q}, \mathbf{K})$ is the weighted sum of all values in the input sequence for a given query (\mathbf{q}), i indexes the input sequence \mathbf{U} , and $\alpha(\mathbf{q}, \mathbf{k}_i)$ denotes the scalar attention weights. An interesting aspect is the determination of the function α . Several functions can be employed, from traditional distances like the Boxcar function, defined as $\alpha_{\text{Boxcar}} = 1$ if $\|\mathbf{q} - \mathbf{k}_i\| \leq 1$, to a Gaussian similarity, given by $\alpha_{\text{Gaussian}} = \exp(-\|\mathbf{q} - \mathbf{k}_i\|_2^2)$ [23]. In contemporary deep learning, the function α is often approximated using neural networks, allowing gradient descent to tailor the function optimally to the task itself.

A more recent progression in attention mechanisms introduces the concept of self-attention [18,24]. In this framework, the queries (\mathbf{q}), keys (\mathbf{k}), and values (\mathbf{v}) originate from the same data point as shown in Fig. 1(a). For each individual data point in the input sequence (e.g., the temperature at a specific time step), corresponding queries (\mathbf{q}), keys (\mathbf{k}), and values (\mathbf{v}) are produced. Typically, these elements are generated through matrix transformations, where the matrix weights are refined and optimized using a gradient-descent algorithm.

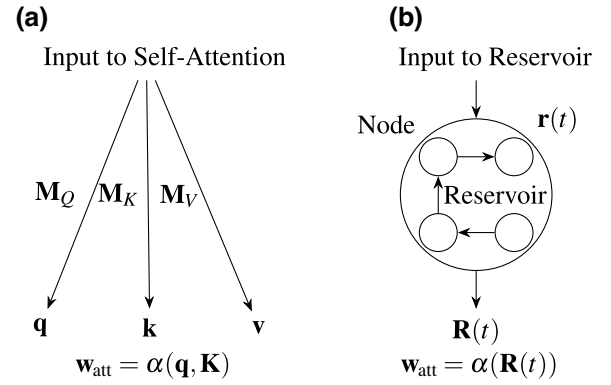


FIG. 1. (a) Self-attention mechanism creating queries (\mathbf{q}), keys (\mathbf{k}), and values (\mathbf{v}) from an input. (b) Reservoir approach transforming an input into a high-dimensional reservoir state $\mathbf{R}(t)$. Both projections are then used in an abstract function α to yield the attention weights \mathbf{w}_{att} .

C. Incorporation of attention layer in reservoir computing

Next, we delve into incorporating the attention mechanism within a reservoir computer. While numerous potential locations for integrating the attention mechanism exist, we focus on its application to improve the reservoir computer’s output layer. Therefore, we replace the traditional weights \mathbf{w}_{reg} with the attention weights \mathbf{w}_{att} .

In our attention-enhanced reservoir computer, the reservoir states $r(t)$ simultaneously serve as the queries, keys, and values, as shown in Fig. 1(b). This is reminiscent of the self-attention scheme where every data point of the input sequence is projected onto queries, keys, and values, often via matrices learned through a gradient-descent algorithm [18,24]. In the case of reservoir computing, the reservoir projects the input into a high-dimensional feature space, similar to the matrix transformation that projects the input data point to the queries, keys, and values.

The concept of attention-enhanced reservoir computing is depicted in Fig. 2. We consider one-dimensional target signals. In the self-attention mechanism, the query of one data point in the sequence is combined with the keys of all other data points in the sequence to generate the attention weights, which are then weighted with the values (see Ref. [18,24]). In the attention-enhanced scheme, the role of the reservoir node states $\mathbf{r}_l \in \mathbb{R}^{N \times 1}$ is equal to the queries, keys, and values of the self-attention approach, where N is the number of reservoir nodes and $l \in [1, L]$ is the l -th data point applied to the reservoir. To predict the l th data point, we concisely write the attention-enhanced reservoir computer via

$$\mathbf{w}_{\text{att},l} = \mathbf{W}_{\text{net}} \mathbf{r}_l, \quad (8)$$

$$d_l = \mathbf{w}_{\text{att},l}^T \mathbf{r}_l, \quad (9)$$

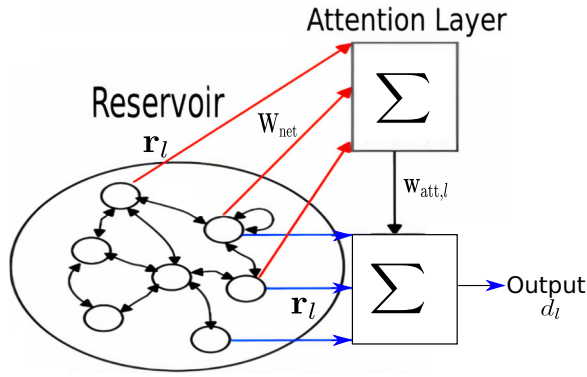


FIG. 2. Schematic of the attention-enhanced reservoir computing setup. The trainable parameters \mathbf{W}_{net} take the reservoir node states and yield the attention weights \mathbf{w}_{att} , which are then weighted with the reservoir node states to generate an output.

where d_l represents the approximation value for a target signal y_l , $\mathbf{w}_{\text{att},l} \in \mathbb{R}^{N \times 1}$ are the input-dependent attention weights, and $\mathbf{W}_{\text{net}} \in \mathbb{R}^{N \times N}$ are the attention-layer weights that are trained via a gradient-descent algorithm. Here, Eq. (8) represents the attention layer, taking the reservoir node state \mathbf{r}_l as input and yielding the attention weights $\mathbf{w}_{\text{att},l}$. Equation (9) represents the output layer of the reservoir computer, computing the weighted sum of the reservoir nodes to approximate the target y_l , in which the classic ridge-regression weights are now substituted by the attention weights \mathbf{w}_{att} . A detailed description of a multidimensional target signal is given in Appendix A.

We point out that the scheme shown in Eqs. (8) and (9) is linear and a substitution of Eq. (8) into Eq. (9) is possible. However, we still keep the equations separated into Eqs. (8) and (9) as it depicts the general architecture of the attention-enhanced approach. For example, we show a result of a nonlinear attention approach in Appendix B.

D. Procedure of attention-enhanced reservoir computing

A detailed visualization of our complete scheme's data stream is shown in Fig. 3. We describe this representation by providing a step-by-step explanation of the underlying operations as follows.

1. Reservoir processing

The standardized data, x_l , is passed into the reservoir, resulting in a reservoir state. In our setup, x_l is processed in Eq. (2) via the injection current of the semiconductor laser. This reservoir state is an intermediate, high-dimensional representation of the input data. To standardize the data, we modify the training data set to have the mean of zero and standard deviation of one for training and testing.

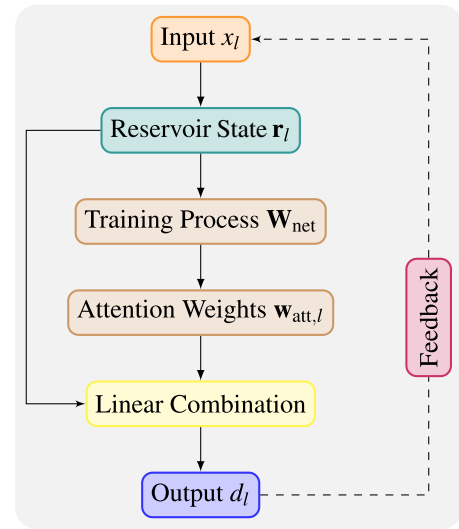


FIG. 3. Schematic representation of the reservoir computing process augmented with an attention mechanism.

2. Training process

The standardized reservoir state, \mathbf{r}_l , undergoes processing through a linear transformation with weights $\mathbf{W}_{\text{net}} \in \mathbb{R}^{N \times N}$. Regarding the training of the weights \mathbf{W}_{net} , deep-learning methods can be applied [25] using a gradient-descent algorithm:

$$\mathbf{W}_{\text{net}}(s+1) = \mathbf{W}_{\text{net}}(s) - \gamma \nabla F(\mathbf{W}_{\text{net}}(s), \mathbf{R}), \quad (10)$$

with γ being a learning rate, which we set to 0.01, and s iterates through the gradient-descent epochs, and the optimization objective is

$$\mathbf{W}_{\text{net}} = \min_{\mathbf{W}} (F(\mathbf{W}, \mathbf{R}, \mathbf{y})), \quad (11)$$

where \mathbf{R} is the set of all reservoir states and \mathbf{y} is the target dataset. We deploy F as a simple squared distance, i.e., $F(\mathbf{W}, \mathbf{R}, \mathbf{y}) = \frac{1}{2} \|d(\mathbf{R}, \mathbf{W}) - \mathbf{y}\|_2^2$, with $d(\mathbf{R}, \mathbf{W})$ being the approximation to the target dataset \mathbf{y} . To clarify, the loss $F(\mathbf{W}, \mathbf{R}, \mathbf{y})$ (in our case the NRMSE) of the full training data set of all K data points is calculated for every epoch and the weights \mathbf{W}_{net} are optimized via the gradient descent. To speed up the process, first we calculate the reservoir states for all K data points and save them. After that we use the reservoir states to calculate $F(\mathbf{W}, \mathbf{R}, \mathbf{y})$ and apply the gradient descent.

The attention-enhanced reservoir thus loses its advantage of utilizing the analytically solvable ridge regression as an optimization approach. Owing to the recent improvement in hardware-accelerated optimization of gradient-descent approaches, we believe that the limiting computation factor of reservoir computing is not the output-layer optimization. Therefore, augmenting reservoir computing with the attention mechanism could couple

the best of two methods. The loss over the epochs of the gradient descent is depicted in Appendix C.

3. Attention weights

The attention weights, $\mathbf{w}_{\text{att},l}$, emerge from the linear transformation \mathbf{W}_{net} . These weights dictate how much significance is placed on each node or neuron in the reservoir at a specific time. In essence, the attention mechanism dynamically adjusts, prioritizing certain nodes over others based on their relevance to the output at that particular time step. The attention weights $\mathbf{w}_{\text{att},l}$ are computed via a linear transformation of the standardized reservoir state \mathbf{r}_l via the weights \mathbf{W}_{net} , represented as

$$\mathbf{w}_{\text{att},l} = \mathbf{W}_{\text{net}} \mathbf{r}_l. \quad (12)$$

It is worth noting that the weights \mathbf{W}_{net} are static after training. The time dependence of the attention weights $\mathbf{w}_{\text{att},l}$ arise from the fact that the reservoir node states are time dependent. Therefore, a full photonic based implementation of the attention-enhanced reservoir computer could be achieved.

4. Linear combination

The attention weights and the standardized reservoir states are then linearly combined. This combination effectively amplifies the states that the attention mechanism deems significant. The corresponding output d_l is given by

$$d_l = \mathbf{w}_{\text{att},l}^T \mathbf{r}_l. \quad (13)$$

Our integration of the attention layer in reservoir computing enables the system to prioritize features in the input data, particularly in complex time-series prediction scenarios. The attention-enhanced reservoir computer offers a more nuanced and accurate representation of the data by dynamically shifting focus to the most relevant nodes in the reservoir based on context.

E. Prediction quality metrics

In the context of reservoir computing for time-series prediction, two primary configurations are commonly used: the open loop and closed loop [26].

1. Normalized root-mean-square error

In the open-loop configuration, the model performs a one-step-ahead predictions. We evaluate the model's prediction accuracy via the normalized root-mean-square

error (NRMSE) [27] as

$$E_{\text{NRMSE}} = \sqrt{\frac{\sum_{l=1}^L \|d_l - y_l\|^2}{N \text{Var}(Y)}}, \quad (14)$$

where y_l is the l th target data point, d_l is the prediction data, l is the index up to the L th data point, and $\text{Var}(Y)$ is the variance over the whole data set.

2. Valid prediction time

Figure 3 illustrates the closed-loop feedback configuration. In this setup, after the model has been trained, it is switched to a closed-loop mode, where the model's own predictions are recursively fed back as input. This method allows for continuous predictions, extending beyond the one-step-ahead prediction. The quality of predictions in a closed-loop configuration is evaluated by a metric called the valid prediction time (VPT) [28], defined as

$$T_{\text{VPT}} = \delta_y(t) > 0.4, \quad \delta_u(t) = \frac{|y(t) - d(t)|^2}{\langle |y(t) - \langle y(t) \rangle|^2 \rangle}, \quad (15)$$

where VPT measures the first time at which $\delta_y(t)$ surpasses the value of 0.4, in which $\delta_y(t)$ measures the average normalized distance over the trajectory between the target $y(t)$ and the reservoirs output $d(t)$, and $\langle \rangle$ denotes the time average. VPT measures the duration for which the model's predictions remain within an acceptable error threshold, thereby indicating the effectiveness of the model in maintaining reliable predictions over time. Unlike open-loop predictions, closed-loop predictions compound potential errors over time, because of the sensitive dependence on initial conditions on chaotic dynamic systems.

III. NUMERICAL RESULTS

This section presents the comparative results of the attention-enhanced photonic reservoir computing to the conventional linear regression technique. The evaluation is performed for a unidirectionally coupled two-Lorenz system introduced in Sec. III A and an alternating Lorenz-Rössler system introduced in Sec. III B.

A. Unidirectionally coupled two-Lorenz system (UCTLS)

1. Model

As a benchmark task, we utilize a unidirectionally-coupled two-Lorenz system [20,29], defined by the following equations:

$$\dot{x}_1 = (a + \sigma_{\text{force}} x_2)(y_1 - x_1), \quad (16)$$

$$\dot{y}_1 = x_1(b + \sigma_{\text{force}} y_2 - z_1) - y_1, \quad (17)$$

$$\dot{z}_1 = x_1 y_1 - (c + \sigma_{\text{force}} z_2) z_1, \quad (18)$$

$$\dot{x}_2 = a(y_2 - x_2), \quad (19)$$

$$\dot{y}_2 = x_2(b - z_2) - y_2, \quad (20)$$

$$\dot{z}_2 = x_2 y_2 - c z_2. \quad (21)$$

These equations represent two Lorenz systems, with the first one being driven by the second. The parameters, namely $a = 10$, $b = 28$, and $c = \frac{8}{3}$, are used as the standard Lorenz parameters [20]. The key distinction is the introduction of a forcing term, σ_{force} , that couples the two systems unidirectionally.

For the reservoir computing task, only the x_1 , y_1 , and z_1 variables are exposed, while x_2 , y_2 , and z_2 are hidden. Predicting such a non-Markovian deterministic chaotic time series is challenging [30], especially when it has underlying hidden states that are not directly observable by the reservoir computer. The system was integrated with a Runge-Kutta 45 approach with a step size of $dt = 0.01$, while the sampling time is $dt_{\text{sample}} = 0.1$. Regarding the largest Lyapunov exponent of the system, our simulations indicate that no significant variation up to a force of $\sigma_{\text{force}} = 0.15$ is exhibited relative to the classic Lorenz value of $\lambda_{\text{Lorenz}} \approx 0.91$. For all simulations we use 25 000 training and 5000 testing data points.

We begin with the UCTLS as defined by Eqs. (16)–(21) subjected to a perturbation strength of $\sigma_{\text{force}} = 0.05$, implemented in a photonic reservoir utilizing 50 nodes, with other parameters specified in Table I.

2. Results of time series and weights

Figure 4(a) illustrates an open-loop time series. The upper graph displays the time series of the x_1 variable from the exposed Lorenz attractor of the true system (in blue) alongside the predicted time series (black line), plotted against the time in Lyapunov times with $\lambda_{\text{Lorenz}} \approx 0.91$. The similarity between the prediction and the actual system's behavior is accurate.

A more interesting insight comes from Fig. 4(b), which shows the computed absolute value of the attention weights w_{att} for each node within the reservoir across the same period. Nodes with higher weight magnitudes are assumed to be more relevant for predictions at specific time points. Vertical dashed lines indicate a pattern change corresponding to moments where the system's trajectory shifts from one side of the attractor to the other. A corresponding shift in the attention weights' structure implies that the attention-enhanced reservoir can adapt dynamically to the time-dependent characteristics of the task. This feature is essential, as the UCTLS demands time-dependent transformations depending on the current position within the attractor space.

The attention mechanism provides a dynamic adjustment unlike the static nature of weights employed in linear

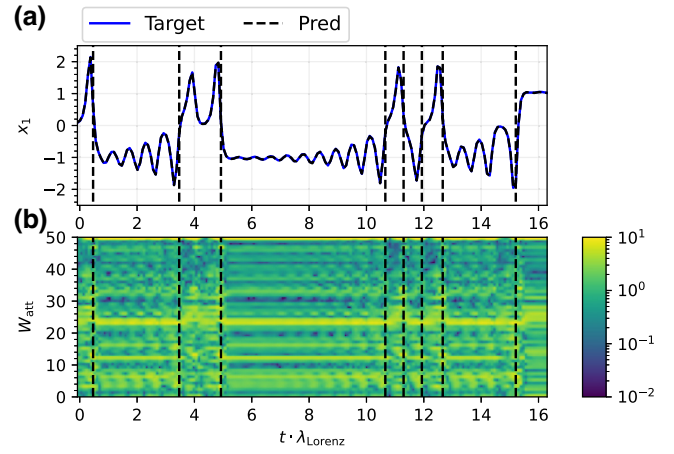


FIG. 4. Example time series of open-loop configuration for the UCTLS. (a) shows the x variable of the time series, and (b) the attention weights in the same time interval for a 50-node reservoir. The color bar shows the magnitude of the attention weights. The dashed black lines indicate a shift from the attractor, pinpointing the dynamical adjustment of the weights to the task.

regression, which cannot distinguish this characteristic. Given that nearly all complex systems evolve with time, adaptively adjusting the significance of different nodes is not just beneficial but necessary. This strongly illustrates the inherent limitations of static approaches when dealing with systems characterized by dynamic evolution.

This behavior continues even if the closed-loop configuration is applied, as shown in Fig. 5. The reservoir's prediction and the true trajectory stay close to each other until around five Lyapunov times, indicated by a vertical dashed black line pinpointing to the time at which the VPT (see Sec. II E) is computed. The weights in the lower graph show the distinctive feature of adjusting to the dynamical problem itself despite being configured in the closed-loop scheme.

3. Results of quality metrics

To showcase the efficacy of dynamic adjustment, we simulated an ensemble prediction across ten distinct trajectories for varying reservoir sizes. The results are shown in Fig. 6. We computed the NRMSE for the open-loop depicted in Fig. 6(a) and the VPT for the closed-loop illustrated in Fig. 6(b). These metrics were calculated for both the classical linear regression model (dashed orange line) and the attention-enhanced model (solid blue line) across a spectrum of reservoir sizes, from 10 to 150 nodes. In all cases, the same reservoir with the exact same reservoir states were used for both the classical approach and the attention-enhanced approach.

The results indicate that the attention-enhanced model yields a lower NRMSE for smaller-sized reservoirs while simultaneously achieving a higher VPT. Although this

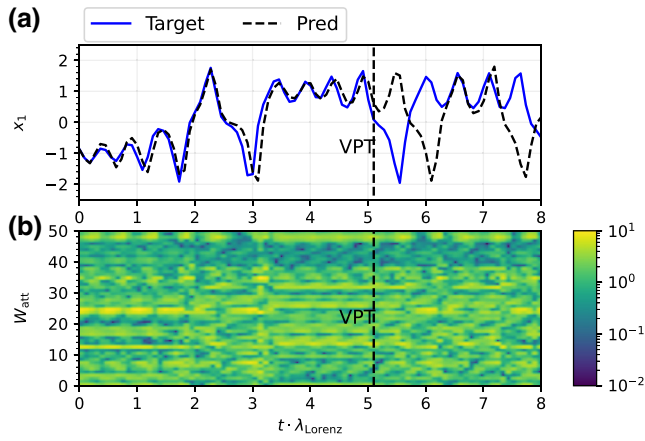


FIG. 5. Example time series of closed-loop configuration for the UCTLS. (a) shows the x variable of the time series, and (b) the attention weights in the same time interval for a 50-node reservoir. The true trajectory and closed-loop prediction are depicted by the solid blue and dashed black lines, respectively. The color bar shows the magnitude of the attention weights. The vertical dashed black line indicates the time point for VPT computation.

advantage diminishes as the reservoir size increases, it stays consistent. Notably, the optimal performance of the attention-enhanced reservoir is observed with as few as 20 nodes. This suggests that the attention mechanism significantly reduces the required size of the reservoir. However, this size reduction comes at the costs of additional weights

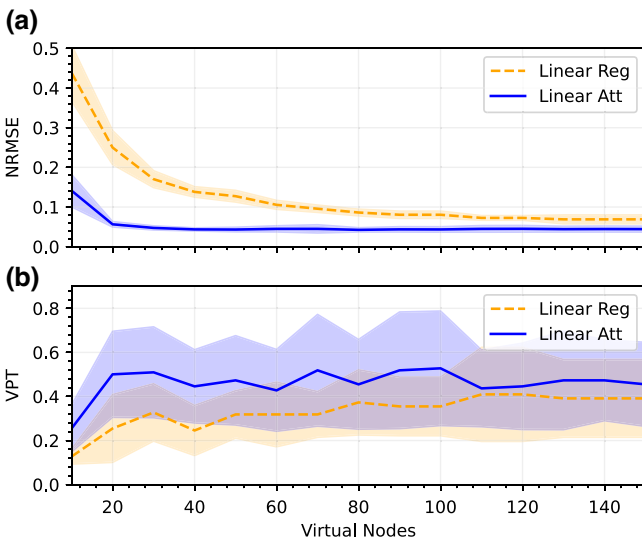


FIG. 6. (a) NRMSE and (b) VPT for classical linear regression (dashed orange line) and attention-enhanced reservoir computing (solid blue line) with varying reservoir size N for UCTLS. The shaded areas depict the standard deviations of NRMSE and VPT over ten time series.

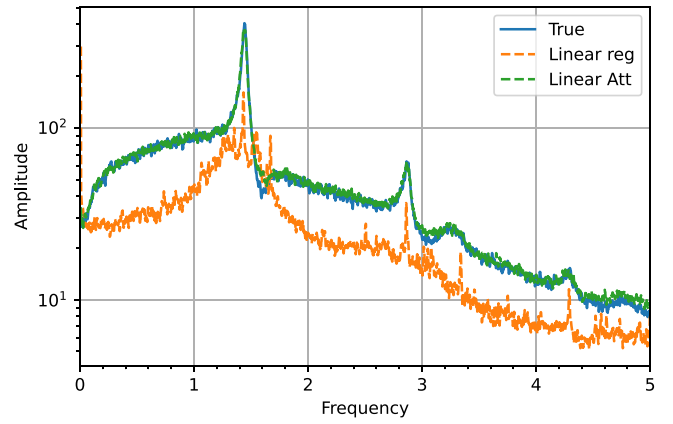


FIG. 7. Fourier spectrum of a predicted time series in the closed-loop configuration with a reservoir size of $N = 30$. The true system is depicted as a solid blue line, while the classic linear regression and the attention-enhanced reservoir are shown as dashed orange and green lines, respectively.

in the output layer. Nevertheless, this approach is particularly interesting when only limited reservoir sizes are realistically accessible.

We analyze the average power spectrum of a time series obtained with a 30-node reservoir, reproducing the UCTLS task with $\sigma_{\text{force}} = 0.05$. Figure 7 displays the resulting Fourier spectra: the true system's spectrum is depicted by the blue curve, the orange curve represents the closed-loop outcome of the classical linear regression approach, while the green curve corresponds to the linear attention approach. An examination of Fig. 7 reveals that the linear regression approach significantly deviates in reconstructing the spectrum of the original attractor, failing to identify the correct main frequency and the secondary peak at a frequency of 2.9 Hz. In contrast, the linear attention method successfully captures both frequencies, reproducing the power spectrum very accurately.

To deepen our analysis of the UCTLS, we evaluate the dependency of the NRMSE and VPT on the driving force σ_{force} . We keep the reservoir size constant with $N = 50$ nodes, while the other parameters are unchanged. The results are illustrated in Fig. 8, using the same color coding and line style as in Fig. 6. The behavior of the classic single Lorenz system is represented on the left side of the graph with a driving force of $\sigma_{\text{force}} = 0$. The linear attention method outperforms the classical linear regression approach for the VPT and NRMSE metrics in this regime. When the forcing strength is increased, the observed improvement for VPT diminishes, while the NRMSE gradually increases for both methods. However, the attention-enhanced approach yields lower NRMSE values throughout. These results underline the improved prediction performance of the attention-enhanced reservoir relative to the traditional approach.

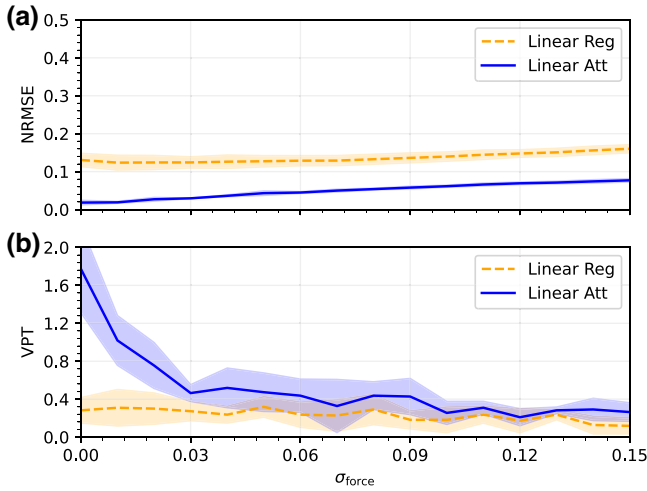


FIG. 8. (a) NRMSE and (b) VPT for classical linear regression (dashed orange line) and attention-enhanced reservoir computing (solid blue line) with varying coupling strength σ_{force} of UCTLS. The shaded areas depict the standard deviations of NRMSE and VPT over ten time series. The photonic reservoir size is set to $N = 50$ nodes.

B. Alternating the Lorenz and Rössler system (ALRS)

1. Model

The second benchmark is a target dataset constructed using two canonical chaotic systems: the Rössler and Lorenz models [21]. For the Rössler model, the equations of motion are given by

$$\frac{dx_R}{dt} = -y_R - z_R, \quad (22)$$

$$\frac{dy_R}{dt} = x_R + a_R y_R, \quad (23)$$

$$\frac{dz_R}{dt} = b_R + x_R z_R - c_R z_R. \quad (24)$$

For the Lorenz model, the dynamics are described by

$$\frac{dx_L}{dt} = a_L(y_L - x_L), \quad (25)$$

$$\frac{dy_L}{dt} = -x_L z_L + b_L x_L - y_L, \quad (26)$$

$$\frac{dz_L}{dt} = x_L y_L - c_L z_L. \quad (27)$$

The parameters used in our simulations are $a_R = 0.2$, $b_R = 0.2$, $c_R = -5.7$ for the Rössler system and $a_L = 10$, $b_L = 28$, $c_L = -\frac{8}{3}$ for the Lorenz system. Both of these systems are known for generating deterministic chaos. For data generation, we exclusively consider the x -variable trajectories (x_R for Rössler and x_L for Lorenz) if the open-loop configuration is applied, while for the closed-loop configuration the x , y , z variables are exposed. Therefore, a

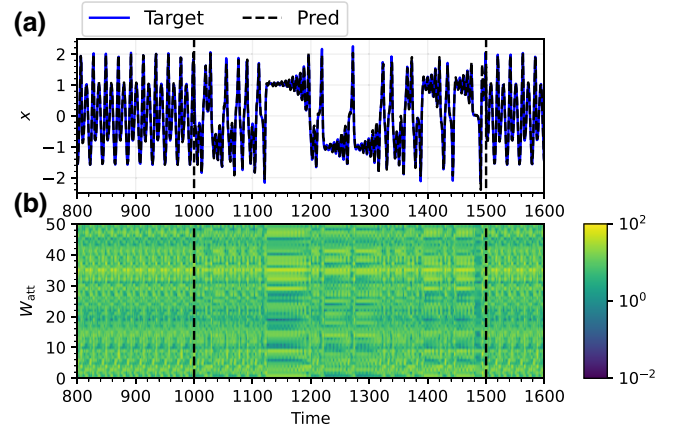


FIG. 9. Example time series of open-loop configuration for ALRS. (a) shows the x variable of the time series, and (b) the attention weights in the same time interval. The color bar shows the magnitude of the attention weights. The dashed black lines indicate the change in task from Rössler to Lorenz or vice versa, pinpointing to the dynamical adjustment of the weights to the task.

multidimensional target is used for attention and enhanced reservoir computing, as described in Appendix A.

The Rössler system is integrated with a time step of $dt = 0.05$, while every ten points are downsampled in an effective time step of $dt_{\text{sample}} = 0.5$. Similarly, the Lorenz system is integrated using a time step of $dt = 0.01$ and subsequently downsampled to achieve an effective time step of $dt_{\text{sample}} = 0.1$. For each system, multiple 500-step trajectories are generated. These trajectories are then alternately concatenated, yielding a time series that alternates between Rössler and Lorenz every 500 points for 25 000 training data points and 5000 testing data points [21]. Before concatenation, the time-series data from both models are standardized to ensure zero mean and unit standard deviation. We call this benchmark the ALRS. For both systems, the largest Lyapunov exponents are calculated with $\lambda_{\text{Lorenz}} \approx 0.91$ and $\lambda_{\text{Rössler}} \approx 0.071$.

2. Results

We investigate the ALRS, defined by Eqs. (22) to (27) in the same manner. An exemplary time series of this system in the open-loop configuration is depicted in Fig. 9. Figure 9(a) indicates that the prediction closely approximates the true system, suggesting a high degree of accuracy in predicting the next state.

Figure 9(b) provides the interesting part of the attention-enhanced reservoir. The vertical dashed lines denote the transitions between the Lorenz and Rössler dynamics within the task. A shift in the attention weights in response to these changes can be observed, implying that the attention-enhanced reservoir can adapt to different dynamical problems presented by the input signal.

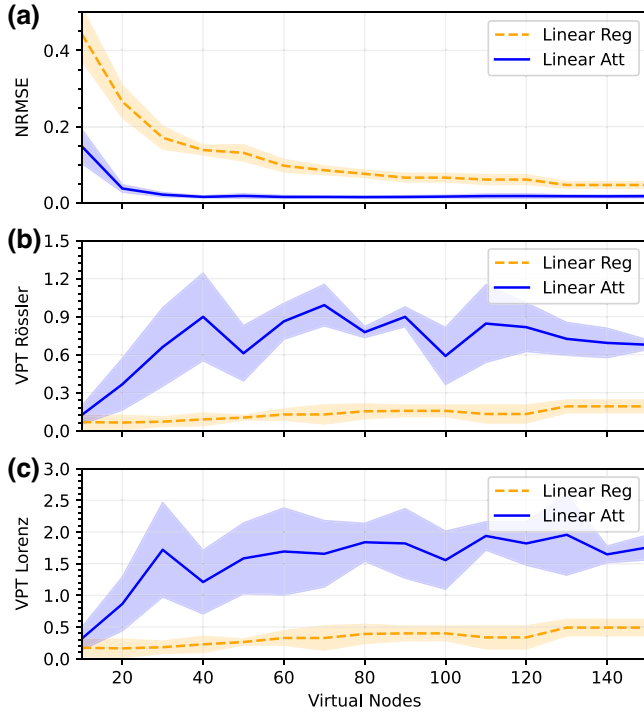


FIG. 10. (a) NRMSE, (b) VPT for Rössler inputs, and (c) VPT for Lorenz inputs for classical linear regression (dashed orange line) and attention-enhanced reservoir computing (solid blue line) with varying coupling strength reservoir sizes N . The shaded areas depict the standard deviations of NRMSE and VPT over ten time series.

To quantitatively analyze the prediction quality, we computed the NRMSE for the open loop and the VPT for the close loop in ALRS across reservoir sizes, spanning from 10 to 150 nodes, as shown in Fig. 10. Note that in the ALRS benchmark, two different VPTs are computed by starting the closed-loop configuration either in a window of Rössler inputs or in a window of Lorenz inputs. Here, the NRMSE for the classical linear regression method is presented by the dashed orange line, while the solid blue line represents the attention approach. These findings agree with previous observations: the attention-enhanced reservoir yields smaller NRMSE (higher VPT), with the effect being particularly pronounced for smaller reservoir sizes. Interestingly, this dynamic adjustment can tackle a problem that swiftly changes between two distinct attractors.

Finally, we analyze the NRMSE for predicting more than one step ahead for the UCTLS and ALRS in the open-loop configuration [8]. The results for UCTLS are shown in Fig. 11(a) and for ALRS in Fig. 11(b), depicting the NRMSE over multisteps for future predictions. The UCTLS system is simulated with a driving force of $\sigma_{\text{force}} = 0.05$. Both reservoirs for the UCTLS and ALRS tasks utilize 50 nodes. The dashed orange line shows the classical ridge-regression approach while the solid blue

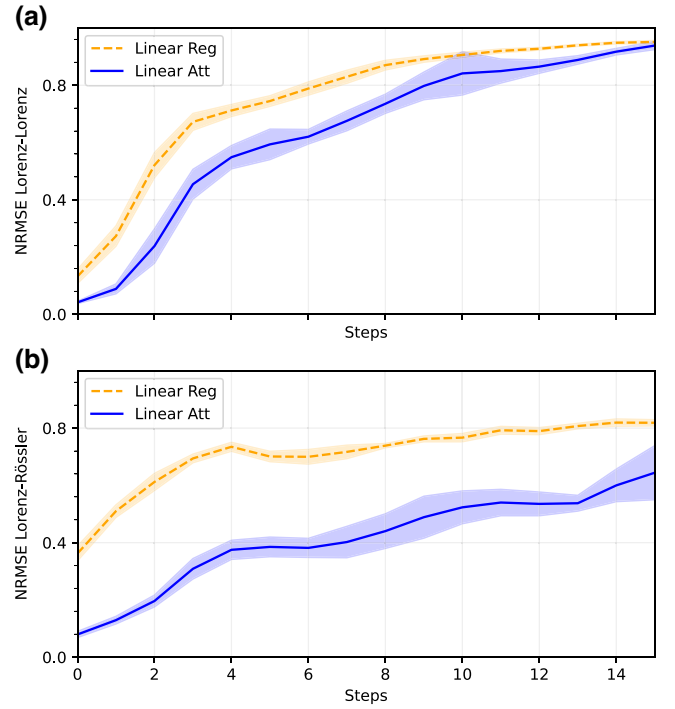


FIG. 11. NRMSE for (a) UCTLS and (b) ALRS for classical linear regression (dashed orange line) and attention-enhanced reservoir computing (solid blue line) with varying the number of steps for future prediction. The shaded areas depict the standard deviations of NRMSE and VPT over ten time series. The photonic reservoir size is set to $N = 50$ nodes.

line depicts the attention-enhanced scheme. The improvement for the UCTLS case is clear for all prediction steps into the future, decreasing the NRMSE by a small amount. More impressively, the NRMSE decreases for the ALRS in Fig. 11(b), where a significant reduction in the NRMSE is clearly observed using the attention-enhanced scheme.

IV. CONCLUSION

We introduced an attention mechanism to reservoir computing to demonstrate a notable improvement for predicting complex dynamical systems. Across our simulations of the UCTLS and ALRS, the attention-enhanced reservoir consistently outperformed classical linear regression methods. It has shown particular advantages in scenarios with smaller reservoir sizes, indicating its potential to reduce system requirements without sacrificing performance. This result is particularly advantageous for integrated photonic reservoirs, where a small reservoir can be easily implemented. It also highlights the possibility of a full photonic implementation, as a photonic accelerator.

This study shows the importance of dynamically adjustable weights, characteristics of the attention mechanism, and their effect in predicting the complex behaviors

of nonlinear dynamic systems. Notably, the attention-enhanced reservoir exhibits swift adaptability to sudden changes in dynamic input, a quality that promises significant improvements in real-time applications. By achieving lower prediction errors and capturing key spectral features more accurately, the attention-enhanced approach provides an efficient method for integrating attention within various reservoir computing schemes.

ACKNOWLEDGMENTS

This study was partly supported by JSPS KAKENHI (Grants No. JP19H00868, No. JP20K15185, No. JP22H05195, JP23H04800). We thank Ryugo Iwami, André Röhm, and Dhruvit Patel for productive discussions.

APPENDIX A: MULTIDIMENSIONAL TARGET SIGNAL

We clarify how the prediction of an m -dimensional target signal for the linear attention-enhanced reservoir computer works in detail. From Eqs. (8) and (9) we get the modified versions for M dimensions as follows:

$$\mathbf{w}_{\text{att},l,m} = \mathbf{W}_{\text{net},m} \mathbf{r}_l, \quad (\text{A1})$$

$$d_{l,m} = \mathbf{w}_{\text{att},l,m}^T \mathbf{r}_l. \quad (\text{A2})$$

Here the matrix $\mathbf{W}_{\text{net},m} \in \mathbb{R}^{N \times N}$ consists of the weights that are trained via the gradient-descent algorithm, where N is the number of reservoir nodes. The index m runs over all target dimensions M . The matrix $\mathbf{W}_{\text{net},m}$ is multiplied with the standardized reservoir state vector \mathbf{r}_l yielding the attention vector for the m th dimension $\mathbf{w}_{\text{att},l,m} \in \mathbb{R}^{N \times 1}$. Thus every single dimension of the target system has its own attention weight vector $\mathbf{w}_{\text{att},l,m} \in \mathbb{R}^{N \times 1}$, and thus its own attention layer weights $\mathbf{W}_{\text{net},m} \in \mathbb{R}^{N \times N}$ matrix.

After computing the attention weights, $\mathbf{w}_{\text{att},l,m}$ is multiplied with the reservoir state \mathbf{r}_l to yield the final prediction $d_{l,m}$.

APPENDIX B: NONLINEAR ATTENTION

In our proof-of-concept demonstration, we used a linear transformation as a simplest form of a “neural network” for the attention layer. To show a more complex architecture, we use a three-layer feed-forward neural network to compute the attention weights for every dimension of the target system, i.e., $M = 3$ for the tasks in this paper. A sketch of the nonlinear attention approach is shown in Fig. 12. We can alter Eqs. (A1) and (A2) accordingly as follows:

$$\mathbf{w}_{\text{att},l,m} = F(\mathbf{W}_{\text{net},m}, \mathbf{r}_l), \quad (\text{B1})$$

$$d_{l,m} = \mathbf{w}_{\text{att},l,m}^T \mathbf{r}_l. \quad (\text{B2})$$

Here $F(\mathbf{W}_{\text{net},m}, \mathbf{r}_l)$ is a neural network that takes \mathbf{r}_l as input and has $\mathbf{W}_{\text{net},m}$ trainable weight parameters. The input

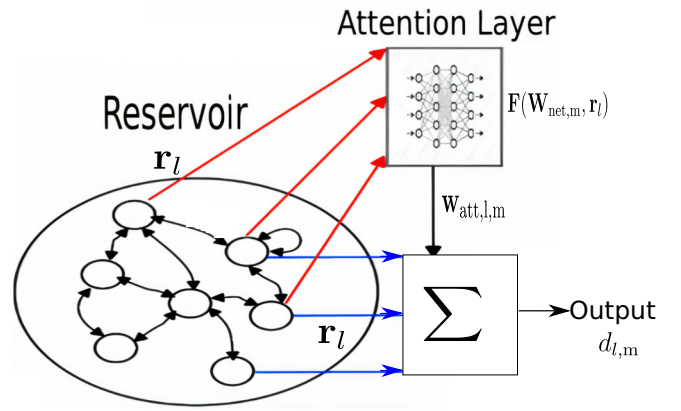


FIG. 12. Schematic of the attention-enhanced reservoir computing setup with a nonlinear attention layer. The attention layer is replaced by a neural network, that yields the attention weights $\mathbf{w}_{\text{att},l,m}$, which are then weighted with the reservoir node states to generate an output.

layer consists of the N reservoir states, while the remaining architecture can be chosen freely. One hidden layer consisting of N nodes is chosen. The output layer of each network has N nodes, which compute the N attention weights $\mathbf{w}_{\text{att},l,m}$ for each target dimension $m \in \{M\}$. Collecting all weights, the attention layer consists of $6N^2$ weights, where N^2 comes from the interlayer weights, and the factor 6 results from three networks all having two interlayer weights.

We use this approach as a nonlinear attention, and compare its performance for our UCTLS system to the linear attention and the ridge-regression approach over the reservoir size. The results are depicted in Fig. 13. It is shown that a significantly lower NRMSE and higher VPT is achieved using the nonlinear attention scheme, especially for smaller reservoirs. This advantage is reduced for larger reservoirs, stemming from the limited complexity of the problem. Therefore, future plans involve applying the attention-enhanced reservoir to more complex tasks.

APPENDIX C: CONVERGENCE OF THE LOSS

The attention-enhanced reservoir computer loses the advantage of the analytically solvable ridge-regression approach, because a gradient-descent algorithm has to be used. The output-layer optimization of reservoir computing is not a limiting factor due to the recent advances in hardware acceleration. Considering the amount of research conducted to develop photonic based accelerators for neural networks, we see no disadvantage using a gradient descent, in optimizing the output layer. In addition, the advantage of ridge regression consists only for small reservoirs due to the complexity of ridge regression compared to a gradient-descent-based approach. The order of the ridge regression is roughly $\mathcal{O}(L^3)$, where L is the number

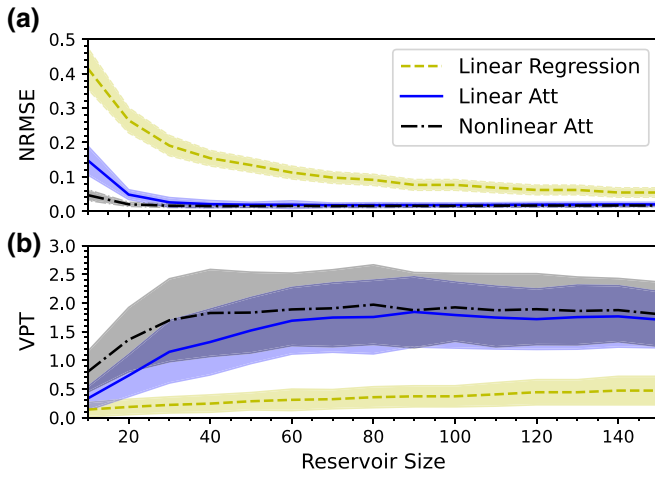


FIG. 13. (a) NRMSE and (b) VPT for classical linear regression (dashed yellow line), linear attention (solid blue line), and nonlinear attention-enhanced reservoir computing (dashed-dotted black line) with varying reservoir size N for UCTLs with a forcing strength of zero, essentially making it a classic single Lorenz task. The shaded areas depict the standard deviations of NRMSE and VPT over ten time series.

of parameters used. In contrast, the order of the gradient-descent algorithm is $\mathcal{O}(LKT)$, where K is the number of steps needed for convergence, and T is the number of training data points. For our approach, the two complexities thus become $\mathcal{O}(N^3)$ and $\mathcal{O}(N^2KT)$ for the ridge regression and the gradient-descent approach, respectively, where N is the size of the reservoir. Thus, generally for large L and N , the gradient-descent algorithm may be faster, depending on the number of data points and epochs for convergence. We believe that an increase in reservoir computer size with an increase in the output layer may enable the reservoir computer to tackle more complex tasks.

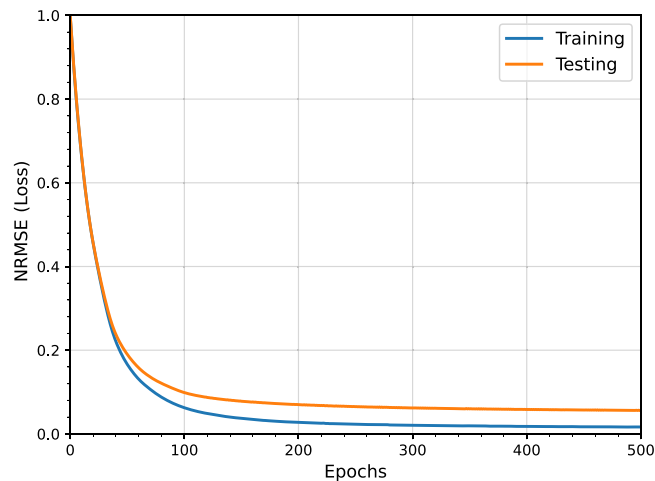


FIG. 14. Training and testing NRMSE (loss) over the epochs of the gradient-descent algorithm.

To test the complexity, we plotted the NRMSE over the number of epochs for an 50-node reservoir ($N = 50$) using the linear attention approach, as shown in Fig. 14. It is clear that the NRMSE exponentially and asymptotically converges to a minimum.

- [1] Y. Shen, N. C. Harris, S. Skirlo, M. Prabhu, T. Baehr-Jones, M. Hochberg, X. Sun, S. Zhao, H. Larochelle, D. Englund, and M. Soljačić, Deep learning with coherent nanophotonic circuits, *Nat. Photonics* **11**, 441 (2017).
- [2] L. Larger, M. C. Soriano, D. Brunner, L. Appeltant, J. M. Gutiérrez, L. Pesquera, C. R. Mirasso, and I. Fischer, Photonic information processing beyond turing: An optoelectronic implementation of reservoir computing, *Opt. Express* **20**, 3241 (2012).
- [3] M. Nakajima, K. Tanaka, and T. Hashimoto, Scalable reservoir computing on coherent linear photonic processor, *Commun. Phys.* **4**, 20 (2021).
- [4] D. Brunner, M. C. Soriano, C. R. Mirasso, and I. Fischer, Parallel photonic information processing at gigabyte per second data rates using transient states, *Nat. Commun.* **4**, 1364 (2013).
- [5] J. Nakayama, K. Kanno, and A. Uchida, Laser dynamical reservoir computing with consistency: An approach of a chaos mask signal, *Opt. Express* **24**, 8679 (2016).
- [6] J. Bueno, D. Brunner, M. C. Soriano, and I. Fischer, Conditions for reservoir computing performance using semiconductor lasers with delayed optical feedback, *Opt. Express* **25**, 2401 (2017).
- [7] C. Sugano, K. Kanno, and A. Uchida, Reservoir computing using multiple lasers with feedback on a photonic integrated circuit, *IEEE J. Sel. Top. Quantum Electron.* **26**, 1500409 (2020).
- [8] K. Takano, C. Sugano, M. Inubushi, K. Yoshimura, S. Sunada, K. Kanno, and A. Uchida, Compact reservoir computing with a photonic integrated circuit, *Opt. Express* **26**, 29424 (2018).
- [9] S. Sackesyn, C. Ma, J. Dambre, and P. Bienstman, Experimental realization of integrated photonic reservoir computing for nonlinear fiber distortion compensation, *Opt. Express* **29**, 30991 (2021).
- [10] M. Rafayelyan, J. Dong, Y. Tan, F. Krzakala, and S. Gigan, Large-scale optical reservoir computing for spatiotemporal chaotic systems prediction, *Phys. Rev. X* **10**, 041037 (2020).
- [11] S. H. Strogatz, *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry and Engineering* (Westview Press, Boulder, 2000).
- [12] L. Appeltant, M. C. Soriano, G. Van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C. R. Mirasso, and I. Fischer, Information processing using a single dynamical node as complex system, *Nat. Commun.* **2**, 468 (2011).
- [13] J. Torrejon, M. Riou, F. A. Araujo, S. Tsunegi, G. Khalsa, D. Querlioz, P. Bortolotti, V. Cros, K. Yakushiji, A. Fukushima, H. Kubota, S. Yuasa, M. D. Stiles, and J. Grollier, Neuromorphic computing with nanoscale spintronic oscillators, *Nature* **547**, 428 (2017).

- [14] K. Nakajima, H. Hauser, T. Li, and R. Pfeifer, Information processing via physical soft body, *Sci. Rep.* **5**, 10487 (2015).
- [15] T. Kotooka, S. Lilak, A. Stieg, J. Gimzewski, N. Sugiyama, Y. Tanaka, H. Tamukoh, Y. Usami, and H. Tanaka, Ag₂Se nanowire network as an effective in-materio reservoir computing device (2021), preprint, [ResearchSquare:10.21203/rs.3.rs-322405/v1](https://arxiv.org/abs/10.21203/rs.3.rs-322405/v1).
- [16] H. Jaeger, The “echo state” approach to analysing and training recurrent neural networks-with an erratum note, Bonn, Germany: German National Research Center for Information Technology GMD Technical Report **148**, 13 (2001).
- [17] G. Tanaka, T. Yamane, J. A. He, R. Nakane, S. Kanazawa, H. Takeda, H. Numata, D. Nakano, and A. Fujii, Recent advances in physical reservoir computing, *Neural Netw.* **115**, 100 (2019).
- [18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, in *Advances in Neural Information Processing Systems* (Long Beach, 2017), p. 5998.
- [19] S. Bai, J. Z. Kolter, and V. Koltun, An empirical evaluation of generic convolutional and recurrent networks for sequence modeling, [arXiv:1803.01271](https://arxiv.org/abs/1803.01271).
- [20] E. N. Lorenz, Deterministic nonperiodic flow, *J. Atmos. Sci.* **20**, 130 (1963).
- [21] K. Kanno, M. Naruse, and A. Uchida, Adaptive model selection in photonic reservoir computing by reinforcement learning, *Sci. Rep.* **10**, 10062 (2020).
- [22] A. Uchida, *Optical Communication with Chaotic Lasers: Applications of Nonlinear Dynamics and Synchronization* (Wiley-VCH, Weinheim, 2012).
- [23] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, *Dive Into Deep Learning* (Cambridge University Press, Cambridge, 2023). <https://D2L.ai>.
- [24] D. Bahdanau, K. Cho, and Y. Bengio, Neural machine translation by jointly learning to align and translate, [arXiv:1409.0473](https://arxiv.org/abs/1409.0473).
- [25] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, Cambridge, 2016).
- [26] M. Cucchi, S. Abreu, G. Ciccone, D. Brunner, and H. Kleemann, Hands-on reservoir computing: a tutorial for practical implementation, *Neuromorphic Comput. Eng.* **2**, 032002 (2022).
- [27] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: with Applications in R* (Springer, New York, 2013), includes bibliographical references and index.
- [28] 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, *Phys. Rev. Lett.* **120**, 024102 (2018).
- [29] M. O. Fen, Persistence of chaos in coupled lorenz systems, *Chaos Solitons Fractals* **95**, 200 (2017).
- [30] H. D. Abarbanel, R. Brown, J. J. Sidorowich, and L. S. Tsimring, The analysis of observed chaotic data in physical systems, *Rev. Mod. Phys.* **65**, 1331 (1993).