Machine Learning Potential of a Single Pendulum

Swarndu Mandal,1,* Sudeshna Sinha,2,† and Manish Dev Shrimali1,‡

1Central University of Rajasthan, Ajmer, Rajasthan, India - 305817
2Indian Institute of Science Education and Research Mohali, Punjab, India - 140306
Abstract

Reservoir Computing offers a great computational framework where a physical system can directly be used as computational substrate. Typically a “reservoir” is comprised of a large number of dynamical systems, and is consequently high-dimensional. In this work, we use just a single simple low-dimensional dynamical system, namely a driven pendulum, as a potential reservoir to implement reservoir computing. Remarkably we demonstrate, through numerical simulations, as well as a proof-of-principle experimental realization, that one can successfully perform learning tasks using this single system. The underlying idea is to utilize the rich intrinsic dynamical patterns of the driven pendulum, especially the transient dynamics which has so far been an untapped resource. This allows even a single system to serve as a suitable candidate for a “reservoir”. Specifically, we analyze the performance of the single pendulum reservoir for two classes of tasks: temporal and non-temporal data processing. The accuracy and robustness of the performance exhibited by this minimal one-node reservoir in implementing these tasks strongly suggest a new direction in designing the reservoir layer from the point of view of efficient applications. Further, the simplicity of our learning system offers an opportunity to better understand the framework of reservoir computing in general and indicates the remarkable machine learning potential of even a single simple nonlinear system.

The ability of dynamical systems to process information has commanded long-standing interdisciplinary research interest[1–3]. There are several examples of natural systems with the capability to perform different forms of intrinsic computation[4–6]. In the context of machine learning, the overarching question is how ideas from physics or physical systems can enhance existing concepts. On one hand, research directions that can enhance the performance of algorithms in handling data from dynamical systems is a very pertinent question [7–11]. On the other hand, research efforts to utilize physical systems to implement machine-learning learning algorithms have serious implications for new concepts in the field of artificial intelligence. This line of enquiry also has consequences for gauging the information processing capacity of naturally-occurring or human-engineered physical, chemical, and biological systems [12].

Here we consider the Reservoir Computing (RC) technique to exploit a dynamical system for machine learning. RC is a recurrent neural network (RNN) based computational
framework, in which the memory capability and rich dynamics of an RNN can be used for computation without actually training the network structure itself. Instead, training the readout layer is sufficient to achieve good performance [13, 14]. In this framework, the network is called the reservoir, as it stores the input as a high dimensional Spatio-temporal pattern, such that a linear transformation can efficiently extract the desired output in readout. Formally, a low-dimensional temporal input \( u(t) \) is transformed into a much higher dimensional state vector \( x(t) \) by the reservoir. These state vectors are processed further by the linear readout to get a desired output. For its simplicity, scalability and lower training costs, reservoir computing has attracted widespread research interest, both in terms of applications [15–24] as well as basic development of the general framework [25–28].

In this letter, we show through both numerical simulations and experimentation, that a surprisingly simple system, namely a single forced pendulum has sufficient richness in its dynamics to process information for intelligent computation. The central idea is that, instead of multiplexing the input in state space, we encode the inputs in the temporal patterns, effectively making it act like a high dimensional system [29, 30]. In this work, we will assess the performance of our reservoir in the arena of both temporal and non-temporal tasks, and demonstrate that both these classes of tasks can be performed using our minimal one-node reservoir with strikingly good performance.

**Reservoir Dynamics:** Specifically, we consider a pendulum of length \( l \) with a bob of mass \( m \), periodically driven with a force of amplitude \( F \), depicted schematically in Fig. 1. Given a damping coefficient \( b \) of the medium, the equation of motion can be written as

\[
\frac{d^2x}{dt^2} = -\frac{g}{l}\sin(x) - k\frac{dx}{dt} + f \text{ sign} [\sin(\omega t)].
\]

(1)

where the \( \text{sign}[\cdot] \) function represents a square wave that toggles between +1 and −1 when the argument switches from positive to negative. Here \( f = \frac{F}{m} \) and \( k = \frac{b}{m} \) are the amplitude of the force and damping coefficient per unit mass. This system can yield quasiperiodicity (as depicted in Fig. 1) and has been studied extensively [31–33]. With no loss of generality, we have considered \( k = 5 \times 10^{-2} \) and \( l = 1.0 \) for the numerical simulations presented in this study.

The unique dynamics arising at each point in \( f - \omega \) parameter space is evident from the bifurcation diagrams shown in Fig. 2. Unlike most studies, here we examine the temporal
FIG. 1. (a) Schematic of the reservoir; (b) dynamics of an under-damped pendulum without forcing (solid black line) with the period of the driving force shown by dotted red lines; (c) Transient dynamics of the system (solid black line) under periodic forcing (indicated by alternating red and blue backgrounds). In the presence of periodic driving, the dynamics is quasiperiodic here, as the frequency of force is not equal to the natural frequency of the pendulum.

patterns arising, not just in the asymptotic case, but in the transient dynamics as well. The comparative features of the transient reservoir dynamics and the asymptotic dynamics can be seen from the two columns displayed in the figure. Clearly, the transient dynamics provides a richer repertoire of nonlinear patterns than the asymptotic behaviour, and we will crucially use this aspect to encode information more efficiently.

**Input Encoding:** Multiplexing the input signal efficiently into the reservoir dynamics is a crucial step for reservoir computing. The complete information should be stored into the reservoir. For our case, we have three possible choices to feed the input to the system. One option is to encode the input information with the initial condition. But this is not an efficient choice. As different trajectories can evolve to the same attractor, the pre-images are not unique after transience, with different initial conditions producing the same asymptotic trajectories. Hence this will lead to input information loss and hinder robust and consistent input encoding. Alternately, one can encode the input using the two system parameters: amplitude \( f \), and frequency \( \omega \) of the applied force. Either of these two options is a better choice than input encoding with initial conditions, as each point in \( f - \omega \) space gives rise to
FIG. 2. Bifurcation Diagram of the reservoir dynamics, (a-b) with respect to amplitude \( f \) of the driving force, with \( \omega = 1.0 \), and (c-d) with respect to the driving frequency \( \omega \), with \( f = 1.5 \). The first column (a), (c) represents the asymptotic dynamics, and the second column (b), (d) shows the transient dynamics starting from initial state \([x, \dot{x}] = [0, 0]\). The region between dashed red lines are used to encode the input to the reservoir.

an unique dynamical sequence.

In this work we focus on tasks involving one-dimensional inputs, and hence only one parameter needs to be varied for input encoding, while the other parameter can be kept fixed. We will consider input-encoding using both the amplitude and the frequency of forcing, in order to compare the efficacy of these two alternate methods of input-encoding for different tasks, including their robustness in the presence of noise. We will denote the scheme of input-encoding using forcing amplitude \( f \) as amplitude encoding, while the scheme where inputs are encoded using the forcing frequency \( \omega \) will be simply referred to as frequency encoding.

First consider the amplitude encoding scheme where we multiplex the input with the amplitude of force \( f \). In this scheme we need to choose a range of the parameter, say \( f \in [f_{\text{min}}, f_{\text{max}}] \), and we then need to scale all input points into this range. This scaling transformation \((u \rightarrow f)\) can be expressed as \( f = f_{\text{min}} + (f_{\text{max}} - f_{\text{min}})u \), where, \( u \) is the normalized input in range \([0, 1]\). Formally, \( u = \frac{\min[\tilde{u}(t)] + \tilde{u}(t) - \min[\tilde{u}(t)]}{\max[\tilde{u}(t)] - \min[\tilde{u}(t)]}, \) \( \tilde{u}(t) \) being the original input signal. Specifically, the range of \( f \) for this scheme is taken to be \([1, 2]\) as shown in Fig. 2(b). For the frequency encoding scheme, we can proceed in a similar fashion, with parameter \( \omega \) replacing parameter \( f \) in the formalism.
As a test-bed to gauge the performance of our system we will consider two distinct classes of tasks: one task will involve processing non-temporal signals and another task will consider processing temporal signals. In general, Reservoir Computing has proven to be successful in solving time dependent data processing, stemming from the nonlinear memory effect of the reservoir. But for non-temporal tasks, we need to remove the memory effect. This can be achieved by resetting the reservoir to a fixed point after feeding an input data point to it. Specifically for our case, the reservoir is set to $[x, \dot{x}] = [0, 0]$ after each input.

**Reservoir State and Regression:** The transient dynamics of the reservoir is stored in a discretized form as the state vector. Only the transient part of the dynamics is considered as it produces richer nonlinear repertoire than its asymptotic behavior. More formally, the state vector is the set of variables $x(t)$ recorded at a fixed sampling rate $\kappa \Omega [\kappa = 1, 2, 3...], \kappa$ an integer multiple of the sampling cycle frequency $\Omega$, i.e. for each sampling cycle we record $\kappa$ values of $x(t)$. Thus if we store the data for $N$ cycles for each input, states to be stored can be written as $S = [x(0), \ x(\tau), \ x(2\tau), \ ...., \ x(\kappa N \tau)]$, where $\tau = \frac{2\pi}{\kappa \Omega}$ is the sampling interval. For each point $\tilde{u}(t_i), i = 1, 2, 3, ...$ of input signal, one state vector $X_i$ is formed from $S$.

Now, forming the reservoir state vector $X_i$ from stored states $S$ is different for temporal and non-temporal tasks. For non-temporal tasks, we produce the state vector as the column matrix $X_i = [S_i]^T$, where $[\cdot]^T$ represents the transpose. For temporal tasks, we form the state vector corresponding to any particular input with the current states as well as states corresponding to certain number of previous states, i.e. we take $X_i = [w_0 S_{i-m}, \ w_1 S_{i-m-1}, \ ......., \ w_{m-1} S_{i-1}, \ w_m S_i]^T$, where $w_j, j = 0, 1, 2..., m$ are the weights of previous input states, following a linear distribution in the range $[0, 1]$. Here $m$ is the finite memory, which can be considered as a hyper-parameter to be tuned for different kinds of temporal tasks, allows us to achieve the required fading memory to process temporal data. In our numerical simulations, we have considered $m = 100$.

For the two schemes of input encoding the value of $\Omega$ is different. In the amplitude encoding scheme we take $\Omega = \omega$, the frequency of driving force and for the frequency encoding scheme we consider $\Omega = \omega_0$, the natural frequency of the oscillator. In general, $\Omega$ can be treated as another hyper-parameter for both schemes.

Thus, for a complete input signal $\tilde{u}(t)$ one has the reservoir state vector matrix $\mathcal{R} = [X_1, \ X_2, \ X_3, ..., X_L], L$ being the length of input signal. So the matrix $\mathcal{R}$ has the dimension $\kappa N \times L$ for Task - I and $m \kappa N \times L$ for Task-II, where $\kappa$ and $N$ are hyper-parameters that
can be optimized for best results.

Now if the corresponding output for ˜u(t) is ˜v(t), the linear transformation between the output signal and the reservoir state vector matrix can be written as ˜v = Wℜ, where, W is the 1 × κN dimensional connection matrix. This matrix can be evaluated using training data set by regression method as W = ˜vℜ⁻¹. Specifically for this purpose, we have used the Moore-Penrose pseudoinverse [34].

**Machine Learning Tasks:** To check the performance of the reservoir we consider two tasks. The first task is non-temporal, and involves the learning of a high-degree polynomial. The second task involves temporal data processing, and considers the difficult task of using data from one state variable to infer another state variable in a chaotic system.

Specifically, the aim of Task - I is to approximate a 7th degree polynomial given by 
\[ f(x) = (x - 3)(x - 2)(x - 1)x(x + 1)(x + 2)(x + 3) \]
in the range \( x \in [-3, 3] \). As this task corresponds to non-temporal input processing, one input point \( x \) is necessary and sufficient to get the corresponding output \( f(x) \).

Our second task (Task - II) pertains to the reconstruction of a chaotic attractor dealing with temporal data processing. As an illustrative example, here we consider the state variable \( x(t) \) of a chaotic Lorenz system \[ \dot{x} = 10(y - x), \quad \dot{y} = x(28 - z) - y, \quad \dot{z} = xy - \frac{8}{3}z \]
as input to infer another state variable \( y(t) \) in output [35, 36].

**Results:** The efficiency of the reservoir is analysed by estimating the accuracy of the tasks it performs, quantified by root mean square error (RMSE) of the predicted output with reference to the target one. We find that the reservoir works with great accuracy for both temporal and non-temporal tasks, for both the schemes, as is clearly discernible from Fig. 3. The success of our single-node reservoir is also evident quantitatively from Table I, which lists the order of accuracy obtained for the tasks.

For Task-I, we find that a reservoir trained with only 500 data points, can approximate the polynomial with RMSE of the order of \( 10^{-10} \). Further, smaller training data sets does not significantly degrade the accuracy obtained. For instance, even training data sets with size as low as 100, yields accuracy of the order of \( 10^{-6} \).

For the case of Task-II, the reservoir was trained with data set of length 5000, and it yielded an accuracy of the order of \( 10^{-3} \). For the tasks we have considered, the prediction accuracy of the trained reservoir is independent of the testing data length, and for the
FIG. 3. (a) The comparison of predicted output with target for Task-I (a - b) and Task-II (c - d). (a), (c) are the results obtained with the amplitude encoding scheme and (b), (d) are those obtained with frequency encoding.

numerical results listed in Table I we have taken the testing data length to be the same as the training data length. So from the results it is clear that even for the tasks involving intensive and complex information processing, the one-node reservoir predicts the output successfully.

Performance in Presence of Noise: We now assess the robustness of the performance in the presence of a noise floor. In order to examine the effect of noise on the performance, we have perturbed each state variable with a random noise, uniformly distributed in the range \([-0.01 : 0.01]\). The results are displayed in Table I. It is clear that the performance is reasonably stable even in the presence of such significantly large noise. Further, we notice that encoding inputs via the frequency of the drive is more robust and accurate than encoding inputs via the amplitude of forcing. This suggests that for optimal and most robust implementation, different control parameters for encoding information should be investigated, as the nature of the dynamics could be quite different under variation of different parameters, leading to different robustness in the presence of noise.

Proof-of-principle experiment: We have also investigated the performance of a single-node reservoir that utilizes actual laboratory data from an experimental realization of a
| Input encoding | Task - I          | Task - II         |
|----------------|-------------------|-------------------|
| $f$            | $10^{-10} / 10^{-2}$ | $10^{-3} / 10^{-3}$ |
| $\omega$      | $10^{-8} / 10^{-3}$ | $10^{-3} / 10^{-3}$ |

TABLE I. Comparison of performance, as quantified by RMSE, under two different schemes of input encoding, with one method using the forcing amplitude $f$ and the other method using forcing frequency $\omega$ to encode inputs. The first value reports the order of the RMSE obtained from noise-free systems, while the second value (marked in red) gives the results obtained in the presence of noise.

![Graph](image.png)

FIG. 4. Accuracy of Task-I, using the time series of a laboratory realization of a pendulum as a reservoir, demonstrating the ability of a simple experimental system to execute computational tasks. Here the *frequency encoding* scheme is used.

forced pendulum. Remarkably, even this simple experimental system yields very good performance, as seen from the results displayed in Fig. 4. The detailed discussion of experimental setup and procedure is listed in appendix B of the supplementary material.

Comparison of performance with multi-node reservoirs: We have also compared the efficiency of our single-node reservoir with the multi-node reservoirs utilized in earlier studies. As a representative example, we have considered the reservoir of environmentally coupled Lorenz oscillator network [37] to show the comparison in terms of the similar tasks performed by the two reservoirs. The tasks considered are the attractor reconstruction of chaotic Rössler and chaotic Chua systems, and the filtering of a Mackey-Glass time series.
TABLE II. Comparison of accuracy obtained from Reservoir Computing implemented by a multi-node network reservoir and a single-node pendulum reservoir, for three tasks: (I) attractor reconstruction of a chaotic Rössler system, (II) attractor reconstruction of a chaotic Chua system, and (III) filtering of a Mackey-Glass time series.

| Reservoir            | Task - I | Task - II | Task - III |
|----------------------|----------|-----------|------------|
| Multi-node Network   | $10^{-9}$| $10^{-6}$ | $10^{-4}$  |
| Single Pendulum      | $10^{-14}$| $10^{-6}$ | $10^{-11}$|

Table II lists the accuracy obtained for these tasks performed by both the reservoirs when trained with same training data. These results suggest that our single-node reservoir has the potential to perform better than a reservoir comprised of a large network of dynamical systems.

**Duffing Oscillator:** In order to explore the generality of our results we have also investigated another low-dimensional nonlinear system that can be readily implemented in the laboratory, the Duffing oscillator. Detailed results demonstrating the successful implementation of reservoir computing with a single Duffing oscillator in different dynamical regimes, ranging from periodic and quasi-periodic to chaotic, can be found in appendix A of the supplementary material. Further these results offer us a test-bed for gauging the comparative performance of systems with different dynamical complexity serving as a single-node reservoir. The crucial feature we exploit here is that transient periodic and quasi-periodic behaviour offers a rich repertoire of temporal sequences, while not suffering from the extreme sensitivity to initial conditions that comes alongside the complexity of chaos. So we find that the combination of stability and complexity offered by periodic and quasi-periodic transient dynamics makes this class of dynamical behaviour most suited as a reservoir.

**Conclusions:** In summary, we have successfully demonstrated that a *single simple dynamical system*, such as a pendulum, can be used effectively as a reservoir in Reservoir Computing. Specifically, we exploited the rich dynamics of a driven pendulum for a single-node reservoir to perform complex artificial intelligence tasks. To the best of our knowledge, a single simple pendulum working as an efficient and powerful reservoir is being reported for the first time.

In this study, we have undertaken two classes of tasks, one processing temporal signals,
and the other non-temporal inputs. One of the new directions our work suggests is the use of the transient dynamics of non-chaotic nonlinear systems as the “reservoir” in single-node reservoir computing, as it offers both stability and complexity. The temporal patterns embedded in the transient dynamics of a nonlinear system can thus provide a rich set of transformations for the readout layer. We present results from numerical simulations, with the parameters of the dynamical system utilized as reservoir chosen in such a way that it can be easily realized in laboratory experiments.

Importantly, this work can also be extended to deal with noisy real-world data sets containing impurities. Further, physical implementations of the idea can be potentially extended to much smaller, faster, and power-efficient systems, for instance, dynamical systems realized with integrated circuit chips. So these ideas can lead to the foundation of powerful machine-learning enabled chips.

In conclusion then, we have demonstrated that a single low-dimensional nonlinear dynamical system has remarkable potential for information processing, and can serve as a “reservoir” for Reservoir Computing. These results also open up the possibility of other simple dynamical systems for single-node Reservoir Computing, and a wide variety of natural systems can be considered as potential candidates for the reservoir. Thus this work provides a significant step forward towards the broad goal of exploiting intrinsic dynamics of natural systems for intelligent computation.

* swarnenduphy35@gmail.com
† sudeshna@iisermohali.ac.in
‡ shrimali@curaj.ac.in

[1] R. Shaw, Strange attractors, chaotic behavior, and information flow, Zeitschrift für Naturforschung A 36, 80 (1981).
[2] S. Sinha and W. L. Ditto, Dynamics based computation, Physical Review Letters 81, 2156 (1998).
[3] T. Munakata, S. Sinha, and W. Ditto, Chaos computing: Implementation of fundamental logical and arithmetic operations and memory by chaotic elements, IEEE Transactions on Circuits and Systems 49, 1629 (2002).
[4] J. P. Crutchfield, W. L. Ditto, and S. Sinha, Introduction to focus issue: intrinsic and designed computation: information processing in dynamical systems—beyond the digital hegemony (2010).

[5] K. Mainzer, Thinking in complexity: The computational dynamics of matter, mind, and mankind (Springer Science & Business Media, 2007).

[6] T. Toffoli, Nothing makes sense in computing except in the light of evolution., Int. J. Unconv. Comput. 1, 3 (2004).

[7] A. Choudhary, J. F. Lindner, E. G. Holliday, S. T. Miller, S. Sinha, and W. L. Ditto, Physics-enhanced neural networks learn order and chaos, Phys. Rev. E 101, 062207 (2020).

[8] S. Miller, J. Lindner, A. Choudhary, S. Sinha, and W. Ditto, The scaling of physics-informed machine learning with data and dimensions, Chaos, Solitons & Fractals: X 100046 (2020).

[9] A. Choudhary, J. Lindner, E. Holliday, S. Miller, S. Sinha, and W. Ditto, Forecasting hamiltonian dynamics without canonical coordinates, Nonlinear Dynamics 103, 1553 (2021).

[10] C.-D. Han, B. Glaz, M. Haile, and Y.-C. Lai, Adaptable hamiltonian neural networks, Phys. Rev. Research 3, 023156 (2021).

[11] J. Meiyazhagan, S. Sudharsan, and M. Senthilvelan, Model-free prediction of emergence of extreme events in a parametrically driven nonlinear dynamical system by deep learning, Eur. Phys. J. B 94, 156 (2021).

[12] D. Beniaguev, I. Segev, and M. London, Single cortical neurons as deep artificial neural networks, Neuron 109, 2727 (2021).

[13] 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).

[14] W. Maass, T. Natschläger, and H. Markram, Real-time computing without stable states: A new framework for neural computation based on perturbations, Neural computation 14, 2531 (2002).

[15] M. Lukoševičius and H. Jaeger, Reservoir computing approaches to recurrent neural network training, Computer Science Review 3, 127 (2009).

[16] M. Lukoševičius, H. Jaeger, and B. Schrauwen, Reservoir computing trends, KI-Künstliche Intelligenz 26, 365 (2012).

[17] G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata,
D. Nakano, and A. Hirose, Recent advances in physical reservoir computing: A review, Neural Networks 115, 100 (2019).

[18] K. Nakajima, Physical reservoir computing—an introductory perspective, Japanese Journal of Applied Physics 59, 060501 (2020).

[19] J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott, Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach, Physical review letters 120, 024102 (2018).

[20] Y. Zhong, J. Tang, X. Li, B. Gao, H. Qian, and H. Wu, Dynamic memristor-based reservoir computing for high-efficiency temporal signal processing, Nature Communications 12, 1 (2021).

[21] M. Rafayelyan, J. Dong, Y. Tan, F. Krzakala, and S. Gigan, Large-scale optical reservoir computing for spatiotemporal chaotic systems prediction, Physical Review X 10, 041037 (2020).

[22] S. Ghosh, A. Senapati, A. Mishra, J. Chattopadhyay, S. K. Dana, C. Hens, and D. Ghosh, Reservoir computing on epidemic spreading: A case study on covid-19 cases, Physical Review E 104, 014308 (2021).

[23] S. Saha, A. Mishra, S. Ghosh, S. K. Dana, and C. Hens, Predicting bursting in a complete graph of mixed population through reservoir computing, Physical Review Research 2, 033338 (2020).

[24] H. Zhang, H. Fan, L. Wang, and X. Wang, Learning hamiltonian dynamics with reservoir computing, Physical Review E 104, 024205 (2021).

[25] P. R. Vlachas, J. Pathak, B. R. Hunt, T. P. Sapsis, M. Girvan, E. Ott, and P. Koumoutsakos, Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics, Neural Networks 126, 191 (2020).

[26] T. L. Carroll and L. M. Pecora, Network structure effects in reservoir computers, Chaos: An Interdisciplinary Journal of Nonlinear Science 29, 083130 (2019).

[27] N. A. Silva, T. D. Ferreira, and A. Guerreiro, Reservoir computing with solitons, New Journal of Physics 23, 023013 (2021).

[28] T. L. Carroll, Do reservoir computers work best at the edge of chaos?, Chaos: An Interdisciplinary Journal of Nonlinear Science 30, 121109 (2020).

[29] J. H. Jensen and G. Tufte, Reservoir computing with a chaotic circuit, in Artificial Life Conference Proceedings 14 (MIT Press, 2017) pp. 222–229.
[30] 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, Nature communications 2, 1 (2011).

[31] F. J. Romeiras and E. Ott, Strange nonchaotic attractors of the damped pendulum with quasiperiodic forcing, Phys. Rev. A 35, 4404 (1987).

[32] A. Bondeson, E. Ott, and T. M. Antonsen, Quasiperiodically forced damped pendula and schrödinger equations with quasiperiodic potentials: Implications of their equivalence, Phys. Rev. Lett. 55, 2103 (1985).

[33] M. Ding, C. Grebogi, and E. Ott, Evolution of attractors in quasiperiodically forced systems: From quasiperiodic to strange nonchaotic to chaotic, Phys. Rev. A 39, 2593 (1989).

[34] J. C. A. Barata and M. S. Hussein, The moore–penrose pseudoinverse: A tutorial review of the theory, Brazilian Journal of Physics 42, 146 (2012).

[35] E. N. Lorenz, Deterministic nonperiodic flow, Journal of atmospheric sciences 20, 130 (1963).

[36] C. Sparrow, The Lorenz equations: bifurcations, chaos, and strange attractors, Vol. 41 (Springer Science & Business Media, 2012).

[37] S. Mandal and M. D. Shrimali, Achieving criticality for reservoir computing using environment-induced explosive death, Chaos: An Interdisciplinary Journal of Nonlinear Science 31, 031101 (2021).
Appendix

DUFFING OSCILLATOR

In order to explore the generality of our results we have investigated another low-dimensional nonlinear system, that can be readily implemented in the laboratory. We have also chosen this system as it offers us a test-bed for gauging the comparative performance of systems with different dynamical complexity. Specifically we implement Reservoir Computing with a single Duffing oscillator serving as a “reservoir”. The dynamics is given by the evolution equations:

\[ \dot{x} = y \]
\[ \dot{y} = -\delta y - \beta x - \alpha x^3 + f \cos(\omega t) \]

with parameter set chosen as follows: \( \delta = 0.2, \beta = -1.0, \alpha = 1.0, \omega = 1.0 \).

| Periodic | Quasi-periodic | Chaotic |
|----------|---------------|---------|
| Task-I   | Task-II       | Task-I  | Task-II | Task-I | Task-II |
| Asymptotic | \( 2 \times 10^{-7} \) | \( 3 \times 10^{-3} \) | \( 1 \times 10^{-7} \) | \( 1 \times 10^{-2} \) | \( 1 \times 10^{0} \) | \( 2 \times 10^{0} \) |
| Transient | \( 6 \times 10^{-12} \) | \( 2 \times 10^{-3} \) | \( 2 \times 10^{-8} \) | \( 1 \times 10^{-3} \) | \( 1 \times 10^{-8} \) | \( 7 \times 10^{-3} \) |

TABLE III. Comparison of performance quantified by Root Mean Square Error (RMSE), obtained by Reservoir Computing implemented using a single periodic, quasi-periodic and chaotic Duffing oscillator as a reservoir. The first row presents results obtained using asymptotic dynamics, and the second row presents results obtained using transient dynamics.

Table III shows the comparison of the accuracy obtained for two classes of tasks (see main text for details), with the non-temporal task labelled as Task-I and the temporal task labelled as Task-II. Single oscillators with different levels of dynamical complexity are used as the reservoir, ranging from periodic and quasi-periodic, to chaotic. Further we also display the results from Reservoir Computing implemented utilizing asymptotic dynamics shown in the first row, and the results using transient dynamics shown in the second row. It
FIG. 5. (a) Bifurcation diagram of the asymptotic dynamics of the Duffing oscillator. Panels in the bottom row (b - d) display the bifurcation diagram for the transient counterpart in the periodic, chaotic and quasi-periodic regions respectively, as marked in (a) by dashed rectangles.

is clear that a periodic or quasi-periodic dynamical system serves as a better reservoir in this one-node Reservoir Computing framework, than a chaotic system, where the information loss due to the intrinsic chaos leads to uncertainty in the encoding of inputs. Importantly, it is clearly evident that transient dynamics serves as a better reservoir in Reservoir Computing, with \textit{all} systems, ranging from periodic to chaotic performing significantly better when the transient dynamics is utilized. The markedly superior potential of transient dynamics, in comparison to asymptotic dynamics, in tackling computing tasks has not been exploited in earlier work, and presents a new direction for enhanced performance of machine learning
EXPERIMENT

Components

The components used for this experiment are listed and described below.

1. A hollow aluminium rod of length 50 cm and cross-sectional diameter of 1 cm, one end attached to a rigid platform to hang from by a pivot. A bob is attached to the other end.

2. A cylindrical bob of length 6 cm and cross-sectional diameter 4.5 cm, holds two opposite facing propellers aligned in the plane of oscillation. This also contains the control unit of propellers inside with some added weight. The total weight of the bob is 0.5 Kg approximately.

3. Two A2212/13T (1000KV) brush-less DC motor is attached to the bob with two 10 inch (1045) propellers each.

4. Two 30A electronic speed controller (ESC), kept inside the bob, is used for controlling the speed of two motors by a micro-controller.

5. A micro-controller (*arduino nano*) attached with bob, is used to receive wireless data and passing it to ESC. This is actually the part of the circuit responsible for generating the driving force function.

6. *HC 05* Bluetooth module enables the possibility to receive control input wirelessly without effecting the natural dynamics.

7. *MPU − 6050GY − 521* gyro sensor is attached to the pendulum near the pivot to collect the angular deflection data of the pendulum with time. This sensor is interfaced with another micro-controller via narrow flexible wires.

8. *Arduino Uno* is used to received and decode the gyro sensor data and to communicate it with the computer for calculation.
9. An external DC power supply is used to power the whole setup.

**Experimental steps**

*Step - I:* We start by setting up the system with all the components and required circuit connections.

*Step - II:* First, we need to calibrate two BLDC motors. The controllers (ESCs) take a high frequency square wave signal input and according to its pulse width the speed is decided. So, by the micro-controller we need to generate pulse width modulated (PWM) signal specifying the width in range [0,180]. Basically, 0 corresponds to no rotation and 180 to full speed. Now, The speed decides the amount of force exerted ($F$) by the propellers. We need to find out a relation between $f = F/m$ and the input pulse width for PWM.

This is done using a simple setup. Say, for pulse width $p$ a propeller generates a force $F$ and due to that the pendulum rests at an angular displacement $\theta$. In this case, $F = mg \sin(\theta)$ or $f = g \sin(\theta)$. Hence, for any $p$ we can find the value of $f$. Repeating this process with both the motors sufficient number of times with the setup, and fitting the data
to straight lines, we can find calibration curves for any value of $f$ for the two motors.

**Step - III:** We need to program the micro-controllers according to the requirement of operation. Arduino nano, attached to the bob should receive wireless signal of the ESC inputs and the $\omega$ value to generate the driving force function. Similarly, Arduino Uno, interfacing the gyro sensor should be programmed to sample data at rate defined by $\tau$.

**Step - IV:** Running the setup with required inputs and storing the gyro sensor data from Arduino Uno. Reservoir states are generated for both training and testing data inputs.

**Step - V:** Training reservoir states are used for regression with their corresponding output.

**Step - VI:** Using the optimal output weight evaluated by regression, test reservoir states are used to find the output, and the predicted output is compared with target output.

**Drop in performance with the amplitude encoding scheme:**

Fig. 4 of the letter shows the results from this experiment using the frequency encoding scheme. However, there is a significant effect of noise on the performance when the amplitude encoding scheme is used. The factors that affect the performance under the amplitude encoding scheme are rationalized as follows. To use amplitude encoding scheme one need to multiplex the input with the amplitude of force. In the experimental setup, we can control only the pulse width of the ESC input, and two transformations need to be implemented. First, the pulse width information is converted into speed and secondly, according to varying speeds different magnitudes of the reaction force generated by the propellers is exerted on the system. So there are many potential factors affecting the control of input force, such as the electronic or thermal noise effecting the ESCs and air density, ambient temperature, environment’s aerodynamics with many other. So there is no direct control over the forcing amplitude, i.e. the value of $f$. On the other hand, the frequency encoding scheme simply encodes input information using the frequency of the force $\omega$, on which there is a direct control. Since the frequency can be controlled with a precision of $\sim (\mu s)^{-1}$ by the microcontroller, the frequency-encoding scheme yields better results.