PLOS COMPLEX SYSTEMS

RESEARCH ARTICLE

Signal demixing using multi-delay multi-layer reservoir computing

S. Kamyar Tavakoli[1]*, **André Longtin**[1,2,3]*

1 Department of Physics, University of Ottawa, STEM Complex, 150 Louis-Pasteur Pvt, Ottawa, Ontario, Canada, **2** Center for Neural Dynamics and AI, University of Ottawa, Ottawa, Ontario, Canada, **3** Department of Cellular and Molecular Medicine, Faculty of Medicine, University of Ottawa, Ottawa, Ontario, Canada

* stava089@uottawa.ca (SKT); alongtin@uottawa.ca (AL)

OPEN ACCESS

Citation: Tavakoli SK, Longtin A (2025) Signal demixing using multi-delay multi-layer reservoir computing. PLOS Complex Syst 2(2): e0000034. https://doi.org/10.1371/journal.pcsy.0000034

Editor: Dimitris Kugiumtzis, Aristotle University of Thessaloniki, GREECE

Received: June 14, 2024

Accepted: December 16, 2024

Published: February 18, 2025

Data Availability Statement: All relevant data are within the manuscript.

Funding: This work was supported by the Natural Sciences and Engineering Research Council of Canada (RGPIN/06204-2014 to AL). The funder had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing interests: The authors have declared that no competing interests exist.

Abstract

Brain circuitry involves a large number of recurrent feedback loops whose dynamics depend on interaction delays. Brain-inspired reservoir computing leverages the rich recurrent dynamics of interconnected units for performing tasks on inputs. In particular, time-delay reservoir computing uses the high-dimensional transient dynamics in nonlinear delayed feedback loop architectures for e.g. time series prediction and speech classification. The modification of the dynamical properties of delay-differential systems through the inclusion of multiple delays has also recently been shown to improve the performance of time-delay reservoir computing. Here we explore another aspect of such neuro-inspired computing of fundamental and technological importance: the ability to separate and predict two signals in a mixture, where each has some intrinsic predictability due to its underlying dynamics. This is illustrated using multi-delay and multi-layer reservoir computing with chaotic input signal mixtures. In contrast to Independent Component Analysis and related unsupervised learning techniques, the context here consists in the parallel supervised learning of the dynamics for each signal in order to predict each of them beyond the training set. Further, the superposition of the chaotic signals into a single input channel adds to the difficulty of the task. We quantify and explain this performance with various signals emanating from both deterministic and stochastic systems. Additionally, we explore the architecture of deep time-delay reservoir computers. Our findings demonstrate that multi-delay reservoir computing can learn and predict the future of two superimposed deterministic signals. Prediction—and thus separation—accuracy can be significantly higher in single and multi-layer time-delay reservoir computing when the first layer contains multiple delays. Bandpass filtering of the mixed signal to remove lower and higher frequencies improved the prediction by a few percent. In some cases, paradoxically, increasing the proportion of one chaotic signal in the mixture can actually help the learning of another chaotic signal, and thus slightly improve its prediction.

Author summary

Chaotic dynamical systems generate datasets that are challenging to predict due to their sensitive dependence on initial conditions. This is especially the case in the presence of

noise. Despite this complexity, these systems possess properties known as dynamical invariants—including Kolmogorov-Sinai entropy, Kaplan-Yorke dimension, and correlation dimension—as well as distinctive temporal correlation properties that facilitate the analysis of their behaviour. Reservoir computing is a class of biologically-inspired machine learning models that can accurately predict these time series. A particularly challenging scenario occurs when trying to extract and predict individual signals in a mixture, where each has its own underlying deterministic and stochastic properties. This is required e.g. for demixing sources of human speech or other animal sounds, as well as superimposed biomedical signals such as heart rate and EEG signals. In this study, we employ a supervised learning approach to train two weight matrices that combine the activity of the reservoir's computing elements into two output signals. This enables the prediction—and therefore separation—of both signals from a linear mixture. Our approach not only advances nonlinear signal processing techniques but also provides insight into the design of systems that mimic some aspects of human auditory perception.

1 Introduction

In many real-world applications, we frequently encounter mixed signals emanating from complex systems, necessitating advanced methods for effective separation [1–3]. Effective signal demixing might be achieved by uncovering genuine interactions despite the presence of noise [4]. An interesting case is found in the context of weakly electric fish, where global feedback mechanisms cancel redundant self-generated inputs such as body motion in order to better focus on unpredictable environmental signals [5–7].

Classical methods such as Independent Component Analysis (ICA) can effectively disentangle statistically independent sources, when the signal components are non-Gaussian, thus broadening the scope of techniques for signal separation [8, 9]. A recurrent neural network (RNN) model has been studied to separate original signals from mixed signals using minimal prior knowledge about the statistical characteristics of the sources and the mixing process [10]. RNNs are well-suited for this task due to their dynamical properties, inherent memory, and their ability to enhance data separability, facilitating more effective signal processing. Deep learning has also been shown to be an effective method for signal separation [11–13].

An innovative method for extracting chaotic signals from a mixture was introduced by Bassett et al. [14]. Their technique utilizes the high-dimensional properties of a dynamical system, in their case water waves in a tank that is driven by an "input" mixture of chaotic signals, in order to separate out its components. Their technique utilizes a reservoir computing (RC) framework, with the computer exemplified by a tank of water, to process these mixed signals. Reservoir computing is a type of machine learning that utilizes a dynamical system made up of many interconnected units or computational elements, in order to process input signals. This computer is trained solely on the output layer, usually in a supervised manner, to perform various tasks without altering the internal structure of the network. RC has demonstrated potential in machine learning by embedding input signals into higher-dimensional spaces through high-dimensional dynamics [15]. In this paper, we employ time delay RC for the prediction of two complex signals, thereby enabling their separation. While classical RC approaches rely on large networks of interconnected nonlinear nodes to create a high-dimensional dynamic system, time-delay RC offers a lower-complexity alternative that can be effectively implemented experimentally using electronic or photonic devices.

Time delay reservoir computing is an innovative approach widely applied in machine learning tasks, including speech recognition and time series prediction [16, 17]. It relies on the infinite-dimensional properties of delay differential equations. In these equations, time delays significantly alter the system's dynamical behavior and properties. For instance, in semiconductor lasers, the introduction of time delays has been shown to increase the complexity of the dynamics, making these systems suitable for applications such as random bit generators and encryption devices [18, 19]. Time delays can also have a stabilizing effect, such as in neural networks where long delayed feedback decreases the complexity of the dynamics [20], or in control methods where selecting a time delay that matches the period of unstable periodic orbits suppresses chaotic dynamics [21, 22]. In brain circuitry, the presence of interaction delays in recurrent loops results in multistability and plays a role in memory [23]. Although time delay reservoir computing operates in the fixed point regime, the complexity introduced by time delays results in a rich repertoire of transient responses that can enhance learning. These transient responses can be exploited to train weight matrices for performing different machine learning tasks. Unlike typical reservoir computing neural networks, which consist of many units where input is injected and their dynamics are used to project the input into a high-dimensional space, time delay reservoir computing can utilize a single nonlinear element to process the input. In both cases however, the reservoir is the high-dimensional dynamics.

Time delay reservoir computing has also proven effective in classifying mixed sinusoidal and square wave signals using mutually coupled semiconductor lasers [24]. While effective implementation of this system requires maintaining operations within the stable fixed point regime, the performance is closely linked to dynamical properties such as bifurcation characteristics and dynamical invariants [25]. However, recent research suggests there may be benefits to operating reservoir computers in oscillatory or bistable regimes, which can potentially enhance their computational capacity [26]. To optimize performance, various techniques are employed to enhance training effectiveness by eliciting high-dimensional responses. These techniques often involve refining how signals are injected into the equations of the reservoir computer or altering the system's architecture and configuration, such as implementing layered or parallel architectures, to enhance prediction accuracy [27–29]. Furthermore, approaches such as Kalman filtering, which is employed in adaptive training, and transfer learning, have recently been implemented to further improve the performance of reservoir computing systems [30, 31].

In this work, we exploit the capabilities of time-delay reservoir computing to tackle two complex signal separation tasks. Our study reveals the capability of time-delay reservoir computing systems to make accurate signal predictions, even when trained on perturbed signals. This demonstrates that time-delay RC can effectively adapt to variations in input, which is particularly useful in real-world applications where signals are often contaminated by noise or other extraneous inputs. However, this capability is not always guaranteed and largely depends on the complexity of the dynamical systems that generated the data in the mixture. In our study, the perturbations originate either from another chaotic system or from a noisy system, highlighting the versatility of a time-delay RC in managing different types of interference.

The first scenario we considered involves analyzing a mixed signal from two chaotic time series, where preliminary autocorrelation analysis indicates that one signal has a much shorter correlation time compared to the other. Detailed autocorrelation plots of the input signals are provided in the Methods section. It has been shown that the properties of such input signals critically influence the choice of parameters for optimizing reservoir computing systems. However, techniques such as incorporating multiple delays in the dynamical equations [32] or optimizing internal and external delays have been found to reduce the sensitivity of prediction

errors to hyperparameters [33]. The second scenario uses a mixture of a deterministic chaotic signal and a stochastic signal from a linear noisy system.

For both of these scenarios, we consider training two types of time-delay RCs. The first is a multiple delay RC approach to demonstrate the effects of multiple delays and dynamic behavior on prediction accuracy. Furthermore, we explore a more complex architecture of time-delay reservoir computing that incorporates additional layers. In this advanced setup, the first layer can be configured with either multiple delays or a single delay, while all subsequent layers in both cases are configured with a single delay.

While we employ established techniques from the literature, such as multiple delays and layers, we combine these approaches to tackle the more intricate problem of supervised learning of two signals linearly added to form a mixture (two chaotic signals from different dynamical systems, or a chaotic signal and a stochastic signal) followed by the prediction of those signals beyond the training set. Our goal is not to provide the best predictions by benchmarking delay-RC against other RC algorithms, but to show that supervised learning to separate signals and predict them is feasible. This approach can thus benefit from the technical advantages of delay-RC such as fast computing with electro-optical hardware, low circuit complexity (there is just one nonlinear device plus delay loops), and one location where input enters the RC. The novel algorithm proposed here combines techniques to enhance the capabilities of reservoir computing with time-delayed dynamics.

2 Results

Our approach involves feeding a "training" mixed signal into the RC and obtaining a single system response at all its virtual nodes. This response is then used to compute two separate linear regressions, thus performing supervised learning of the temporal evolution of both signals in the mixture. This learning method produces a separate column matrix of coefficients for each signal. We could stack these two matrices horizontally we form a $N \times 2$ weight matrix (N is the number of virtual nodes in the RC—see Methods), where each column corresponds to one of the individual signals, or keep working with both matrices; it is a matter of computational preference. In either case, the algorithm then computes the response of the RC to a new (i.e. unseen) "test" signal mixture that immediately follows the training set. With these new responses and the weight matrices, the algorithm then performs two predictions beyond the test set, producing two output streams.

We considered two scenarios for the inputs and the methods used to generate the time series data that constitute these inputs are outlined in Sec. 4. The first one involves the linear superposition of time series generated from the Mackey-Glass (MG) system and taking the x component of the chaotic Lorenz system. The second scenario involves a mixture of time series from the MG system and a linear noise system, namely the Ornstein-Uhlenbeck (OU) process, as shown for different values of the correlation time τ_{OU} in Fig 1. The goal of this second scenario is two-fold: 1) to explore how it is possible to extract one deterministic input when it is added to background noise with low or high correlation, and 2) to quantify whether some aspects of the results for the mixture of MG and Lorenz x can be explained by the interference of a random process with a deterministic prediction task, particularly when both have similar variances.

We first present the results for the time-delay reservoir computer with a single layer and multiple delays. Next, we demonstrate the effectiveness of adding layers to the reservoir computer, where the first layer contains single or multiple delays, and the added layers containing a single delay. We evaluate the prediction accuracy for different mixing ratios.

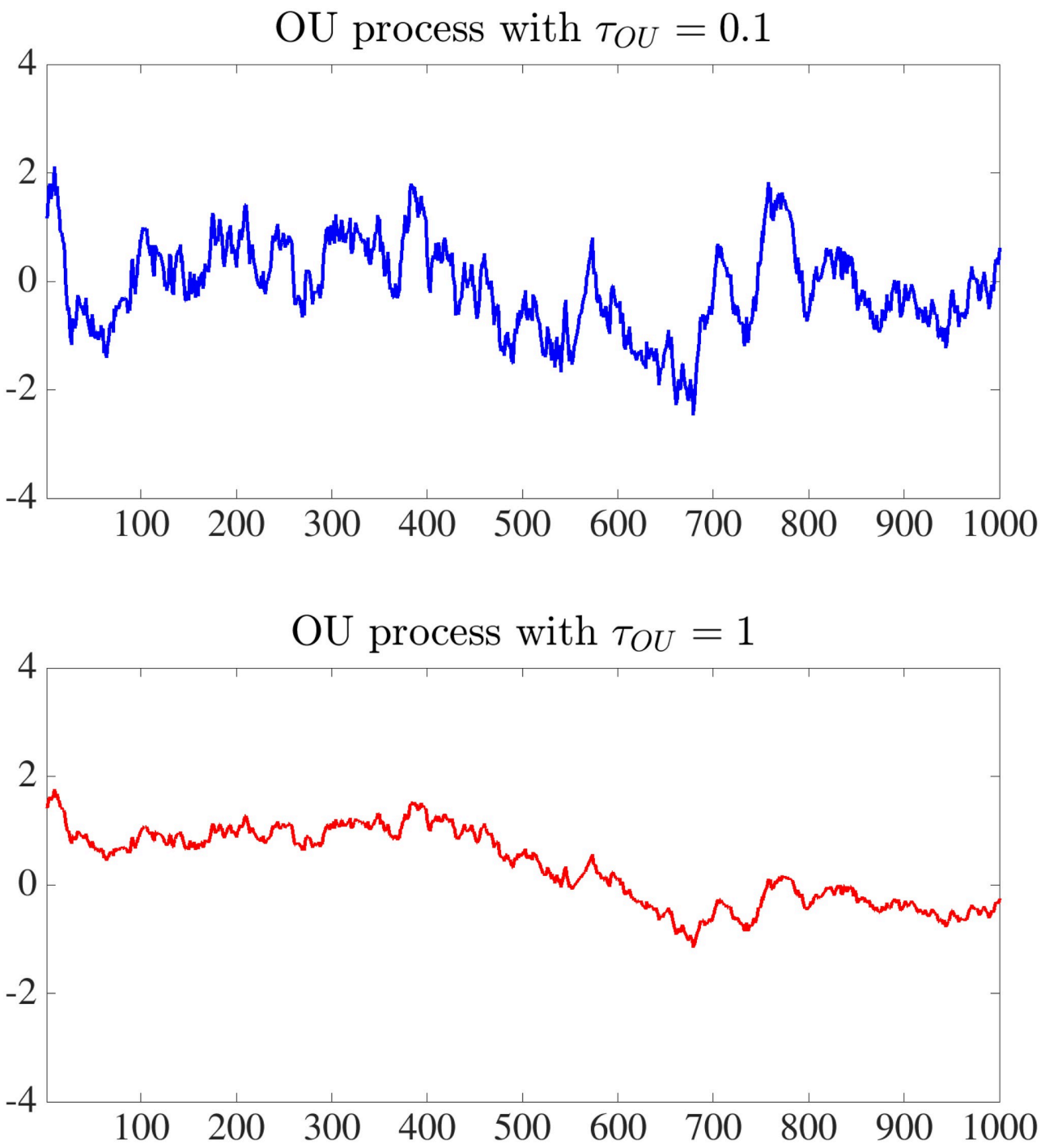

Fig 1. Time series associated with Eq 13. The top panel depicts the time series for the OU process with $\tau_{OU} = 0.1$, and the lower panel shows the time series for $\tau_{OU} = 1$. The data are normalized to have zero mean and unit variance.

https://doi.org/10.1371/journal.pcsy.0000034.g001

2.1 Multiple delays, single layer

Incorporating multiple delays into the dynamical equations of a reservoir computer has been shown to significantly enhance its memory capacity, thus improving prediction accuracy [32]. Memory capacity in the context of reservoir computing is defined as the ability of the reservoir of high-dimensional dynamics to remember previous inputs, which is typically calculated by setting the target to previous inputs at specific steps before and then calculating the correlation between the predicted values and actual values.

Various models employing delay differential equations, ranging from opto-electronic to photonic to electronic devices, have been explored for implementing time delay reservoir computers. In this study, we chose to focus on the electro-optic oscillator model as the core nonlinear element of the reservoir computer, due to its rich dynamical properties and the extensive range of parameters available when incorporating multiple delays and a filter [32, 34]. The incorporation of a secondary equation, which serves as a filter, is a well-established approach that modifies the dynamics. By incorporating multiple delay terms this results in an expanded stabilized regime within the parameter space, notably a stable fixed point regime. The stability diagram (Fig 2) demonstrates how the stabilized regime expands within the parameter space as the number of delays increases. We computed this diagram using DDE-Biftool [35] by solving the characteristic equations of the delayed system to determine the stability. From the stability diagram illustrated in Fig 2, we select two distinct parameter sets for implementing the reservoir computers. The first set involves large spacing between delays, specifically $\Delta\tau = 39.8$, where the feedback coefficient required for fixed point destabilization closely approximates that of the single delay case. This value is chosen based on a clock cycle of $T = 40$, which we introduce in detail in Section 4.1. The second set uses a smaller spacing between delays, with $\Delta\tau = 2.66$ for two delays ($M = 2$) and $\Delta\tau = 1.6$ for five delays ($M = 5$). Fig 2 shows that the threshold for destabilization is significantly higher for moderate spacing between delays when $M > 1$ compared to the single delay scenario. This necessitates a higher feedback coefficient for smaller spacings to approach the destabilization point, whereas larger spacings require feedback coefficients similar to those seen in the single delay case.

In systems characterized by a single delay, parameter selection is constrained to factors such as τ (time delay), γ (input scaling), and β (feedback strength), with the feedback strength generally maintained at a low level. However, with multiple delays, the parameter choices broaden, including $\Delta\tau$ (spacing between delays). This necessitates strategic decisions, such as opting for either moderate spacing with a high feedback coefficient or large spacing with weak feedback strength.

We have set the minimum time delay, denoted as τ_{min} (or simply τ in scenarios with a single delay), to 39.8 ($\tau_{min} = T - \theta$). We illustrate the prediction error in the $\beta - \gamma$ parameter space for three different signal mixtures in Fig 3. The top row displays the 'prediction error' from the task of separating a mixture of the time series of Lorenz x and MG. The second row shows the MG time series mixed with the OU process for $\tau_{OU} = 0.1$, and the third row presents the same for $\tau_{OU} = 1$. As shown in Fig 2, for the five delay case ($M = 5$), a lower feedback coefficient (β) is required when the spacing between delays is $\Delta\tau = 39.8$. In contrast, when the spacing is $\Delta\tau = 1.6$, a higher feedback coefficient is needed for destabilization. Therefore, the parameter ranges in Fig 3(c), 3(f) and 3(i) differ from those in the other panels.

When the mixture contains equal proportions of both signals, the system requires a higher feedback coefficient and a small input scaling. As the feedback coefficient increases, the system approaches the bifurcation where the fixed point loses stability, leading to richer transient responses elicited by the input mixture. Meanwhile, smaller input scaling helps maintain stability, preventing the system from being overly driven by input fluctuations. From Fig 3, it's

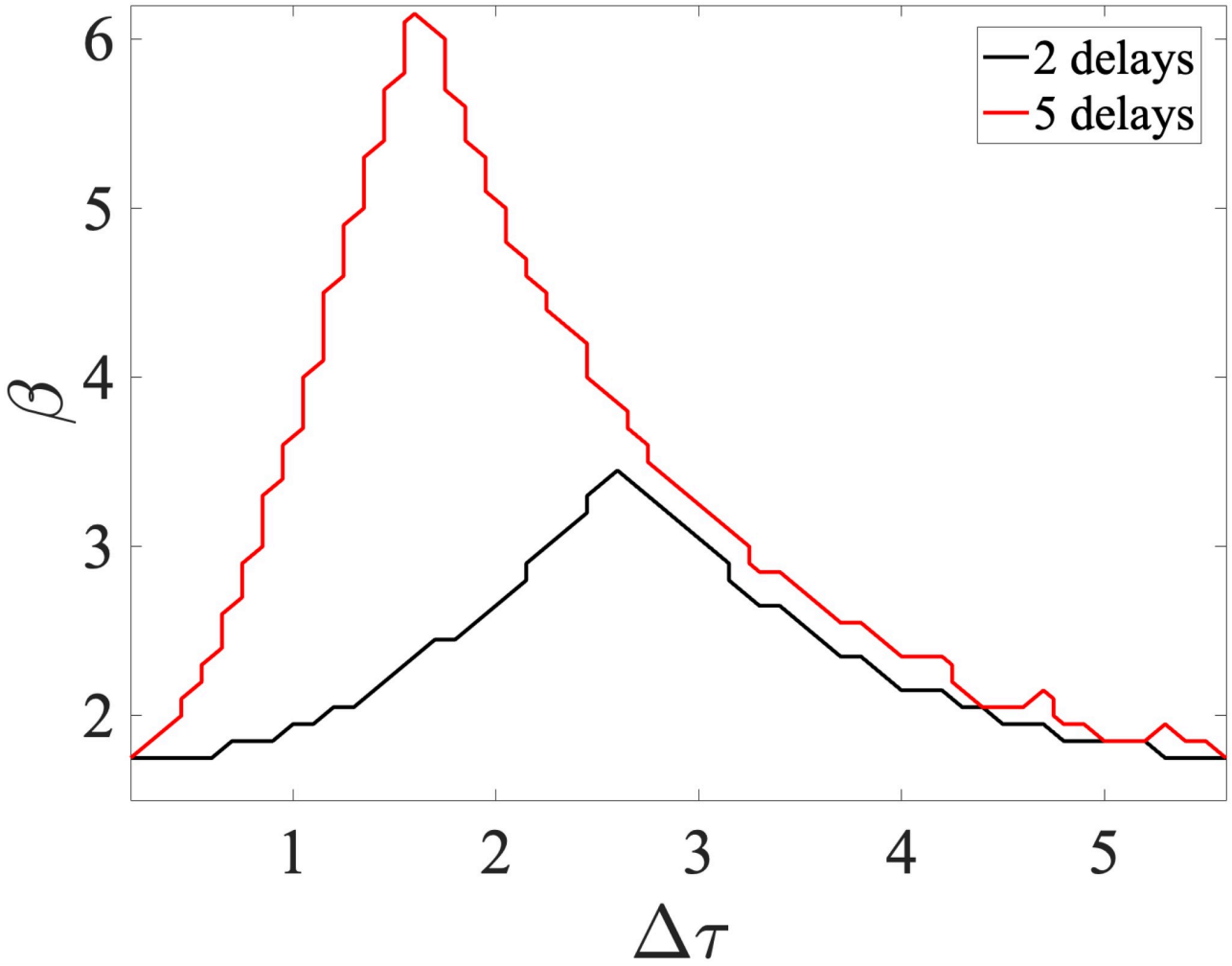

Fig 2. Linear stability analysis of Eq 1 for a multi-delay RC system in the absence of the multiplexed masked input signal, shown for (black line) two delays and (red line) five delays. Increasing the number of delays causes the system to require a higher feedback coefficient at moderate spacings between delays ($\Delta\tau$) to become unstable. Beneath the curves, the fixed points are stable. Above the curves, the stable fixed points lose their stability, leading to oscillatory and chaotic dynamics.

https://doi.org/10.1371/journal.pcsy.0000034.g002

clear that the various RC setups achieve relatively similar error on the corresponding tasks. This provides us with the insight that each signal essentially interferes with the learning and prediction of the other signal, regardless of whether it is deterministic or stochastic. But the supervised learning of the deterministic signal can nevertheless occur in spite of the relatively poor learning of the stochastic signal, especially when its correlation time is short, i.e. that it is close to white noise. In all cases, introducing additional delays into the reservoir computer configuration provides a broader range of parameter choices, enhancing the flexibility of the system setup. Specifically, for the first task involving Lorenz x and MG, and the third task featuring MG with the OU process at $\tau_{OU} = 1$, the introduction of additional delays leads to a modest 3% improvement. For the second task, which combines MG with the OU process at $\tau_{OU} = 0.1$, there is a more noticeable improvement of 8%.

In Fig 4, the prediction error for different mixing ratios (s; see Methods section) is shown. In the Lorenz x + MG mixture, when MG is the dominant component, the prediction accuracy for MG remains similar across all configurations, whereas for Lorenz x, the most effective

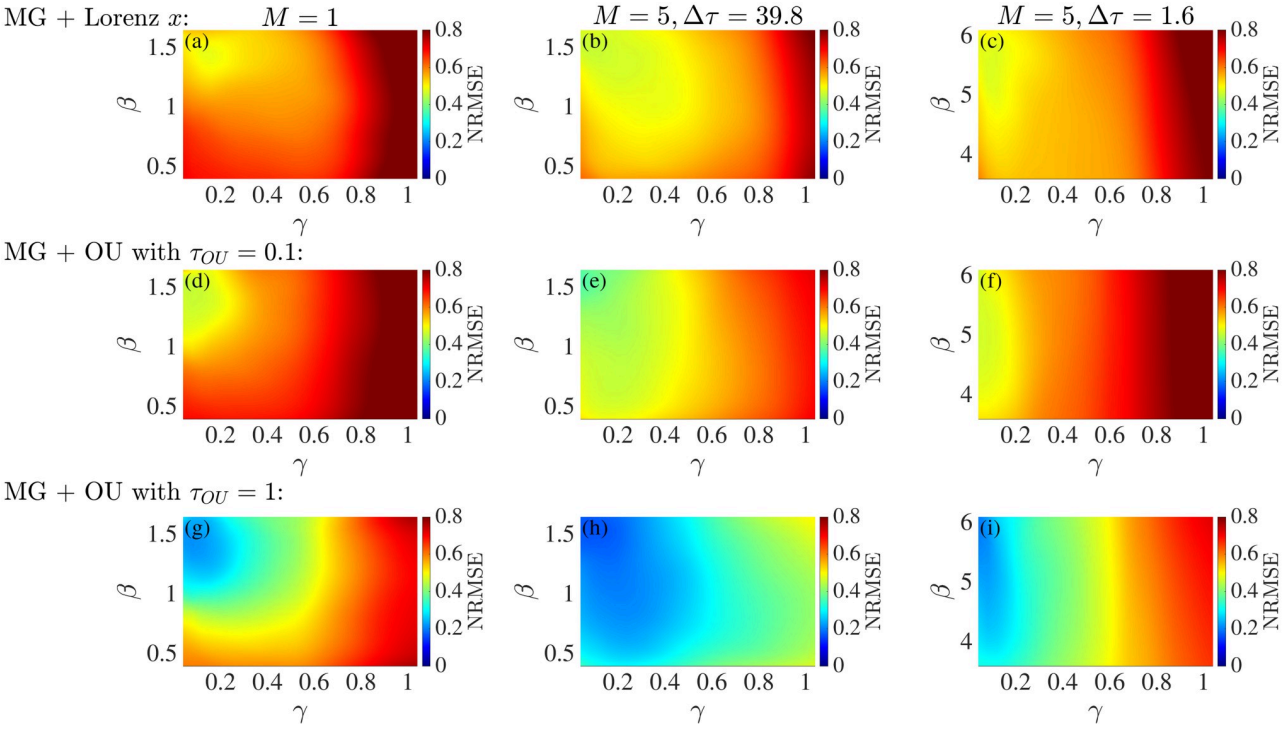

Fig 3. Error color maps displaying the performance of different configurations of the RC for a mixture of two distinct time series at a mixing ratio of $s = 0.5$ (the mixing ratio s determines the proportion of the signals in the mixture; details are in the Methods section 4.1). The top row illustrates results for the mixture of Lorenz x and MG time series, while the second and third rows represent the MG time series mixed with the OU process $\tau_{OU} = 0.1$ and $\tau_{OU} = 1$, respectively. All signals are scaled to have zero mean and unit variance. The columns correspond to different system configurations: the first column shows a system with a single delay, the second column a system with five delays ($M = 5$) and a large spacing between delays, and the third column a system with five delays and moderate spacing.

https://doi.org/10.1371/journal.pcsy.0000034.g003

setup is found to be the multi-delay system with large spacing between delays. However, the prediction error for Lorenz x remains too high, indicating that additional techniques are needed to reduce the error to an acceptable range.

When Lorenz x is the dominant signal, a multi-delay system with moderate spacing between delays is preferred, especially for accurately predicting the MG time series, which has a smaller contribution to the mixture. This aligns with our findings for single signal prediction, where highly complex time series benefit from configurations with moderate spacing between delays and high feedback coefficients. In contrast, for a correlated time series such as MG, a multi-delay system with large spacing between delays is more effective. The prediction errors for single-signal prediction tasks are significantly smaller for both signals. Interestingly, at certain points, a higher proportion of Lorenz x in the mixture unexpectedly improves the prediction accuracy for the MG signal. This is an unusual observation and warrants further investigation.

For the mixture of MG and a linear noisy time series, large spacing between delays yields much better performance compared to the other configurations. It is also noted that increasing the correlation time of the OU Process makes the signals more straightforward to predict. However, when the OU process is the dominant signal, the MG time series cannot be predicted with high accuracy.

2.2 Multiple delays, multiple layers

In our efforts to improve performance, we applied the framework of Deep Time Delay Reservoir Computing [25, 29]. Here, we considered that the first layer can have either a single delay

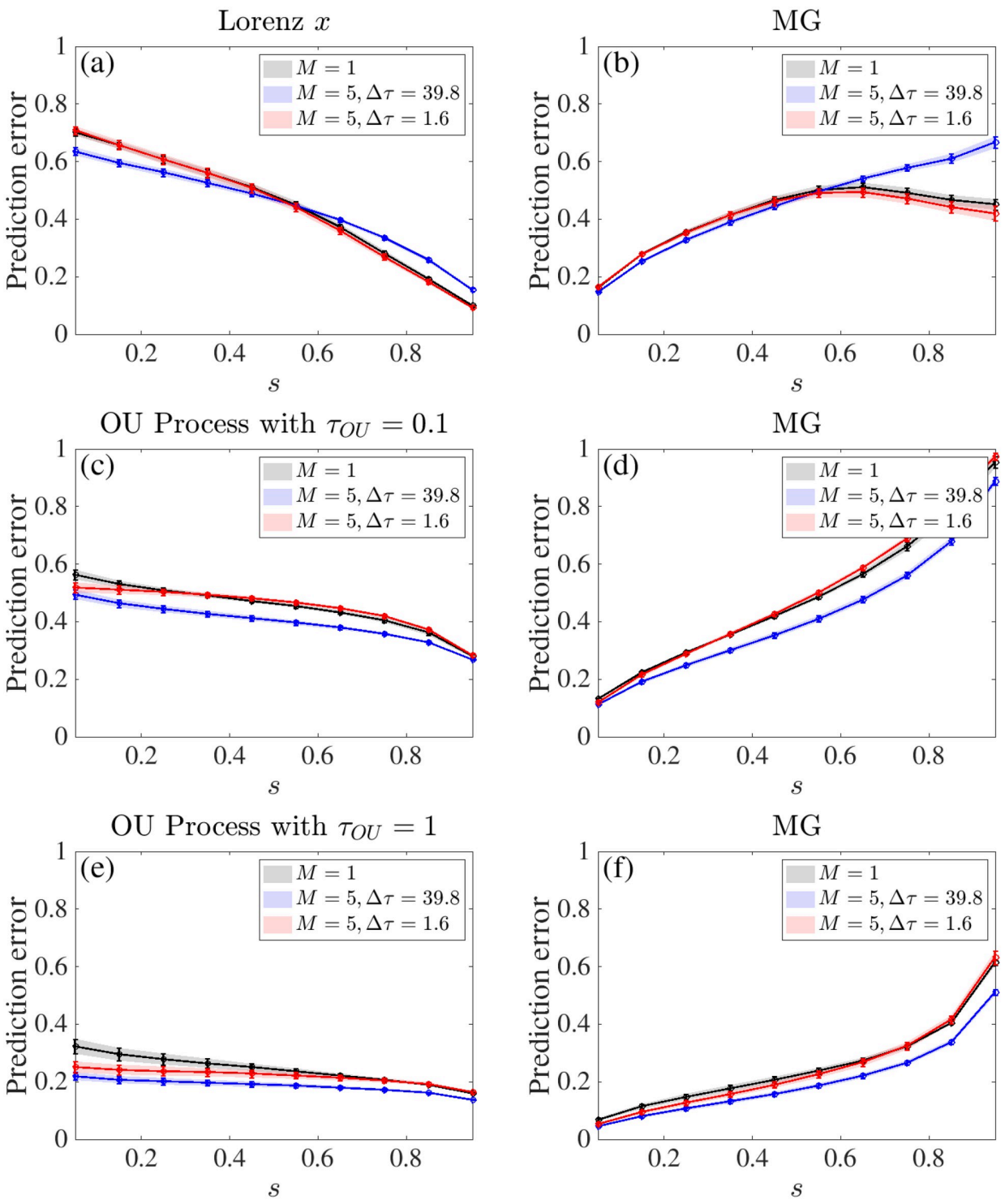

Fig 4. Prediction error as a function of the mixing ratio *s* for various system configurations, represented by different colors. The top row illustrates the prediction error for the mixture of Lorenz *x* and MG, while the second and third row pertains to the mixture of OU process and MG with different τ_{OU}. A high *s* value indicates a dominance of the OU process or the Lorenz *x* signal, while a small *s* corresponds to a dominance of the MG signal.

https://doi.org/10.1371/journal.pcsy.0000034.g004

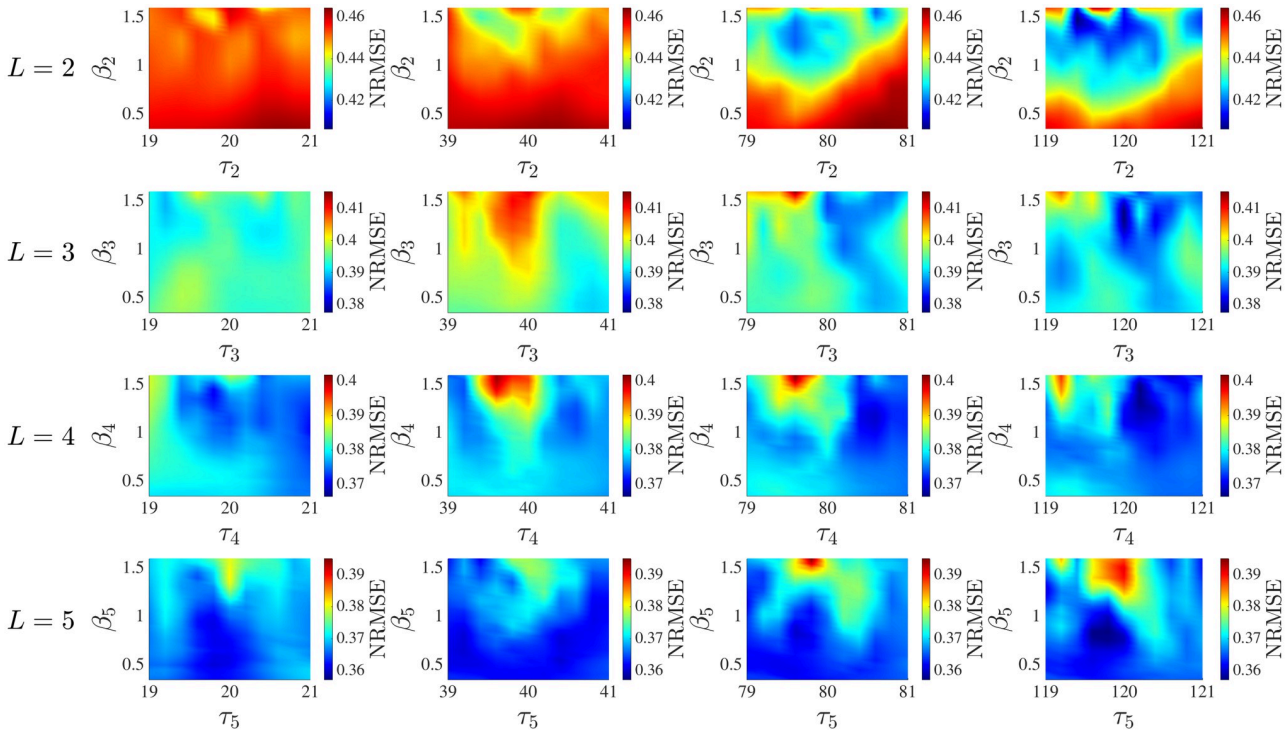

Fig 5. Prediction error for increasing number of Layers from top to the bottom to find the minimum error across different parameters. The first layers' parameters were obtained in Fig 3.

https://doi.org/10.1371/journal.pcsy.0000034.g005

or five delays, with either a small or large spacing between the delays when more than one is used. For the first layer, the parameters obtained in the previous section were applied. For the subsequent layers, the selection process for a single delay in the first layer is shown in Fig 5. In this figure, the input consists of a mixture of time series from the Lorenz x and the MG. It is difficult to find the optimum parameters for all layers. The parameters for each layer were determined incrementally. For instance, the optimal parameters for the first layer were derived from the findings of the previous section. Next, for the second layer, we plotted the prediction error across different values of τ_2 and β_2, and the best parameters were extracted from this analysis. The process was then repeated for the third layer; the parameters for the first and second layers were fixed to the previously found optimum values, and we searched for the optimal parameters for the third layer. It should be noted that the time delays considered were close to half of the clock cycle and also integer multiples of the clock cycle.

It has been observed that accuracy typically decreases and error values increase when the time delay associated with an added layer matches the clock cycle exactly. Optimal values for each layer's parameters, which prevent such coincidences, are detailed in Tables 1–3. The impact of adding layers on the reservoir computer's performance is depicted in Fig 6. Across all configurations, a significant enhancement in RC performance is noted with the addition of more layers. This improvement is particularly pronounced to those with multiple delays. However, there is a threshold beyond which further layer addition does not yield significant improvements in performance, suggesting an optimal number of layers beyond which additional complexity does not translate into further gains in accuracy.

For the mixture of Lorenz x and MG time series, as the number of layers increases, configurations with moderate spacings between delays begin to outperform other setups, and the

Table 1. Table of parameters utilized for prediction of MG signal and Lorenz *x*. Results can be found in Fig 7(a) and 7(b). Input scaling γ is set to 0.16 for the single delay scenario, and to 0.08 for both moderate and large spacings between delays.

Layer j	β_j			τ_{j1}		
	$M_1 = 1$	$M_1 = 5, \Delta\tau_1 = 39.8$	$M_1 = 5, \Delta\tau_1 = 1.6$	$M_1 = 1$	$M_1 = 5, \Delta\tau_1 = 39.8$	$M_1 = 5, \Delta\tau_1 = 1.6$
1	1.45	1.5	5.4	39.8	39.8	39.8
2	1.5	1.6	1.2	119.4	80	119.6
3	1.35	1.3	1.2	120	120	120
4	1.45	1.3	0.8	120.2	119.2	120
5	0.85	1.6	1.3	119.8	80.8	119.2

https://doi.org/10.1371/journal.pcsy.0000034.t001

Table 2. Parameter values for prediction of MG signal and OU Process with $\tau_{OU} = 0.1$. Input scaling is set to 0.04 across all scenarios. Results are presented in Fig 7(c) and 7(d).

Layer j	β_j			τ_{j1}		
	$M_1 = 1$	$M_1 = 5, \Delta\tau_1 = 39.8$	$M_1 = 5, \Delta\tau_1 = 1.6$	$M_1 = 1$	$M_1 = 5, \Delta\tau_1 = 39.8$	$M_1 = 5, \Delta\tau_1 = 1.6$
1	1.55	1.55	6	39.8	39.8	39.8
2	1.6	1.55	1.5	119.4	119.4	119.4
3	1.6	1.1	1.3	79.4	20.2	119.4
4	1.6	1.55	1.6	20	119.2	119.2
5	1.1	1.45	1.25	40.8	19.4	120.2

https://doi.org/10.1371/journal.pcsy.0000034.t002

Table 3. Parameter values obtained for prediction of MG signal and OU Process with $\tau_{OU} = 1$. Corresponding results are depicted in Fig 7(e) and 7(f). Input scaling γ is set to 0.08 for scenarios with a single delay and large spacing between delays, and to 0.04 for scenarios with moderate spacing between delays.

Layer j	β_j			τ_{j1}		
	$M_1 = 1$	$M_1 = 5, \Delta\tau_1 = 39.8$	$M_1 = 5, \Delta\tau_1 = 1.6$	$M_1 = 1$	$M_1 = 5, \Delta\tau_1 = 39.8$	$M_1 = 5, \Delta\tau_1 = 1.6$
1	1.45	1.6	6.1	39.8	39.8	39.8
2	1.55	1.4	1.45	119.4	119.4	119.4
3	0.7	1.45	1.5	119	20	19.6
4	0.9	1.05	1.6	21	39.4	119.4
5	1.6	1.25	1.5	80.2	120.6	39

https://doi.org/10.1371/journal.pcsy.0000034.t003

disparity in performance error increases. For both signals, there is at least a 15% increase in accuracy by increasing the number of layers from one to five. This trend also holds true for the OU process and MG time series; however, for these two correlated processes, configurations with multiple delays and large spacings consistently outperform those with smaller spacings.

In Fig 7, we have plotted the prediction error across different mixing ratios for a five-layer time delay reservoir computing system. When comparing this with Fig 4, where the mixture involves Lorenz x and MG time series, it is observed that for higher values of s (indicating a higher proportion of Lorenz x and a lower proportion of MG), the error difference across all configurations is relatively small for predictions of Lorenz x. However, for MG predictions, the difference in error becomes much more significant. This is particularly noticeable in configurations with moderate spacing between delays, where the performance improvement can be as large as 18%. For smaller s values (indicating a higher proportion of MG and a lower proportion of Lorenz x), improvements are noticeable in configurations with a single delay and those with multiple delays with large spacings between them.

For the mixture of MG time series and OU process, when $\tau_{OU} = 0.1$ and the mixing ratio s is small, adding layers significantly reduces the prediction error in multi-delay configurations,

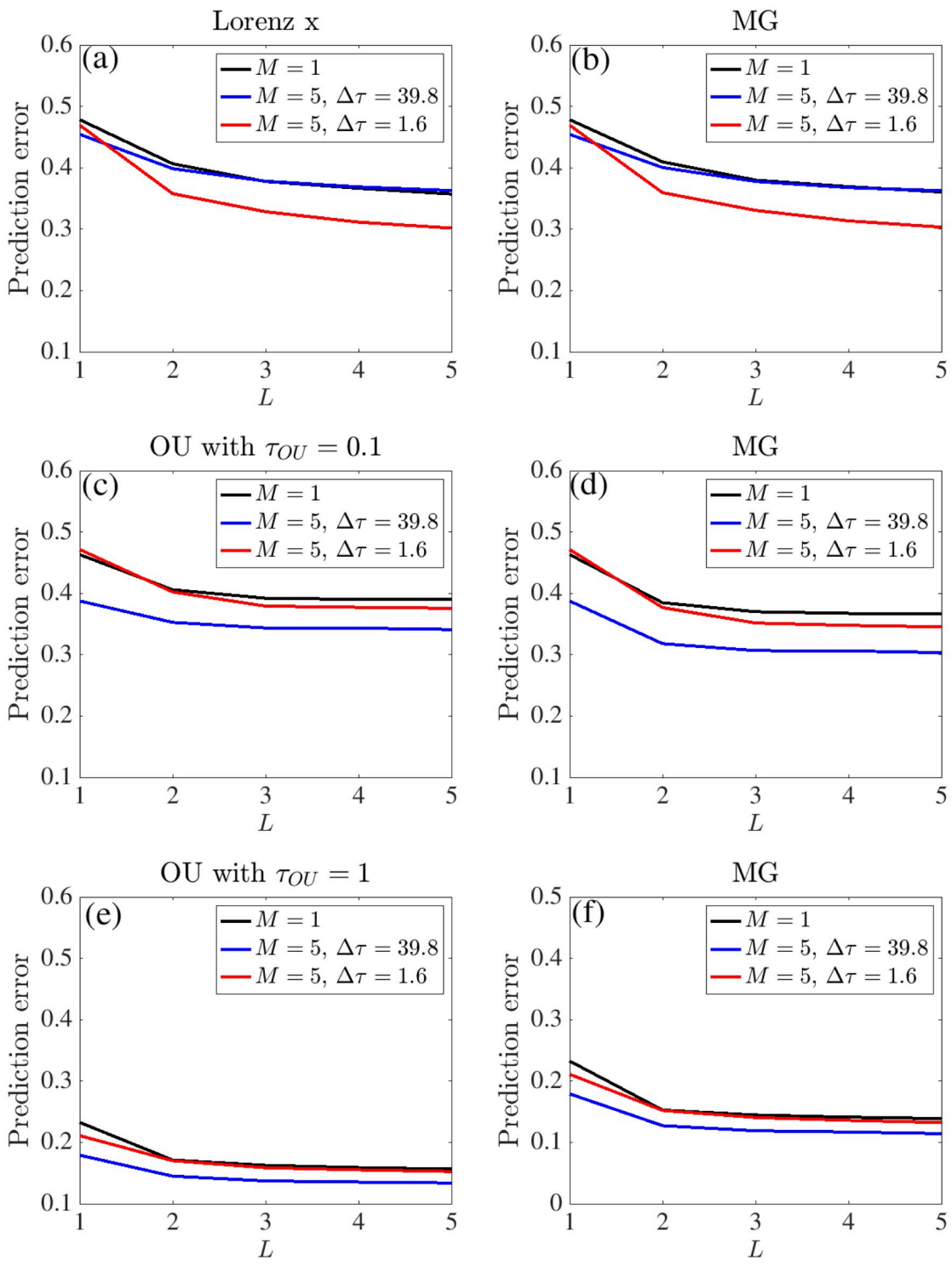

Fig 6. Performance evaluation of time delay reservoir computers at a mixing ratio $s = 0.5$, for different numbers of layers (L). The top row illustrates the error for the mixtures of Lorenz x and MG, while the bottom rows depicts the mixtures of MG and OU process with different τ_{OU}. First column display the prediction error for Lorenz x and OU process, and the second column the prediction error for MG.

https://doi.org/10.1371/journal.pcsy.0000034.g006

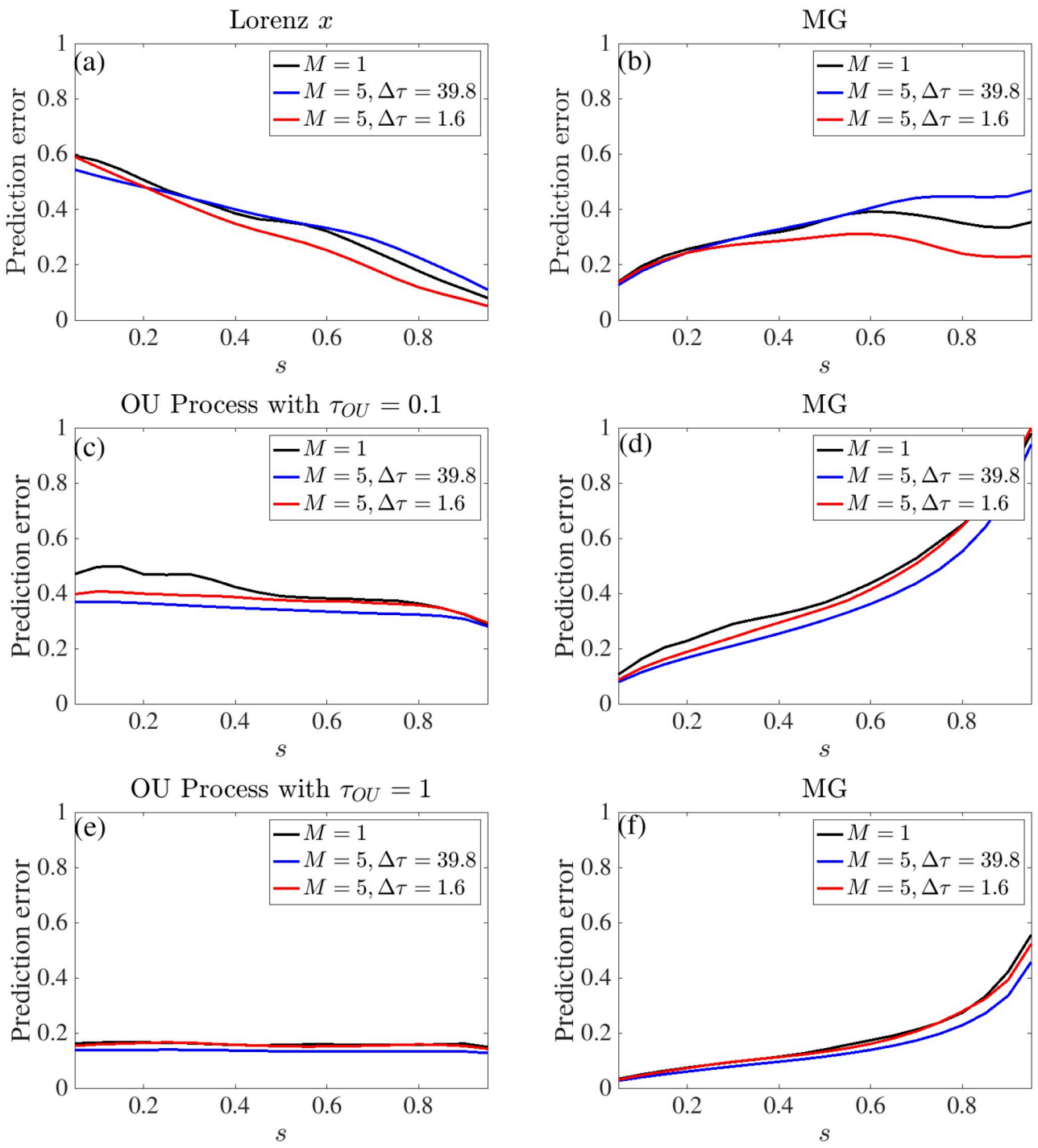

Fig 7. NRMSE for different mixing ratios s with $L = 5$. The top row illustrates the error for mixtures of Lorenz x and MG, while the second and third rows depict mixtures of MG and the OU Process. A large value of s indicates a higher proportion of the Lorenz x or OU process in the mixture, while a smaller value of s signifies a higher proportion of the MG signal in the mixture.

https://doi.org/10.1371/journal.pcsy.0000034.g007

bringing it to an acceptable range. However, in the single delay case, the error remains relatively large. Conversely, for large s, where the noisy time series dominates, adding layers does not substantially improve accuracy, and predicting the MG values becomes challenging, contrasting with the Lorenz x and MG mixture scenario.

However, as τ_{OU} increases, the prediction error markedly decreases, leading to stable predictions across all multi-delay configurations, regardless of the mixing ratio. When the proportion of the second signal u_2 is small (indicative of high s), there is a decrease in the prediction error for the MG time series.

2.3 Effect of bandpass-filtering on predictability

In this section, for the case of a mixture involving a chaotic signal and a signal derived from a stochastic process, we aim to enhance the prediction of the MG signal by applying a band-pass filter to the mixed input before feeding it into the RC equations. The purpose of filtering is to attenuate the stochastic noise and focus the computations on the relevant frequencies of the MG signal. To implement this strategy, we first computed the power spectrum of both signals to identify the dominant frequency ranges for the MG signal and the stochastic signal. The OU process was sampled every 0.002 seconds, whereas the MG time series was sampled every 1 second. To calculate the power spectrum of the signals, we used a frequency resolution corresponding to a sampling rate of 500 Hz. By selecting the appropriate cut-off frequencies based on these power spectra, we can attenuate the unwanted frequencies in the mixed input signal associated with the stochastic component.

As shown in Fig 8, the dominant frequencies of the signal lie between 4 and 20 Hz. To isolate the Mackey-Glass signal, we applied a Butterworth band-pass filter, setting the low cut-off

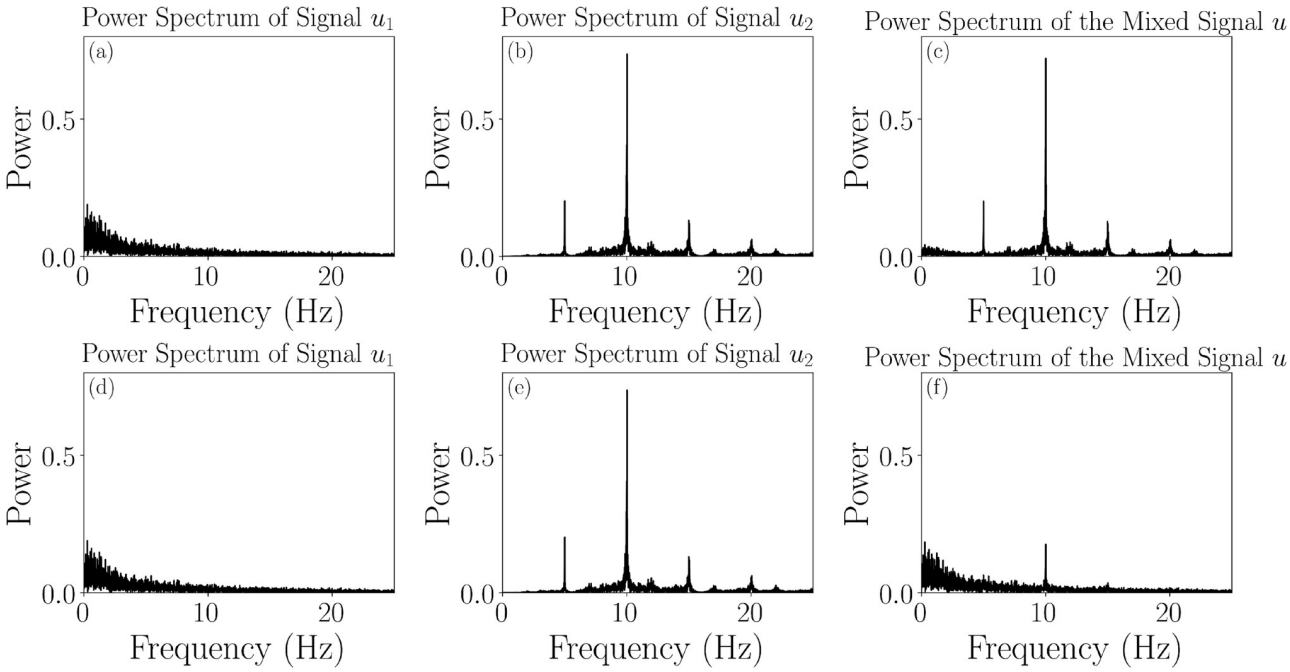

Fig 8. Power spectrum of the (a,d) OU process with $\tau_{OU} = 0.1$, (b,e) MG and (c,f) the mixture of both signals. In the first row, the mixing ratio is set to $s = 0.05$, making the MG signal the dominant component of the mixture. In the second row, $s = 0.95$, making the OU process the dominant component.

https://doi.org/10.1371/journal.pcsy.0000034.g008

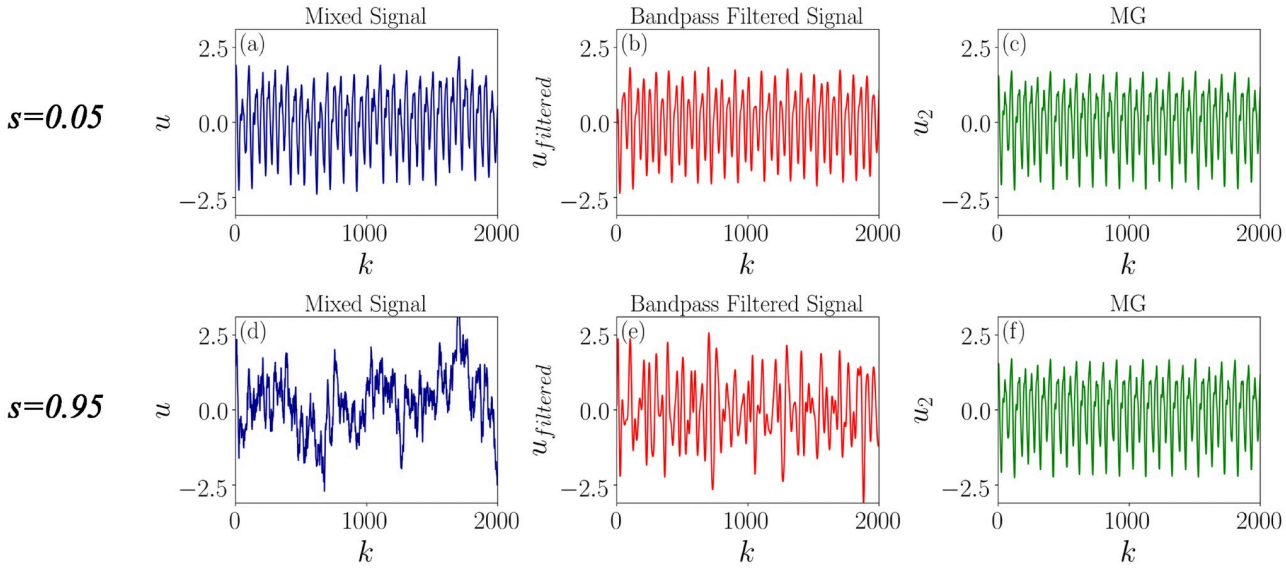

Fig 9. (a,d) Time series for the mixed OU+MG signal (first column) (b,e) The bandpass-filtered version of the mixed signal. (c,f) The MG signal. The mixing ratio s is shown to the left of the rows of panels.

https://doi.org/10.1371/journal.pcsy.0000034.g009

frequency to 4 Hz and the high cut-off frequency to 21 Hz. This filtering process is illustrated in Fig 9.

We then fed this filtered signal into the RC. The comparison of the time-delay RC's performance for the filtered and unfiltered signals is shown in Fig 10. The same parameters used in the previous section were applied, and the experiment was repeated with 36 different mask functions. In all cases, the filtering improved prediction accuracy, with some instances showing enhancements of up to 12%. However, for large values of s, the prediction still falls short of acceptable levels. Notably, the filtering was particularly effective in improving results when the first layer of the RC had a single delay.

3 Discussion

In this paper, we explore the problem of predicting chaotic signals across various architectural configurations. Our examination centers on the prediction errors for both signals when a mixture is injected into the RC, with a specific focus on their effective separation. The challenge of signal separation spans numerous practical applications, from noise-canceling headphones to mechanisms like redundant input cancellation in electric fish. These examples highlight the broad relevance and necessity of effective signal separation techniques in various real-world scenarios.

The context of our study is more specialized to the case of separating two signals that are mixed into one input channel, but not in an unsupervised manner. Instead, the artificial network is trained using a supervised learning approach to predict the future points of each individual signal in a single mixture (similar to using two sources and one microphone in auditory ICA). In some cases, other advanced techniques might be employed to decompose the signal before feeding it into the RC. For example, if one of the components of the mixed signal is noise, a band-pass filter can be applied, allowing supervised learning to be performed on the deterministic signal. In the absence of any information regarding the component signals of the mixture, advanced training methods such as adaptive training, like Kalman filtering, might be used.

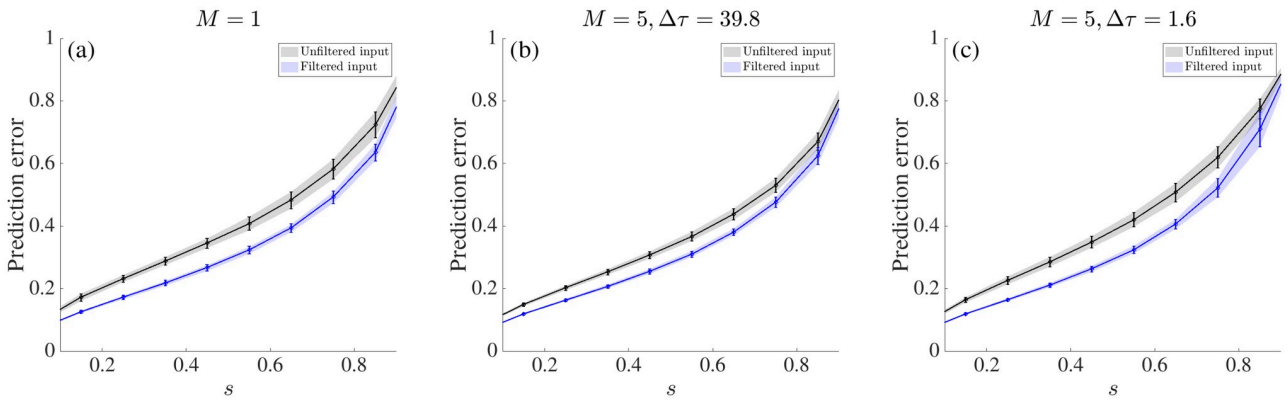

Fig 10. The prediction error for the unfiltered signal (bright black) and the filtered signal (bright blue). (a) RC with 5 layers and a single delay in the first layer. (b) RC with 5 layers and five delays in the first layer with large spacing between delays ($\Delta\tau = 39.8$). (c) RC with 5 layers and 5 delays in the first layer with moderate spacing between delays ($\Delta\tau = 1.6$).

https://doi.org/10.1371/journal.pcsy.0000034.g010

But our RC-based learning algorithm exploits the fact that neighboring trajectories on a chaotic attractor have similar futures in the short term, and thus learns to some degree to reconstruct the dynamics on the attractor that underlies the signal. The prediction will be limited by the usual factors for chaotic systems [36]. For non-chaotic systems in which there is some degree of determinism, one would also expect our algorithm to exploit the predictability that similar trajectory patterns have similar futures in the short term. Here we demonstrate that a second signal with its own chaotic attractor impedes the prediction, but there is nevertheless some level of predictability for both signals. We have also encountered a situation (Fig 4) where this is not the case: a lorenz x signal helps improve the prediction of the MG signal. This effect should be investigated more thoroughly in the future.

In theoretical contexts, signal separation can involve a variety of complex scenarios, including disentangling signals that are linearly superimposed and statistically independent. Techniques such as ICA are crucial in many cases, as they exploit the statistical properties of sources to effectively untangle intricate signal mixtures. The question arises as to whether our multi-delay RC with its mixed input performs the "simultaneous" prediction of both signals. If one defines an RC as having only one core of interacting non-linear elements and one set of output weights, the number of RCs in our system is ambiguous since there is one common core and two sets of output weights. If one chooses the point of view that this amounts to two RCs operating in parallel in spite of using this joint core, then the answer to the question above is that our setup does not perform simultaneous prediction of the two input signals in the mixture. Our approach does share similarities with the multi-functionality strategy of Flynn et al. [37] where two or more tasks are performed simultaneously. It is perhaps more in line with the "conceptor" approach of Jaeger et al. [38] in which a traditional RC is trained to perform different tasks using different sets of trained weights.

Recent findings suggest that the careful selection of time delays is crucial for optimizing the performance of time delay reservoir computing systems. For instance, smaller spacing between delays has been found to enhance task performance when dealing with higher complexity. Conversely, tasks with higher autocorrelation within the datapoints benefit from larger spacing between delays, which tends to augment the system's memory capacity and thus improve prediction accuracy. This paper extends these insights to scenarios involving mixed signals, demonstrating their applicability in predicting components separately. Specifically, in the mixture of the MG and Lorenz x time series, while large spacing between delays proves more effective

for MG alone, moderate spacing is better for Lorenz x. Interestingly, for this mixture, moderate spacing combined with a high feedback coefficient also yields better results.

We acknowledge that optimizing the delays specifically for each mixture combination (s) could potentially yield better performance. However, such optimization would require adjusting feedback coefficients and input scalings for each case, along with conducting a comprehensive set of additional simulations to fully explore these possibilities. It is worth mentioning that this issue is less prominent in the case of multiple delays. With multiple delays, we have a broader range of parameter choices, resulting in less dependency on specific parameters across different ratios. This provides more flexibility and generally leads to better performance across various mixture combinations without the need for extensive optimization for each individual case.

The addition of multiple layers, enhances the predictive performance for the MG series significantly, even when it is not the dominant signal in the mixture. This improvement is particularly noticeable with shorter spacing between delays, highlighting the nuanced interplay between system architecture and signal characteristics.

In real-world scenarios, signals may consist of both deterministic and stochastic components. To test whether it is possible to predict a deterministic signal in the presence of noise, we introduced a linear noisy system. As we introduce a stochastic signal into our analysis, accurately predicting the MG time series becomes notably more challenging. The prediction error associated with a linear noisy system exhibits minimal improvement by addition of layers and time delays, underlining the inherent difficulties in forecasting such time series. When the linear noisy process dominates the mixture, predicting the MG time series, or any other signal, becomes almost impossible, highlighting the complexities of working within mixed signals that include stochastic components. However, this is not always the case; improvements in prediction performance are observed as the correlation time of the linear noise process increases. Nevertheless, a common observation is that the prediction error in multi-delay systems for the OU process remains unaffected by changes in the mixing ratio. From another viewpoint, our analysis with the OU process highlights that the supervised learning of the MG can still proceed with noise, especially for long OU correlation times. This provides insight into why the network can successfully predict and separate two deterministic signals, as it demonstrates robustness in handling noise interference during the learning process.

We have observed how dramatically the addition of layers can influence the performance of the RC, albeit up to a certain limit. Incorporating multiple layers and adding delays in the first layer introduces a complex problem of parameter selection. It is crucial to optimize the parameters in subsequent layers; particularly, we note that aligning the time delay with the clock cycle does not necessarily yield optimal performance. In most instances, adding layers tends to enhance the performance of multi-delay systems more significantly than in single-delay setups. The RC may already possess sufficient memory for the task due to the optimized delays, which could explain the relatively small impact of adding more layers. However, this is not universally true across all tasks. For example, in the first row of Fig 6, we observe that adding layers can increase accuracy by up to 18%, indicating a significant improvement. This suggests that, for certain configurations and tasks, additional layers provide substantial benefits beyond what is achievable by optimizing memory through delay settings alone. The improvement can be attributed to the enhanced processing capabilities and representational power that deeper architectures provide, which can be particularly beneficial when dealing with more complex, non-linear relationships in the input data. The layers in our experiments generally shared common characteristics such as the number of virtual nodes, clock cycle, and phase offset, yet exploring how variations in these parameters might further improve signal separation remains an open question.

4 Methods

In this section, we describe the model used for reservoir computing, followed by a description of the task and the training methodology. We then provide a brief overview of how multi-layered time-delay reservoir computers work and present the dynamical equations corresponding to our different inputs. All numerical simulations were performed in C++, and the Armadillo [39] package was used for training purposes.

4.1 Time delay reservoir computing

The dynamical equations employed for the reservoir computer can be described as follows:

$$\frac{dx}{dt} = -x(t) - \delta y(t) + \frac{\beta}{M}\sum_{i=1}^{M}\sin^2(x(t-\tau_i) + \phi + \gamma J(t)),$$
$$\frac{dy}{dt} = x(t), \tag{1}$$

In this model, β is the gain coefficient, M represents the number of delays, with each τ_i denoting a specific time delay, ϕ is the phase offset, δ is a filter parameter and the input scaling factor γ adjusts the amplitude of the multiplexed masked input $J(t)$, which is a processed and temporal version of the discrete input data. The process of conversion and processing is shown in Fig 11. To convert the discrete input data into temporal data, we first multiplex the signal, as shown in Fig 11(b). Each discrete input data point is held for a duration of one clock cycle T. After multiplexing, the input is multiplied by a binary mask function Mask(t), consisting of a series of −1 or 1.

The reservoir computer comprises N virtual nodes, and therefore, the mask function contains N binary values. The presence of N virtual nodes along a clock-cycle T creates a temporal difference of $\theta = \frac{T}{N}$ between two neighboring nodes. Throughout this paper, we considered $N = 200$ and $T = 40$, resulting in a temporal difference of $\theta = 0.2$. This interval is designed to be smaller than both the time delay and the response time of the system, set at $\tau_R = 1$. These virtual nodes are pivotal in processing the input data, as they are read out and weighted to perform specific tasks. This multiplexed masked input is subsequently fed into the RC, where it drives the RC's dynamics, producing a sequence of transient responses.

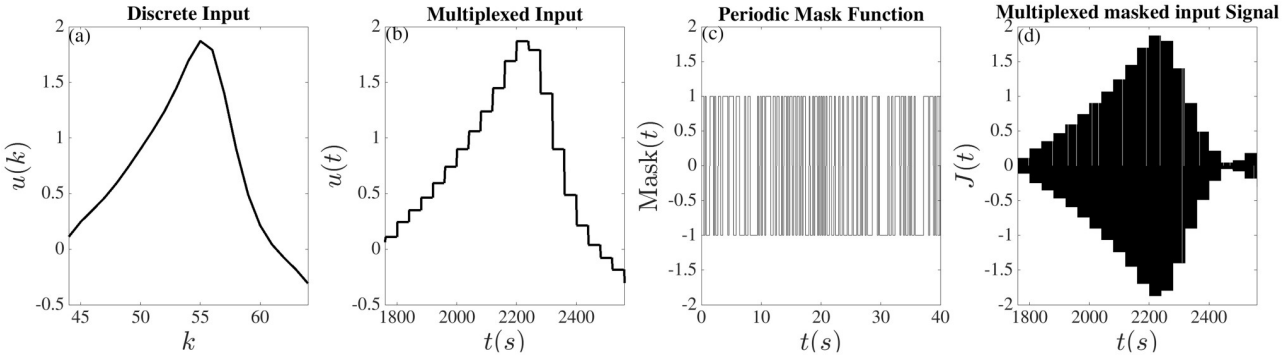

Fig 11. (a) Discrete input data is displayed. (b) The discrete input data is converted into temporal data through multiplexing, where each discrete input data point is held for a clock-cycle $T = 40$. (c) The temporal input data is then multiplied by the mask function. (d) The resulting multiplexed masked input data, denoted $J(t)$.

https://doi.org/10.1371/journal.pcsy.0000034.g011

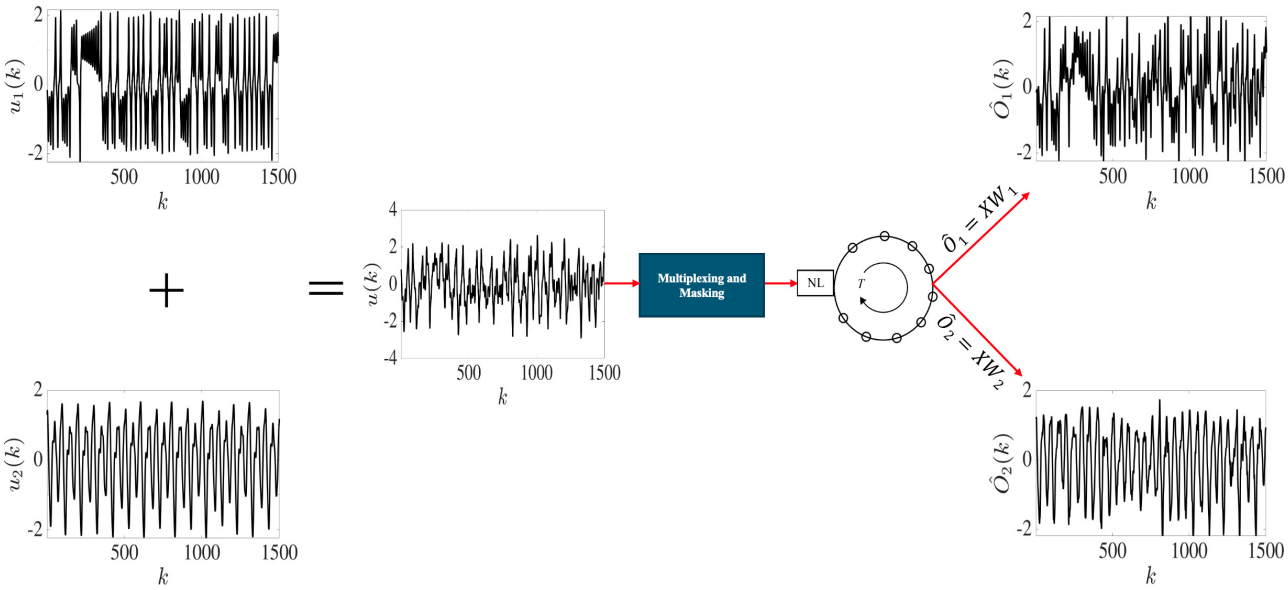

Fig 12. Sketch of de-mixing task for time delay reservoir computing. The mixture of the signals is fed into the reservoir computer and two N-dimensional weight matrices are trained to predict both signals. The resulting network can then be used to predict the evolution of both signals beyond the training sets.

https://doi.org/10.1371/journal.pcsy.0000034.g012

The time delays are picked in the following way:

$$\tau_i = \tau_{min} + (i-1)\Delta\tau, \qquad i = 1, \ldots, M, \tag{2}$$

where τ_{min} is the minimum time delay. For our experiments, the inputs are two distinct normalized signals, u_1 and u_2, each characterized by zero mean and unit variance. The configuration of this setup is depicted in Fig 12. The single combined input u to the RC is defined by the equation:

$$u = \sqrt{s}u_1 + \sqrt{1-s}u_2 \tag{3}$$

This formulation ensures that the mixed input u retains zero mean and unit variance, independent of the mixing ratio s. This approach allows precise control over the input characteristics, facilitating a systematic analysis of the RC's response to varying signal combinations.

By numerically integrating the Eq 1 using the Euler method, we observe that input drives the dynamics by eliciting different transient responses during time. The states of the nodes over the total time of KT are collected into a state matrix X, as illustrated below:

$$X = \begin{bmatrix} x\left(T - \frac{T}{N}(N-1)\right) & x\left(T - \frac{T}{N}(N-2)\right) & \cdots & x(T) \\ \vdots & & \ddots & \vdots \\ x\left(KT - \frac{T}{N}(N-1)\right) & x\left(KT - \frac{T}{N}(N-2)\right) & \cdots & x(KT) \end{bmatrix}_{K\times N}$$

To train the system, we define the target outputs as $O_1(k) = u_1(k+1)$ and $O_2(k) = u_2(k+1)$ where k is the index of a datapoint. We then compute the weight matrices W_1 and W_2 that

map the reservoir states to these targets, as shown below:

$$W_1 = (X^T X + \lambda I)^{-1}(X^T O_1) \tag{4}$$

$$W_2 = (X^T X + \lambda I)^{-1}(X^T O_2) \tag{5}$$

where λ is the regularization parameter which we set it to 10^{-8} [40] and I is the identity matrix. These weight matrices enable the RC to learn and predict the separate components of the mixed input signals, effectively demonstrating the capability of the system in processing complex signals.

The prediction tasks are performed using the weights obtained from the training phase:

$$\hat{O}_1 = XW_1, \quad \hat{O}_2 = XW_2, \tag{6}$$

$\hat{O}_1$ and $\hat{O}_2$ represent the output of the system, which approximates the original signals u_1 and u_2 respectively. By stacking W_1 and W_2 horizontally, we can form a matrix W, which has dimensions $N \times 2$. When this matrix is multiplied by the state matrix X, it estimates the two-dimensional signals that constitute the mixture.

After training, new testing data that are temporally continuous with and independent from the training data are introduced into the reservoir; the states of the virtual nodes are then recorded over time to construct a new state matrix. By multiplying this matrix with the trained weight matrix, the predicted values for the testing phase are obtained. The prediction error is calculated through normalized root mean square error (NRMSE):

$$\text{NRMSE} = \sqrt{\frac{1}{K}\frac{\sum_{k=1}^{K}(O(k) - \hat{O}(k))^2}{\text{VAR}(O)}}. \tag{7}$$

In Eq 7, $O(k)$ represents k-th actual target output data value, $\hat{O}(k)$ is the k-th predicted value, and K is the number of datapoints. We considered 10,000 data points ($K = 10000$) for training, using Eqs 4 and 5. In these equations, the state matrix X has dimensions of 10000×200. For testing, we used 5,000 ($K = 5000$) unseen data points to obtain a new state matrix with dimensions of 5000×200. This matrix is then multiplied by the weight matrix, which has dimensions of 200×1, to get the predicted values one-step ahead in time. The error is measured using Eq 7.

4.2 Multi-layered reservoir computing

In this scheme, each layer contains N virtual nodes and operates with a unique clock-cycle T. The dynamical equations for our multi-layer system are as follows [25]:

$$\begin{aligned} \frac{dx_j}{dt} &= -x_j(t) - \delta y_j(t) + \frac{\beta_j}{M_j}\sum_{i=1}^{M_j}\sin^2(x_j(t-\tau_{ji}) + \phi_j + f_j(t)), \\ \frac{dy_j}{dt} &= x_j(t), \end{aligned} \tag{8}$$

where $j = 1, \ldots, L$ denotes the layer index, with L representing the total number of layers in the system. The first layer receives the initial input, and each subsequent layer is dynamically coupled to the one preceding it. This setup introduces a cascading effect, where the input signal and the outputs from preceding layers interactively influence the dynamics of each subsequent

layer. The function $f_j(t)$, which varies by layer, is defined as follows:

$$f_j(t) = \begin{cases} \gamma J(t) & \text{for } j = 1, \\ \kappa x_{j-1}(t) & \text{for } j > 1. \end{cases}$$

The first layer of the system receives a scaled version of the external input, denoted as $J(t)$, while the outputs of each subsequent layer are driven by the output of the preceding layer, each modified by a coupling coefficient κ. This design enables the propagation and transformation of signals through multiple dynamic layers, significantly enhancing the system's capability to process and predict complex temporal patterns. To effectively analyze these dynamics, we collect the states of the virtual nodes from all layers corresponding to each input data point over time. These states are then stacked to construct the state matrix, X, as shown below:

$$X = \begin{bmatrix} x_1(T - \frac{T}{N}(N-1)) & \dots & x_1(T) & \dots & x_l(T - \frac{T}{N}(N-1)) & \dots & x_l(T) \\ \vdots & & & \vdots & & & \vdots \\ x_1(KT - \frac{T}{N}(N-1)) & \dots & x_1(KT) & \dots & x_l(KT - \frac{KT}{N}(N-1)) & \dots & x_l(KT) \end{bmatrix}_{K \times NL}$$

We have summarized the parameters that lead to improved results in the parameter spaces we tested in the Tables 1–3 for three different scenarios of mixtures: MG and Lorenz x, MG and OU Process with $\tau_{OU} = 0.1$, and MG and OU Process with $\tau_{OU} = 1$, for the mixing ratio of $s = 0.5$.

4.3 Inputs

In this section, we provide the dynamical equations for the input time series. To generate these time series, we used the Runge-Kutta 4 method with a time step of 2×10^{-2} for the MG model and 2×10^{-3} for both the Lorenz model and the OU process. The MG model is described by:

$$\frac{dx}{dt} = -ax(t) + \frac{bx(t - \tau_{MG})}{1 + x^{10}(t - \tau_{MG})}, \tag{9}$$

using standard parameters $a = 0.1$, $b = 0.2$, $\tau_{MG} = 17$. The Lorenz system, describing the second chaotic time series, is given by:

$$\frac{dx}{dt} = \sigma(y(t) - x(t)), \tag{10}$$

$$\frac{dy}{dt} = -x(t)z(t) + \rho x(t) - y(t), \tag{11}$$

$$\frac{dz}{dt} = x(t)y(t) - bz(t), \tag{12}$$

with the paramaters $\sigma = 10$, $\rho = 28$, $b = 8/3$. Lorenz system is sampled every 0.05 second. OU process is described by:

$$\tau_{OU} \frac{dx}{dt} = -x(t) + \zeta(t), \tag{13}$$

where τ_{OU} represents the time constant of the Ornstein-Uhlenbeck process and $\zeta(t)$ is a Gaussian white noise term.

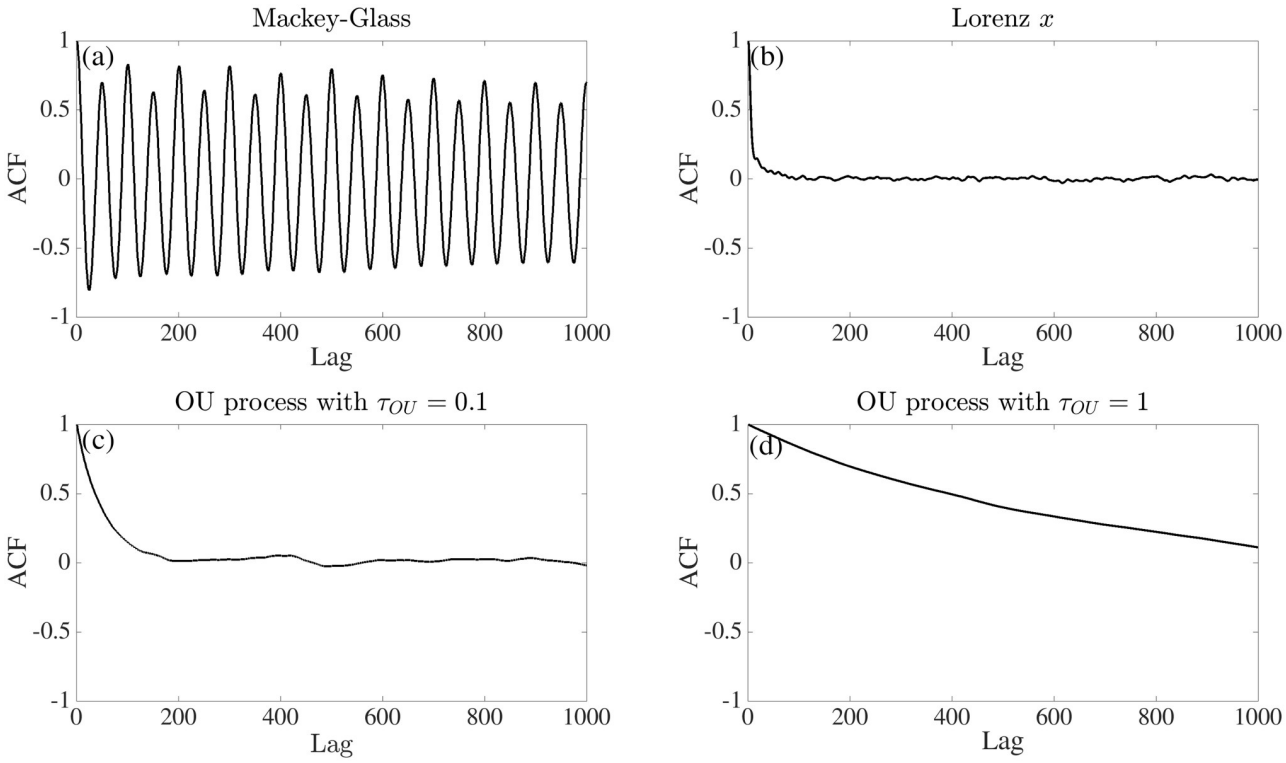

Fig 13. ACF of the input time series. Panel (a) shows the ACF of the Mackey-Glass time series, characterized by regularly spaced but decaying peaks. (b) ACF for the Lorenz x time series. (c) and (d): ACFs for the Ornstein-Uhlenbeck (OU) process with $\tau_{OU} = 0.1$ and $\tau_{OU} = 1$, respectively. A lag value of 1000 corresponds to 1000 seconds in the Mackey-Glass time series, 50 seconds in the Lorenz system, and 2 seconds in the OU process.

https://doi.org/10.1371/journal.pcsy.0000034.g013

The autocorrelation functions (ACFs) of the inputs are illustrated in Fig 13. For the Mackey-Glass time series, there are highly pronounced peaks in the ACF plot. For lorenz x, the correlation time is smaller compared to the Mackey-Glass time series, which makes that prediction task more difficult compared to MG. For both the fast and slow OU processes, the correlation time is larger that the one for the Lorenz x.

Author Contributions

Conceptualization: S. Kamyar Tavakoli, André Longtin.

Formal analysis: S. Kamyar Tavakoli.

Funding acquisition: André Longtin.

Investigation: S. Kamyar Tavakoli.

Software: S. Kamyar Tavakoli.

Supervision: André Longtin.

Visualization: S. Kamyar Tavakoli.

Writing – original draft: S. Kamyar Tavakoli.

Writing – review & editing: S. Kamyar Tavakoli, André Longtin.

References

1. Bell AJ, Sejnowski TJ. An information-maximization approach to blind separation and blind deconvolution. Neural Comput. 1995; 7(6):1129–1159. https://doi.org/10.1162/neco.1995.7.6.1129 PMID: 7584893
2. Molgedey L, Schuster HG. Separation of a mixture of independent signals using time delayed correlations. Phys Rev Lett. 1994; 72:3634–3637. https://doi.org/10.1103/PhysRevLett.72.3634 PMID: 10056251
3. Tong L, Liu Rw, Soon VC, Huang YF. Indeterminacy and identifiability of blind identification. IEEE Transactions on Circuits and Systems. 1991; 38(5):499–509. https://doi.org/10.1109/31.76486
4. Nolte G, Meinecke FC, Ziehe A, Müller KR. Identifying interactions in mixed and noisy complex systems. Phys Rev E. 2006; 73:051913. https://doi.org/10.1103/PhysRevE.73.051913 PMID: 16802973
5. Bol K, Marsat G, Harvey-Girard E, Longtin A, Maler L. Frequency-Tuned cerebellar channels and burst-induced LTD lead to the cancellation of redundant sensory inputs. Journal of Neuroscience. 2011; 31 (30):11028–11038. https://doi.org/10.1523/JNEUROSCI.0193-11.2011 PMID: 21795551
6. Mejías JF, Marsat G, Bol K, Maler L, Longtin A. Learning contrast-invariant cancellation of redundant signals in neural systems. PLoS Comput Biol. 2013; 9(9):e1003180. https://doi.org/10.1371/journal.pcbi.1003180 PMID: 24068898
7. Wallach A, Sawtell NB. An internal model for canceling self-generated sensory input in freely behaving electric fish. Neuron. 2023; 111(16):2570–2582.e5. https://doi.org/10.1016/j.neuron.2023.05.019 PMID: 37321221
8. Comon P. Independent component analysis, A new concept? Signal Processing. 1994; 36(3):287–314.
9. Pearlmutter B, Parra L. Maximum Likelihood Blind Source Separation: A Context-Sensitive Generalization of ICA. In: Mozer MC, Jordan M, Petsche T, editors. Advances in Neural Information Processing Systems. vol. 9. MIT Press; 1996. Available from: https://proceedings.neurips.cc/paper_files/paper/1996/file/dabd8d2ce74e782c65a973ef76fd540b-Paper.pdf.
10. Matsuoka K, Ohoya M, Kawamoto M. A neural net for blind separation of nonstationary signals. Neural Networks. 1995; 8(3):411–419. https://doi.org/10.1016/0893-6080(94)00083-X
11. Luo Y, Chen Z, Yoshioka T. Dual-Path RNN: Efficient Long Sequence Modeling for Time-Domain Single-Channel Speech Separation. In: ICASSP 2020—2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP); 2020. p. 46–50.
12. Subakan C, Ravanelli M, Cornell S, Bronzi M, Zhong J. Attention Is All You Need In Speech Separation. ICASSP 2021—2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2020; p. 21–25.
13. Chen Z, Luo Y, Mesgarani N. Deep attractor network for single-microphone speaker separation. 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2016; p. 246–250.
14. Lu Z, Kim JZ, Bassett DS. Supervised chaotic source separation by a tank of water. Chaos: An Interdisciplinary Journal of Nonlinear Science. 2020; 30(2):021101. https://doi.org/10.1063/1.5142462 PMID: 32113226
15. Krishnagopal S, Girvan M, Ott E, Hunt BR. Separation of chaotic signals by reservoir computing. Chaos: An Interdisciplinary Journal of Nonlinear Science. 2020; 30(2):023123. https://doi.org/10.1063/1.5132766 PMID: 32113243
16. Appeltant L, Soriano MC, Van Der Sande G, Danckaert J, Massar S, Dambre J, et al. Information processing using a single dynamical node as complex system. Nature Communications. 2011; 2(1):1–6. https://doi.org/10.1038/ncomms1476 PMID: 21915110
17. Larger L, Baylón-Fuentes A, Martinenghi R, Udaltsov VS, Chembo YK, Jacquot M. High-Speed Photonic Reservoir Computing Using a Time-Delay-Based Architecture: Million Words per Second Classification. Phys Rev X. 2017; 7:011015.
18. Xu Y, Zhang M, Zhang L, Lu P, Mihailov S, Bao X. Time-delay signature suppression in a chaotic semiconductor laser by fiber random grating induced random distributed feedback. Opt Lett. 2017; 42 (20):4107–4110. https://doi.org/10.1364/OL.42.004107 PMID: 29028024
19. Tavakoli SK, Longtin A. Multi-delay complexity collapse. Phys Rev Research. 2020; 2:033485. https://doi.org/10.1103/PhysRevResearch.2.033485
20. Tavakoli SK, Longtin A. Complexity Collapse, Fluctuating Synchrony, and Transient Chaos in Neural Networks With Delay Clusters. Frontiers in Systems Neuroscience. 2021; 15. https://doi.org/10.3389/fnsys.2021.720744 PMID: 34867219
21. Pyragas K. Control of Chaos via an Unstable Delayed Feedback Controller. Phys Rev Lett. 2001; 86:2265–2268. https://doi.org/10.1103/PhysRevLett.86.2265 PMID: 11289905

22. Kiss G, Röst G. Controlling Mackey–Glass chaos. Chaos: An Interdisciplinary Journal of Nonlinear Science. 2017; 27(11):114321. https://doi.org/10.1063/1.5006922 PMID: 29195319

23. Foss J, Longtin A, Mensour B, Milton J. Multistability and Delayed Recurrent Loops. Phys Rev Lett. 1996; 76:708–711. https://doi.org/10.1103/PhysRevLett.76.708 PMID: 10061527

24. Hou YS, Xia GQ, Jayaprasath E, Yue DZ, Yang WY, Wu ZM. Prediction and classification performance of reservoir computing system using mutually delay-coupled semiconductor lasers. Optics Communications. 2019; 433:215–220. https://doi.org/10.1016/j.optcom.2018.10.014

25. Goldmann M, Köster F, Lüdge K, Yanchuk S. Deep time-delay reservoir computing: Dynamics and memory capacity. Chaos: An Interdisciplinary Journal of Nonlinear Science. 2020; 30(9):093124. https://doi.org/10.1063/5.0017974 PMID: 33003948

26. Goldmann M, Fischer I, Mirasso CR, C Soriano M. Exploiting oscillatory dynamics of delay systems for reservoir computing. Chaos: An Interdisciplinary Journal of Nonlinear Science. 2023; 33(9):093139. https://doi.org/10.1063/5.0156494 PMID: 37748487

27. Kuriki Y, Nakayama J, Takano K, Uchida A. Impact of input mask signals on delay-based photonic reservoir computing with semiconductor lasers. Opt Express. 2018; 26(5):5777–5788. https://doi.org/10.1364/OE.26.005777 PMID: 29529779

28. Hasegawa H, Kanno K, Uchida A. Parallel and deep reservoir computing using semiconductor lasers with optical feedback. Nanophotonics. 2023; 12(5):869–881. https://doi.org/10.1515/nanoph-2022-0440 PMID: 39634361

29. Penkovsky B, Porte X, Jacquot M, Larger L, Brunner D. Coupled nonlinear delay systems as deep convolutional neural networks. Phys Rev Lett. 2019; 123:054101. https://doi.org/10.1103/PhysRevLett.123.054101 PMID: 31491321

30. Bauwens I, Harkhoe K, Bienstman P, Verschaffelt G, der Sande GV. Transfer learning for photonic delay-based reservoir computing to compensate parameter drift. Nanophotonics. 2023; 12(5):949–961. https://doi.org/10.1515/nanoph-2022-0399 PMID: 39634352

31. Jin J, Jiang N, Zhang Y, Feng W, Zhao A, Liu S, et al. Adaptive time-delayed photonic reservoir computing based on Kalman-filter training. Opt Express. 2022; 30(8):13647–13658. https://doi.org/10.1364/OE.454852 PMID: 35472973

32. Tavakoli SK, Longtin A. Boosting reservoir computer performance with multiple delays. Phys Rev E. 2024; 109:054203. https://doi.org/10.1103/PhysRevE.109.054203 PMID: 38907463

33. Jaurigue L, Lüdge K. Reducing reservoir computer hyperparameter dependence by external timescale tailoring. Neuromorphic Computing and Engineering. 2024; 4(1):014001. URL: https://doi.org/https://dx.doi.org/10.1088/2634-4386/ad1d32

34. Martinenghi R, Rybalko S, Jacquot M, Chembo YK, Larger L. Photonic Nonlinear Transient Computing with Multiple-Delay Wavelength Dynamics. Phys Rev Lett. 2012; 108:244101. https://doi.org/10.1103/PhysRevLett.108.244101 PMID: 23004274

35. Engelborghs K, Luzyanina T, Roose D. Numerical bifurcation analysis of delay differential equations using DDE-BIFTOOL. ACM Trans Math Softw. 2002; 28:1–21. https://doi.org/10.1145/513001.513002

36. Farmer JD, Sidorowich JJ. Predicting chaotic time series. Phys Rev Lett. 1987; 59(8):845–848. URL: https://link.aps.org/doi/10.1103/PhysRevLett.59.845. PMID: 10035887

37. Flynn A, Tsachouridis VA, Amann A. Multifunctionality in a reservoir computer. *Chaos: An Interdisciplinary Journal of Nonlinear Science*. 2021; 31(1):013125. https://doi.org/10.1063/5.0019974

38. Jaeger H. Using conceptors to manage neural long-term memories for temporal patterns. *Journal of Machine Learning Research*. 2017; 18(13):1–43. URL: http://jmlr.org/papers/v18/15-449.html

39. Sanderson C, Curtin R. Armadillo: a template-based C++ library for linear algebra. Journal of Open Source Software. 2016; 1(2):26. https://doi.org/10.21105/joss.00026

40. Stelzer F, Röhm A, Lüdge K, Yanchuk S. Performance boost of time-delay reservoir computing by non-resonant clock cycle. Neural Networks. 2020; 124:158–169. https://doi.org/10.1016/j.neunet.2020.01.010 PMID: 32006747