Bridging known and unknown dynamics: machine learning inference from sparse observations

Zheng-Meng Zhai[1], Benjamin D. Stern[2], Ying-Cheng Lai[1,3*]

[1*]School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ, 85287, USA.
[2]Doctor of Physical Therapy Program, Tufts University School of Medicine, 101 E Washington St Suite 950, Phoenix, AZ, 85004, USA.
[3]Department of Physics, Arizona State University, Tempe, AZ, 85287, USA.

*Corresponding author(s). E-mail(s): Ying-Cheng.Lai@asu.edu;

Abstract

In applications, an anticipated issue is where the system of interest has never been encountered before and sparse observations can be made only once. Can the dynamics be faithfully reconstructed? We address this challenge by developing a hybrid transformer and reservoir-computing scheme. The transformer is trained without using data from the target system, but with essentially unlimited synthetic data from known chaotic systems. The trained transformer is then tested with the sparse data from the target system, and its output is further fed into a reservoir computer for predicting its long-term dynamics or the attractor. The power of the proposed hybrid machine-learning framework is demonstrated using various prototypical nonlinear systems, where the dynamics can be faithfully reconstructed even with a high degree of sparsity. The framework provides a paradigm of reconstructing complex and nonlinear dynamics in the extreme situation where training data do not exist and the observations are random and sparse.

In applications of complex systems, observations are fundamental to tasks such as mechanistic understanding, dynamics reconstruction, state prediction, and control. When the available data are complete in the sense that the data points are sampled according to the Nyquist criterion and no points are missing, it is possible to extract the dynamics or even find the equations of the system from data by sparse optimization [1,

2]. In machine learning, reservoir computing has been widely applied to complex and nonlinear dynamical systems for tasks such as prediction [3–22], control [23], and signal detection [24]. Quite recently, Kolmogorov-Arnold networks (KANs) [25], typically small neural networks, were proposed for discovering the dynamics from data, where even symbolic regression is possible in some cases to identify the exact mathematical equations and parameters. It has also been demonstrated [26] that the KANs have the power of uncovering the dynamical system in situations where the methods of sparse optimization fail. In all these applications, an essential requirement is that the time-series data are complete in the Nyquist sense.

A challenging but not uncommon situation is where a new system is to be learned and eventually controlled based on limited observations. Two difficulties arise in this case. First, being "new" means that the system has not been observed before, so no previous data or recordings exist. If one intends to exploit machine learning to learn and reconstruct the dynamics of the system from observations, no training data are available. Second, the observations may be irregular and sparse: the observed data are not collected at some uniform time interval, e.g., as determined by the Nyquist criterion, but at random times with the total data amount much less than that from Nyquist sampling. It is also possible that the observations can be made only once. The question is, provided with one-time sparse observations or time-series data, is it still possible to faithfully reconstruct the dynamics of the underlying system?

Limited observations or data occur in various real-world situations [27, 28]. For example, ecological data gathered from diverse and dynamic environments inevitably contain gaps caused by equipment failure, weather conditions, limited access to remote locations, and temporal or financial constraints. Similarly, in medical systems and human activity tracking, data collection frequently suffers from issues such as patient noncompliance, recording errors, loss of followup, and technical failures. Wearable devices present additional challenges, including battery depletion, user error, signal interference from clothing or environmental factors, and inconsistent wear patterns during sleep or specific activities where devices may need to be removed. A common feature of these scenarios is that the available data are only from random times without any discernible patterns. This issue becomes particularly problematic when the data is sparse. Being able to reconstruct the dynamics from sparse and random data is particularly challenging for nonlinear dynamical systems due to the possibility of chaos leading to a sensitive dependence on small variations. For example, large errors may arise when predicting the values of the dynamical variables in various intervals in which data is missing. However, if training data from the *same* target system is available, machine learning can be effective for reconstructing the dynamics from sparse data [29]. (Additional background on machine-learning approaches is provided in Supplementary Note 1.)

It is necessary to define what we mean by "random and sparse" data. We consider systems whose dynamics occur within certain finite frequency band. For chaotic systems with a broad power spectrum, in principle the "cutoff" frequency can be arbitrarily large, but power contained in a frequency range near and beyond the cutoff frequency can often be significantly smaller than that in the low frequency domain and

thus can be neglected, leading realistically to a finite yet still large bandwidth (see Supplementary Note 3). A meaningful Nyquist sampling frequency can then be defined. An observational dataset being complete means that the time series are recorded at the regular time interval as determined by the Nyquist frequency with no missing points. In this case, the original signal can be faithfully reconstructed. Random and sparse data mean that the data points are sampled at *irregular* time intervals and some portion of the data points as determined by the Nyquist frequency are missing at random times. We aim to reconstruct the system dynamics from random and sparse observations by developing a machine-learning framework to generate continuous time series that meet the Nyquist criteria, i.e., time series represented by regularly sampled data points of frequency close to the Nyquist frequency. When the governing equations of the underlying system are unknown and/or when historical observations of the full dynamical trajectory of the system are not available, the resulting lack of any training data makes the reconstruction task extremely challenging. Indeed, since the system cannot be fully measured and only irregularly observed data points are available, direct inference of the dynamical trajectory from these points is infeasible. Furthermore, the extent of the available observed data points and the number of data points to be interpolated can be uncertain.

In practice, the natural sampling rate Δs is chosen to generate the complete dataset of a dynamical system. It should ensure that the attractor remains sufficiently smooth while limiting the number of sampled points. Let the total number of samples in the dataset be L_s and the actual number of randomly selected observational points be L_s^O. The sparsity measure of the dataset can be defined as $S_m = (L_s - L_s^O)/L_s$. However, this measure depends on the natural sampling rate of the dynamical system.

To quantitatively describe the extent of sparsity in the observational data from an information-theoretic perspective, we introduce a metric that incorporates the constraints from the Nyquist sampling theorem:

$$S_r = \frac{L_s - L_s^O}{L_s - L_s^N},$$

where $L_s^N = 2f_{\max} \cdot T$ represents the minimum number of samples required according to the Nyquist theorem with $f_{\max}$ being the effective cutoff frequency of the signal and T the total time duration corresponding to L_s. In this definition, $S_r = 0$ indicates fully observed data (at the natural sampling rate), $S_r = 1$ corresponds to the theoretical minimum sampling case (at the Nyquist rate), and $S_r > 1$ indicates sub-Nyquist sampling where perfect reconstruction becomes theoretically impossible without additional constraints. Our framework takes into account not only high sparsity but also the randomness in observations. More information about the determination of "cutoff" frequency of chaotic systems can be found in Supplementary Note 3.

In this paper, we develop a machine-learning framework to address the problem of dynamics reconstruction and prediction from random and sparse observations with no training data from the target system. Our key innovation is training a hybrid machine-learning framework in a laboratory environment using a variety of synthetic dynamical systems other than data from the target system itself, and deploy the trained architecture to reconstruct the dynamics of the target system from one-time

sparse observations. More specifically, we exploit the machine-learning framework of transformers with training data *not from the target system* but from a number of known, synthetic systems that show qualitatively similar dynamical behaviors to those of the target system, for which complete data are available. The training process can thus be regarded as a "laboratory-calibration" process during which the transformer learns the dynamical rules generating the synthetic but complete data. The so-trained transformer is then deployed to a real application with the random and sparse data, and is expected to adapt to the unseen data and reconstruct the underlying dynamics. To enable long-term prediction of the target system, we exploit reservoir computing that has been demonstrated to be particularly suitable for predicting nonlinear dynamics [3–22] by feeding the output of the transformer into the reservoir computer. The combination of transformer and reservoir computing constitutes a hybrid machine-learning framework. We demonstrate that it can successfully reconstruct the dynamics of approximately three dozen prototypical nonlinear systems with high reconstruction accuracy even when the available data is only 20% of that required to faithfully represent the dynamical behavior of the underlying system. Our framework provides a paradigm of reconstructing complex and nonlinear dynamics in the extreme situation where training data from the target system do not exist and the observations or measurements are severely insufficient.

Figure 1 highlights the challenge of reconstructing the dynamics from sparse data without training data. In particular, Fig. 1(a) shows the textbook case of a random time series uniformly sampled at a frequency higher than the Nyquist frequency, which can be completely reconstructed. To illustrate random and sparse data in an intuitive setting, we consider a set of six available data points from a unit time interval, as shown in Figs. 1(b) and 1(c). The time interval contains approximately two periods of oscillation, which defines a local frequency denoted as $f_{\rm local} = 2$. As the signal is chaotic or random, the cutoff frequency $f_{\rm max}$ in the power spectrum can be higher than the frequency represented by the two oscillation cycles as shown. As a concrete example, we assume $f_{\rm max} = 3f_{\rm local}$, so the Nyquist frequency is $f_{\rm Nyquist} = 6f_{\rm local}$. If the signal is sampled at the corresponding Nyquist time interval $\Delta T = 1/f_{\rm Nyquist} = 1/12$, 12 data points would be needed. If these 12 points are sampled uniformly in time, then the signal in the two oscillation cycles can be reconstructed. The task becomes quite challenging due to two factors: the limited availability of only six data points and their random distribution across the unit time interval. Consider points #5 and #6, which occur during a downward oscillation cycle in the ground truth data. Accurately reconstructing this downward oscillation presents a key challenge. When training data from the same target system is available, standard machine learning techniques can faithfully reconstruct the dynamics [29], as illustrated in Fig. 1(b). However, without access to training data from the target system, previous methods were unable to reconstruct the dynamics from such sparse observations. A related question is, after the reconstruction, can the long-term dynamics or attractor of the system be predicted? We shall demonstrate that both the reconstruction and long-term prediction problems can be solved with hybrid machine learning, as schematically illustrated in Figs. 1(d-f).

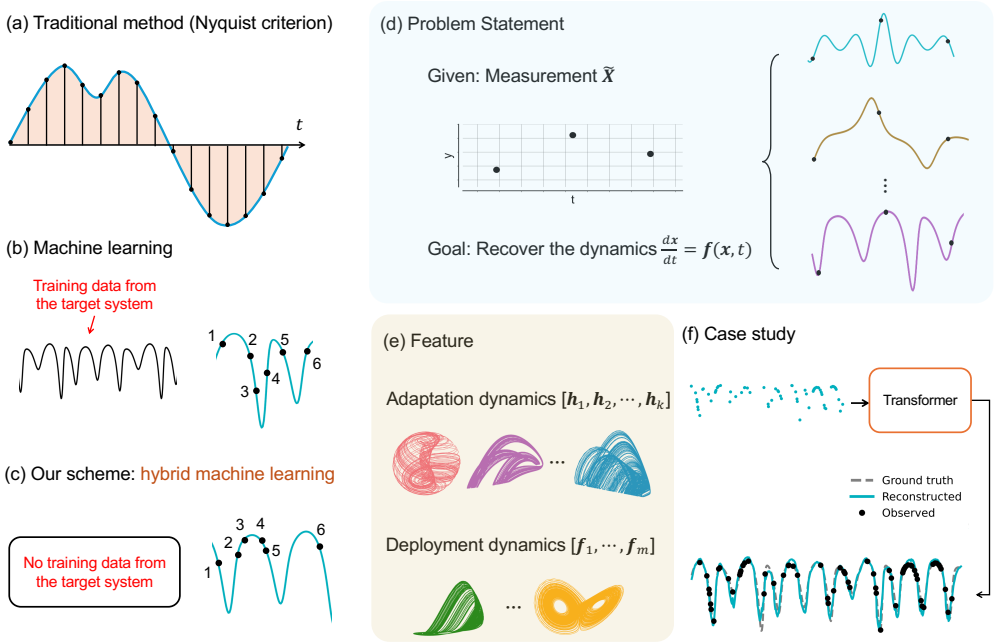

Fig. 1 Dynamics reconstruction from random and sparse data. (a) The textbook case of a random time series sampled at a frequency higher than the Nyquist frequency. (b) Training data from the target system (left) and a segment of time series of six data points in a time interval containing approximately two cycles of oscillation. According to the Nyquist criterion, the signal can be faithfully reconstructed with more than 12 uniformly sampled data points (see text). When the data points are far fewer than 12 and are randomly sampled, reconstruction becomes challenging. However, if training data from the same target system are available, existing machine-learning methods can be used to reconstruct the dynamics from the sparse data [29]. (c) If no training data from the target system are available, hybrid machine learning proposed here provides a viable solution to reconstructing the dynamics from sparse data. (d) Problem statement. Given random and sparse data, the goal is to reconstruct the dynamics of the target system governed by $d\mathbf{x}/dt = f(\mathbf{x}, t)$. A hurdle that needs to be overcome is that, for any given three points, there exist infinitely many ways to fit the data, as illustrated on the right side. (e) Training of the machine-learning framework using complete data from a large number of synthetic dynamical systems $[\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_k]$. The framework is then adapted to reconstruct and predict the dynamics of the target systems $[\mathbf{f}_1, \cdots, \mathbf{f}_m]$. (f) An example: in the testing (deployment) phase, sparse observations are provided to the trained neural network for dynamics reconstruction.

1 Results

We test our approach on three nonlinear dynamical systems in the deployment phase: a three-species chaotic food chain system [30], the classic chaotic Lorenz system [31], and Lotka-Volterra system [32]. The transformer had no prior exposure to these systems during its training (adaptation) phase. We use sparse observational data from each system to reconstruct their underlying dynamics. For clarity, in the main text, we present the results from the food-chain system, with those the other two testing systems in Supplementary Information.

Altogether, 28 synthetic chaotic systems with same dimensions as the target systems are used to train the transformer, enabling it to learn to extract dynamic

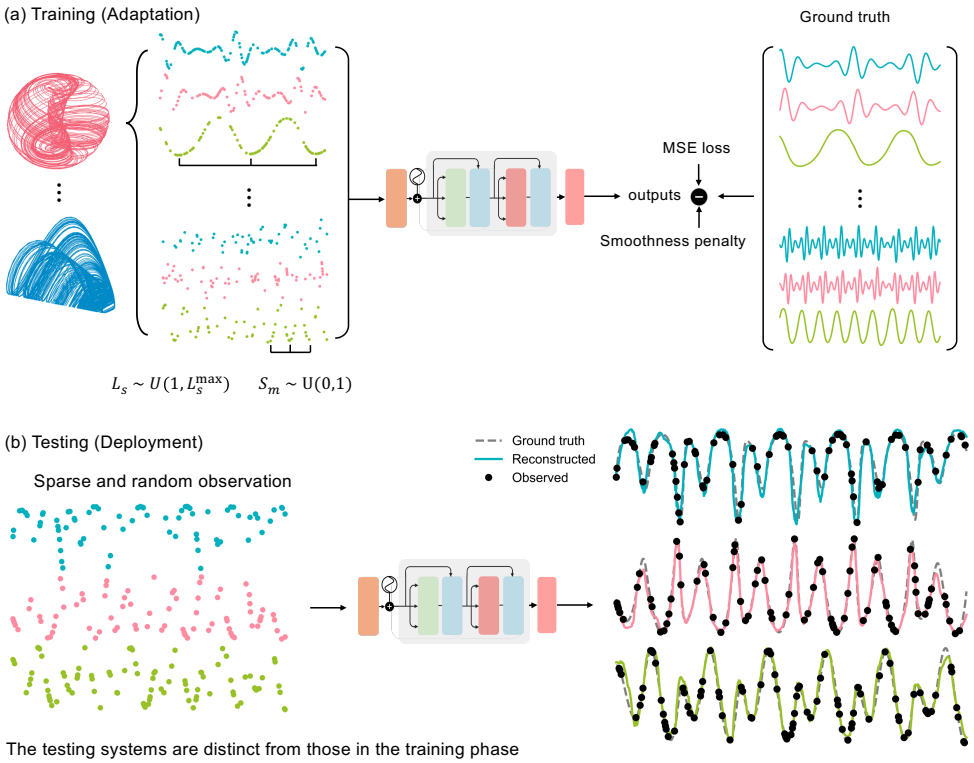

Fig. 2 Illustration of the transformer-based dynamics reconstruction framework. (a) Training (adaptation) phase, where the model is trained on various synthetic chaotic systems, each divided into segments with randomly distributed sequence lengths L_s and sparsity S_r. The data is masked before being input into the transformer, and the ground truth is used to minimize the MSE loss and smoothness loss with the output. By learning a randomly chosen segment from a random training system each time, the transformer is trained to handle data with varying lengths and different levels of sparsity. (b) Testing (deployment) phase. The testing systems are distinct from those in the training phase, i.e., the transformer is not trained on any of the testing systems. Given sparsely observed set of points, the transformer is able to reconstruct the dynamical trajectory.

behaviors from sparse observations (see Supplementary Note 2 for a detailed description). To enable the transformer to handle data from new, unseen systems of arbitrary time series length L_s and sparsity S_r, we employ the following strategy at each training step: (1) randomly selecting a system from the pool of synthetic chaotic systems and (2) preprocessing the data from the system using a uniformly distributed time series length $L_s \sim U(1, L_s^{\max})$, and uniformly distributed sparse measure $S_m \sim U(0, 1)$. By so doing, we prevent the transformer from learning any specific system dynamics too well, thereby encouraging it to treat each set of inputs as a new system. In addition, the strategy teaches the transformer to master as many features as possible. Figure 2(a) illustrates the training phase, with examples shown on the left side. On the right side, the sampled examples are encoded and fed into the transformer. The performance is evaluated by MSE loss and smoothness loss between the output and ground truth, and is used to update the neural-network weights. For predicting the long-term dynamics,

reservoir computing (Supplementary Note 4) is used. (Hyperparameter optimization for both the transformer and reservoir computer is described in Supplementary Note 5.)

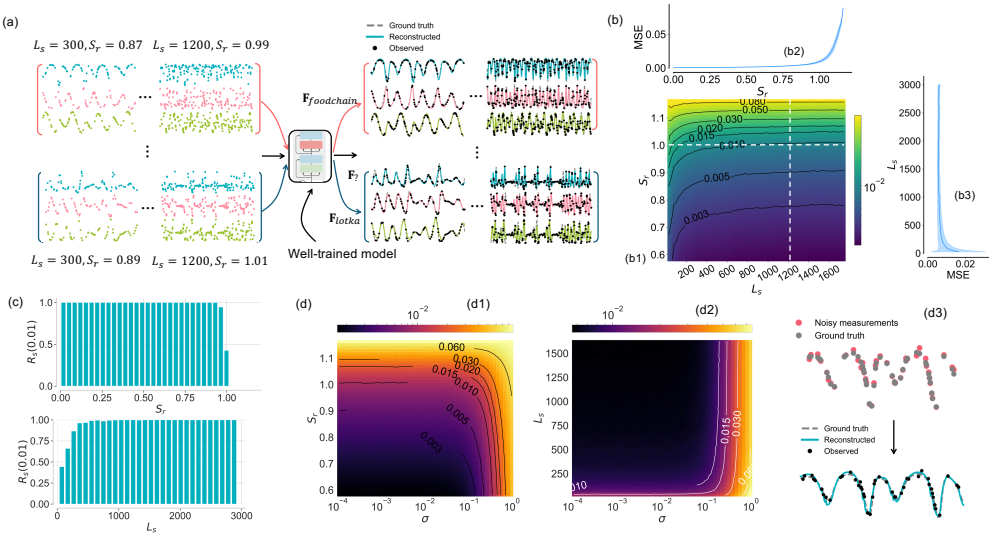

Fig. 3 Performance of dynamics reconstruction. (a) Illustration of reconstruction results for the chaotic food-chain and Lotka-Volterra systems as the testing targets that the transformer has never been exposed to. For each target system, two sets of sparse measurements of different length L_s and sparsity S_r are shown. The trained transformer reconstructs the complete time series in each case. (b) Color-coded ensemble-averaged MSE values in the parameter plane (L_s, S_r) (b1). Examples of testing MSE versus S_r and L_s only are shown in (b2) and (b3), respectively. (c) Ensemble-averaged reconstruction stability indicator $R_s(\mathrm{MSE_c})$ versus S_r and L_s, the threshold MSE is $\mathrm{MSE_c} = 0.01$. (d) Robustness of dynamics reconstruction against noise: ensemble-averaged MSE in the parameter plane (σ, S_r) (d1) and (σ, L_s) (d2), with σ being the noise amplitude. An example of reconstruction under noise of amplitude $\sigma = 0.1$ is shown in (d3). The values of the performance indicators are the result of averaging over 50 independent statistical realizations.

1.1 Dynamics reconstruction

The three species food-chain system [30] is described by

$$\begin{aligned}
\frac{dR}{dt} &= R(1-\frac{R}{\mathrm{K}}) - \frac{\mathrm{x_c y_c} CR}{R+\mathrm{R_0}}, \\
\frac{dC}{dt} &= \mathrm{x_c} C(\frac{\mathrm{y_c} R}{R+\mathrm{R_0}} - 1) - \frac{\mathrm{x_p y_p} PC}{C+\mathrm{C_0}}, \\
\frac{dP}{dt} &= \mathrm{x_p} P(\frac{\mathrm{y_p} C}{C+\mathrm{C_0}} - 1),
\end{aligned} \tag{1}$$

where R, C, and P are the population densities of the resource, consumer, and predator species, respectively. The system has seven parameters: $\mathrm{K}, \mathrm{x_c}, \mathrm{y_c}, \mathrm{x_p}, \mathrm{y_p}, \mathrm{R_0}, \mathrm{C_0} > 0$.

Figure 2(b) presents an example of reconstructing the dynamics of the chaotic food-chain system for $L_s = 2000$ and $S_r = 1$ ($S_m = 0.86$). The target output time series for each dimension should contain 2000 points (about 40 cycles of oscillation), but only randomly selected $L_s^N = 280$ points are exposed to the trained transformer. The right side of Fig. 2(b) shows the reconstructed time series, where the three dynamical variables are represented by different colors, the black points indicate observations, and the gray dashed lines are the ground truth. Only a segment of a quarter of the points is displayed. This example demonstrates that, with such a high level of sparsity, directly connecting the observational points will lead to significant errors. Instead, the transformer infers the dynamics by filling the gaps with the correct dynamical behavior. It is worth emphasizing that the testing system has never been exposed to the transformer during the training phase, requiring the neural machine to explore the underlying unknown dynamics from sparse observations based on experience learned from other systems. Extensive results with varying values of the parameters L_s and S_r for the three testing systems can be found in Supplementary Note 6.

1.2 Performance of dynamics reconstruction

To characterize the performance of dynamics reconstruction, we use two measures: MSE and prediction stability $R_s(\mathrm{MSE_c})$, the probability that the transformer generates stable predictions (see Methods). Figure 3(a) shows the working of the framework in the testing phase: a well-trained transformer receives inputs from previously unseen systems, with random sequence length L_s and sparsity S_r, and is able to reconstruct the dynamics. Some representative time-series segments of the reconstruction and the ground truth are displayed. Figures 3(b) and 3(c) depict the ensemble-averaged reconstruction performance for the chaotic food chain system. As L_s increases and S_r decreases, the transformer can gain more information to facilitate reconstructions. When the available data become more sparse, the performance degrades. Overall, under conditions with random noisy observations, satisfactory reconstruction of new dynamics can be achieved for $S_r \leq 1.0$ and a sequence length larger than 500 (about 10 cycles of oscillation).

It is essential to assess how noise affects the dynamics reconstruction. Figure 3(d) shows the effects of the multiplicative noise (see Methods) on the reconstruction performance. The results indicate that, for reasonably small noise (e.g., $\sigma < 10^{-1}$), robust reconstruction with relatively low MSE values can be achieved. We have also studied the effect of additive noise, with results presented in Supplementary Note 7.

1.3 Key features of dynamics reconstruction

Transformer has the ability to reconstruct the time series of previously unknown dynamical systems, particularly under extreme sparsity. This capability stems from the generalizability of the transformer during its training on sufficiently diverse chaotic systems with large data. OpenAI reported a power-law relation between the transformer performance and model/data size [33]. Here we study how reconstruction performance depends on the number of training systems. Specifically, we train the transformer on k chaotic systems, where k ranges from 1 to 28. For each value of k, we randomly

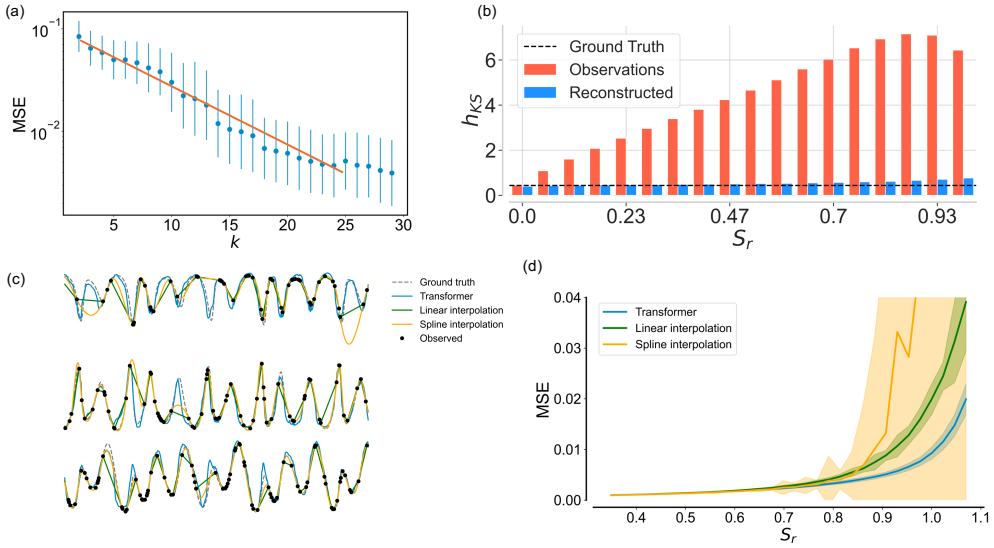

Fig. 4 Demonstration of the capabilities of the transformer-based dynamics reconstruction. (a) Power-law decrease of MSE as the number of training systems k increase. (b) Entropy rate of the observational data and the time series reconstructed by the transformer. The dashed line represents the entropy rate of the ground truth. (c) Example of a time series reconstructed by the transformer, compared with linear and spline interpolations, shown in blue, green, and orange, respectively. Traditional interpolation methods fail to recover the time series accurately due to their inability to capture the underlying dynamics. (d) MSE versus sparsity. While all methods perform similarly under low sparsity, the transformer outperforms the other two methods in reconstructing dynamics when the observational points are sparse. In all cases, 50 independent realizations are used.

sample a subset from the pool of 28 chaotic systems and calculate the average MSE over 50 iterations. To ensure robustness, the MSE is averaged across the sparsity measure S_m whose value ranges from 0 to 1 at the interval of 0.05. As shown in Fig. 4(a), the MSE decreases with increasing k following a power-law trend with saturation, demonstrating that training our transformer-based framework on a diverse of chaotic systems with sufficient data is crucial for successful dynamics reconstruction. Moreover, once the model has acquired this generalization ability, it can infer the governing dynamics of arbitrary new system from sparse observations. To demonstrate this, we have shown that our trained transformer performs well on 28 additional unseen target systems [34] (Supplementary Note 12).

How much information is contained in the chaotic systems being presented to and interpolated by the transformer? We employ Kolmogorov-Sinai (KS) entropy [35, 36], denoted as h_{KS}, to estimate the entropy rate of the dynamical systems (Methods). For the transformer outputs, i.e., the reconstructed time series, and the ground truth data, we estimate h_{KS} directly using this method. However, a challenge is that the input to the transformer is sparse, which we meet by considering only those time steps in the input sequence where all dimensions are observed, while discarding time points with missing values in any variable. The processed input is then used to calculate the entropy rate. The estimated h_{KS} for both the input and the reconstructed time series generated by the transformer are shown in Fig. 4(b), where the black dashed line

denotes the KS entropy calculated from the ground truth. While increased sparsity S_r corresponds to a gradual increase in entropy in the reconstructed time series, it remains closer to the ground truth than the entropy of the sparse observations, which exhibit substantial divergence. From an information theoretic point of view, the transformer receives sparse observations characterized by high uncertainty and disorder, yet it successfully reconstructs the dynamics to closely match the ground truth.

While our method is applicable across a wide range of sparsity, traditional techniques such as linear and spline interpolation can also achieve high accuracy when the sparsity level is low. These classical methods are simple and computationally efficient, and perform adequately in regimes with sufficient observational data. However, as the data sparsity level increases, the limitations of traditional interpolation become evident. To explicitly demonstrate this, we take two examples of traditional interpolation methods as an example: linear and spline interpolation, where the former approximates missing values by connecting the nearest available data points with straight lines and the latter constructs piecewise polynomial functions to produce smooth transitions between observed points [37,38]. Both methods, despite their simplicity, by design lack the capacity to capture the intrinsic dynamics of complex systems. Figure 4(c) shows a representative example for sparsity $S_r = 1.0$ ($S_m = 0.86$), where the transformer successfully reconstructs the underlying dynamics, but the linear and spline interpolation methods fail to recover the correct temporal structure.

To quantify the performance, we calculate the MSE between the reconstructed time series and the ground truth. Figure 4(d) shows the reconstruction performance across a range of sparsity levels for a fixed sequence length, $L_s = 2,000$ (approximately 400 oscillation cycles). Results are averaged over 50 independent realizations, with shadowed areas indicating the standard deviation. When the sparsity measure S_r is low, all three methods - transformer, linear, and spline interpolation - perform comparably. However, once S_r exceeds approximately 0.7, the transformer begins to outperform the other methods, with its advantage becoming more pronounced as the available data become increasingly more sparse.

In addition to traditional interpolation methods, compressed sensing (CS) can also work as a signal reconstruction framework. CS assumes the the signal is sparse in a known basis and often employs optimization-based recovery [37, 38]. It is important to note that the definition of the term "sparse" in CS is referred to as the signal having only a few non-zero components when expressed in an appropriate basis (e.g., Fourier or wavelet). However, the strict assumption can limit the applicability of CS. In contrast, our method is model-free, data-driven, and capable of generalizing across unseen complex dynamical systems, regardless of the dynamics are sparse or not. Simulation results show that for a target system, when the observational sparsity is high, our framework outperforms CS significantly (Supplementary Note 10).

1.4 Prediction of long-term dynamical climate

The results presented so far are for reconstruction of relatively short term dynamics, where the sequence length L_s is limited to below 3000, corresponding to approximately 60 cycles of dynamical oscillation in the data. Can the long-term dynamical behavior or climate as characterized by, e.g., a chaotic attractor, be faithfully predicted? To

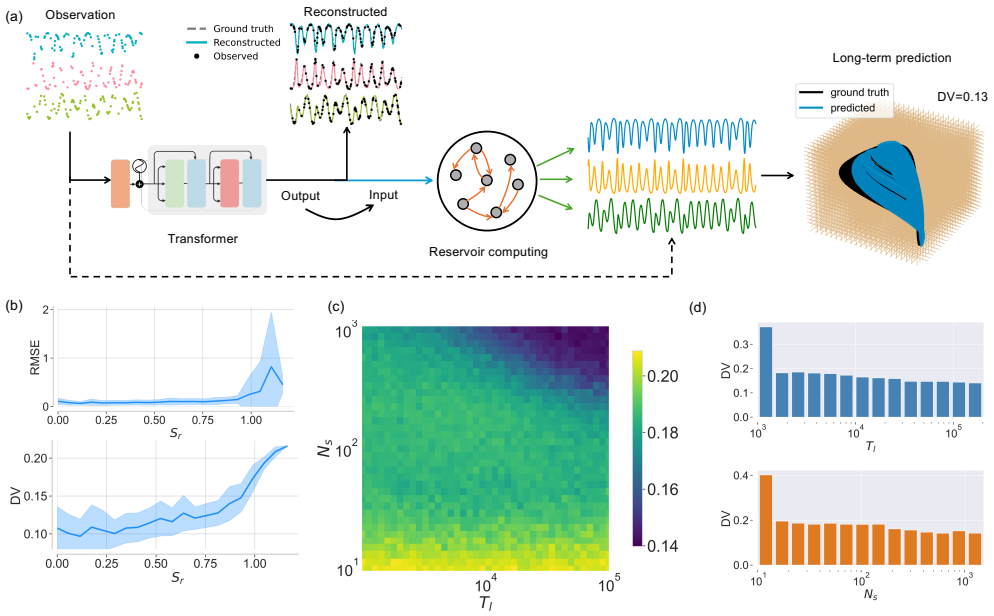

Fig. 5 Reservoir-computing based long-term dynamics prediction. (a) An illustration of hybrid transformer/reservoir-computing framework. The time series reconstructed by the transformer is used to train the reservoir computer that generates time series of the target system of arbitrary length, leading to a reconstructed attractor that agrees with the ground truth. (b) RMSE and DV versus the sparsity parameter. (c) Color-coded ensemble-averaged DV in the reservoir-computing hyperparameter plane (T_l, N_s) for $S_r = 0.93$ ($S_m = 0.8$). (d) DV versus training length T_l for $N_s = 500$ and versus reservoir network size N_s for $T_l = 10^5$. In all cases, 50 independent realizations are used.

address this question, we note that reservoir computing has the demonstrated ability to generate the long-term behavior of chaotic systems [5, 7, 13, 16, 21]. Our solution is then employing reservoir computing to learn the output time series generated by the transformer so as to further process the reconstructed time series. Assume that a number of sparse data segments are available. The corresponding transformer-reconstructed segments are then used as the training data for the reservoir computer for it to find the relationship between the dynamical state at the current step and that in the immediate future. The trained reservoir computer can predict or generate any length of time series of the target system, as exemplified in Fig. 5(a). It can be seen that the reservoir-computing generated attractor agrees with the ground truth. More details about reservoir computing, its training and testing can be found in Supplementary Note 4.

To evaluate the performance of the reservoir-computing generated long-term dynamics, we use two measures: root MSE (RMSE) and deviation value (DV) (Methods). Figure 5(b) presents the short- and long-term prediction performance by comparing the reservoir-computing predicted attractors with the ground truth. We calculate the RMSE using a short-term prediction length of 150 (corresponding to approximately 3 cycles of oscillation), and the DV using a long-term prediction length of 10,000 (approximately to 200 cycles of oscillation). The reconstructed time series

and attractors are close to their respective ground truth when the sparsity parameter S_r is below 0.93, i.e., sparse measure is below 0.8, as indicated by the low RMSE and DV values. The number of available data segments from the target system tends to have a significant effect on the prediction accuracy. Figures 5(c) and 5(d) show the dependence of the DV on two reservoir-computing hyperparameters: the training length T_l and the reservoir network size N_s. As the training length and network size increase, DV decreases, indicating improved performance.

2 Discussion

Exploiting machine learning to understand, predict and control the behaviors of nonlinear dynamical systems have demonstrated remarkable success in solving previously deemed difficult problems [23, 39]. However, an essential prerequisite for these machine-learning studies is the availability of training data. Often, extensive and uniformly sampled data of the target system are required for training. In addition, in most previous works, training and testing data are from the same system, with a focus on minimizing the average training errors on the specific system and greedily improving the performance by incorporating all correlations within the data (iid - independently and identically distributed assumption). While the iid setting can be effective, unforeseen distribution shifts during testing or deployment can cause the optimization purely based on the average training errors to perform poorly [40]. Several strategies have been proposed to handle nonlinear dynamical systems. One approach trains neural networks using data from the same system under different parameter regimes, enabling prediction of new dynamical behaviors including critical transitions [16]. Another method uses data from multiple systems to train neural networks in tasks like memorizing and retrieving complex dynamical patterns [41, 42]. However, this latter approach fails when encountering novel systems not present in the training data. Meta-learning has been shown to achieve satisfactory performance with only limited data, but training data from the target systems are still required to fine-tune the network weights [43]. In addition, a quite recent work used well-defined, pre-trained large language models not trained using any chaotic data and showed that these models can predict the short-term and long-term dynamics of chaotic systems [44].

We developed a hybrid machine-learning framework to construct the dynamics of target systems, under two limitations: (1) the available observational data are random sparse and (2) no training data from the system are available. We address this challenge by training the transformer using synthetic or simulated data from numerous chaotic systems, completely excluding data from the target system. This allows direct application to the target system without fine-tuning. To ensure the transformer's effectiveness on previously unseen systems, we implement a "triple-randomness" training regime that varies the training systems, input sequence length, and sparsity level. As a result, the transformer will treat each dataset as a new system, rather than adequately learning the dynamics of any single training system. This process continues with data from different chaotic systems with random input sequence length and sparsity until the transformer is experienced and able to "perceive" the underlying dynamics from

the sparse observations. The end result of this training process is that the transformer gains "knowledge" through its experience by adapting to the diverse synthetic datasets. It is worth noting that the dimension of the systems (i.e. the number of variables) provided to the transformer in the inference phase should match those in the testing phase. During the testing or deployment phase, the transformer reconstructs dynamics from sparse data of arbitrary length and sparsity drawn from a completely new dynamical system. When multiple segments of sparse observations are available, we are able to reconstruct the system's long-term "climate" through a two-step process. First, the transformer repeatedly reconstructs system dynamics from these data segments. Second, reservoir computing uses these transformer-reconstructed dynamics as training data to generate system evolution over any time duration. The combination of the transformer and reservoir computing constitutes our hybrid machine-learning framework, enables reconstruction of the target system's long-term dynamics and attractor from sparse data alone.

We emphasize the key feature of our hybrid framework: reconstructing the dynamics from sparse observations of an unseen dynamical system, even when the available data has a high degree of sparsity. We have tested the framework on two benchmark ecosystems and one classical chaotic system. In all cases, with extensive training conducted on synthetic datasets under diverse settings, accurate and robust reconstruction has been achieved. Empirically, the minimum requirements for the transformer to be effective are: the dataset from the target system should have the length of at least 20 average cycles of its natural oscillation and the sparsity degree is less than 1. For subsequent reservoir computing learning, at least three segments of the time series data from the transformer are needed for reconstructing the attractor of the target system. We have also addressed issues such as the effect of noise and hyperparameter optimization. The key to the success of the hybrid framework lies in *versatile dynamics*: with training based on the dynamical data from a diverse array of synthetic systems, the transformer will gain the ability to reconstruct the dynamics of the "never-seen" target systems. In essence, the reconstruction performance on unseen target systems follows a power-law trend with respect to the number of synthetic systems used during the training phase. We have provided a counter example that, when dynamics are lacking in the time series, the framework fails to perform the reconstruction task (see Supplementary Note 8).

It is worth noting that both the training and target dynamical systems in our experiments are autonomous. However, real-world systems can often be nonautonomous. To adapt the framework to target nonautonomous systems, we have developed a mixed training strategy that involves both autonomous and nonautonomous systems (Supplementary Note 9). With regard to long-term prediction, climate dynamics are not stationary but often time-variant, i.e., nonautonomous. When providing the reservoir computer with high-fidelity outputs generated by the transformer from sparse observations, long-term climate prediction becomes feasible. Moreover, we have demonstrated the superiority of our proposed hybrid machine-learning scheme to traditional interpolation methods, traditional recurrent neural networks, and compressed sensing (Supplementary Note 10). Additional results from chaotic systems are presented in Supplementary Note 12.

Our hybrid transformer/reservoir-computing (T-RC) framework represents a powerful tool for dynamics reconstruction and prediction of long-term behavior in situations where only sparse observations from a newly encountered system are available. In fact, such a situation is expected to arise in different fields. Possible applications extend to medical and biological systems, particularly in wearable health monitoring where data collection is often interrupted. For instance, smartwatches and fitness trackers regularly experience gaps due to charging, device removal during activities like swimming, or signal interference. Another potential application is predicting critical transitions from sparse and noisy observations, such as detecting when an athlete's performance metrics indicate approaching over training, or when a patient's vital signs suggest an impending health event. In these cases, our hybrid framework can reconstruct complete time series from incomplete wearable device data, serving as input to parameter-adaptable reservoir computing [16, 45] for anticipating these critical transitions. This approach is particularly valuable for continuous health monitoring where data gaps are inevitable, whether from smart devices being charged, removed, or experiencing connectivity issues.

3 Methods

3.1 Hybrid machine learning

Consider a nonlinear dynamical system described by

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{F}(\mathbf{x}(t), t), \quad t \in [0, T], \tag{2}$$

where $\mathbf{x}(t) \in \mathbb{R}^D$ is a D-dimensional state vector and $\mathbf{F}(\cdot)$ is the unknown velocity field. Let $\mathbf{X} = (\mathbf{x}_0, \cdots, \mathbf{x}_{L_s})^\intercal \in \mathbb{R}^{L_s \times D}$ be the full uniformly sampled data matrix of dimension $L_s \times D$ with each dimension of the original dynamical variable containing L_s points. A sparse observational vector can be expressed as

$$\tilde{\mathbf{X}} = \mathbf{g}_\alpha(\mathbf{X})(1 + \sigma \cdot \Xi), \tag{3}$$

where $\tilde{\mathbf{X}} \in \mathbb{R}^{L_s \times D}$ is the observational data matrix of dimension $L_s \times D$ and $\mathbf{g}_\alpha(\cdot)$ is the following element-wise observation function:

$$X_{ij}^O = g_\alpha(X_{ij}) = \begin{cases} X_{ij}, & \text{if } X_{ij} \text{ is observed,} \\ 0, & \text{otherwise,} \end{cases} \tag{4}$$

with α representing the probability of matrix element X_{ij} being observed. In Eq. (3), Gaussian white noise of amplitude σ is present during the measurement process, where $\Xi \sim \mathcal{N}(0, 1)$. Our goal is utilizing machine learning to approximate the system dynamics function $\mathbf{F}(\cdot)$ by another function $\mathbf{F}'(\cdot)$, assuming that $\mathbf{F}$ is Lipschitz continuous with respect to $\mathbf{x}$ and the observation function produces sparse data: $\mathbf{g} : \mathbf{X} \to \tilde{\mathbf{X}}$. To achieve this, it is necessary to design a function $\mathcal{F}(\tilde{\mathbf{X}}) = \mathbf{X}$ that comprises implicitly $\mathbf{F}'(\cdot) \approx \mathbf{F}(\cdot)$ so that it reconstructs the system dynamics by filling the gaps in the

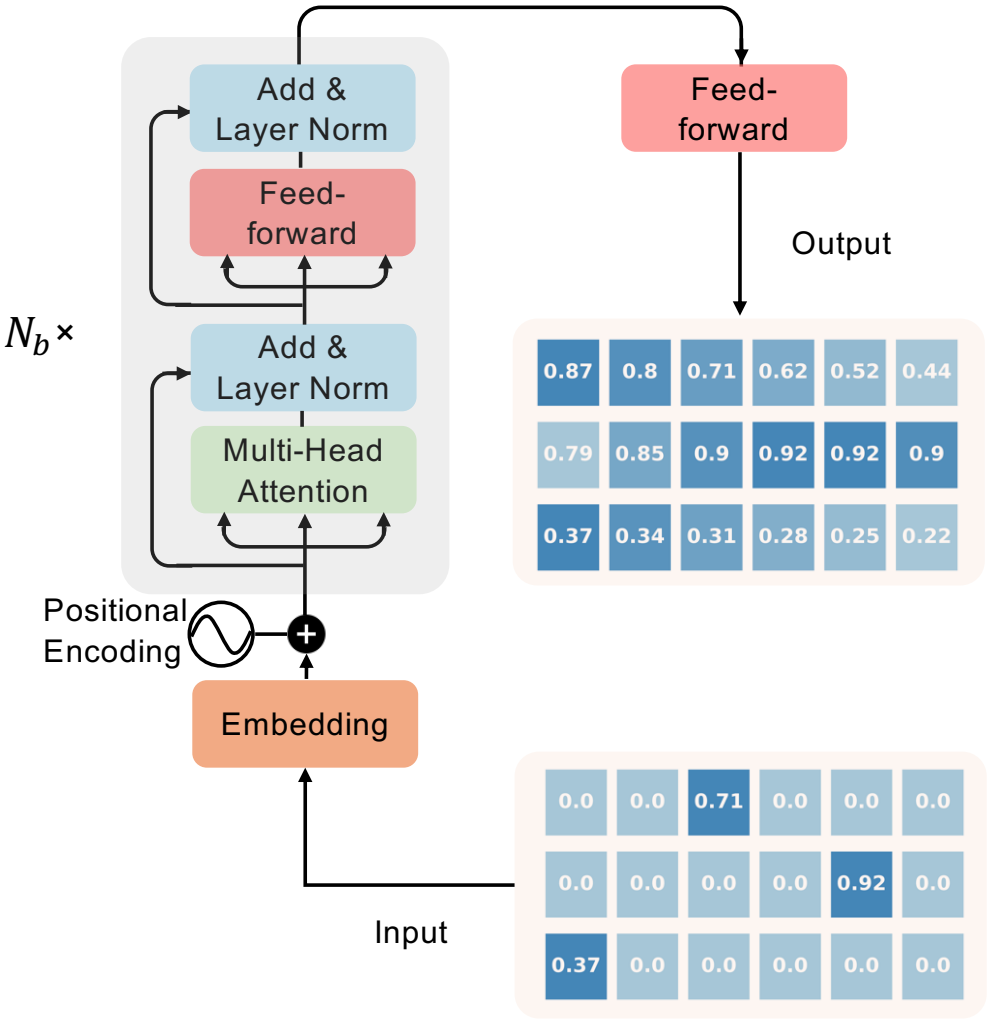

Fig. 6 Transformer architecture. The transformer receives the sparse and random observation as the input and generates the reconstructed output. See text for a detailed mathematical description.

observation, where $\mathcal{F}(\tilde{\mathbf{X}})$ should have the capability of adapting to any given unknown dynamics.

Selecting an appropriate neural network architecture for reconstructing dynamics from sparse data requires meeting two fundamental requirements: (1) dynamical memory to capture long-range dependencies in the sparse data, and (2) flexibility to handle input sequences of varying lengths. Transformers [46], originally developed for natural language processing, satisfy these requirements due to their basic attention structure. In particular, transformers has been widely applied and proven effective for time series analysis, such as prediction [47–49], anomaly detection [50], and classification [51]. Figure 6 illustrates the transformer's main structure. The data matrix $\tilde{\mathbf{X}}$ is first processed through a linear fully-connected layer with bias, transforming it into

an $L_s \times N$ matrix. This output is then combined with a positional encoding matrix, which embeds temporal ordering information into the time series data. This projection process can be described as [52]:

$$\mathbf{X}_p = \tilde{\mathbf{X}}\mathbf{W}_p + \mathbf{W}_b + \mathbf{PE}, \tag{5}$$

where $\mathbf{W}_p \in \mathbb{R}^{D\times N}$ represents the fully-connected layer with the bias matrix $\mathbf{W}_b \in \mathbb{R}^{L_s\times N}$ and the position encoding matrix is $\mathbf{PE} \in \mathbb{R}^{L_s\times N}$. Since the transformer model does not inherently capture the order of the input sequence, positional encoding is necessary to provide the information about the position of each time step. For a given position $1 \le \text{pos} \le L_s^{max}$ and dimension $1 \le d \le D$, the encoding is given by

$$\mathbf{PE}_{pos,2d} = \sin\left(\frac{\text{pos}}{10000^{2d/N}}\right), \tag{6}$$

$$\mathbf{PE}_{pos,2d+1} = \cos\left(\frac{\text{pos}}{10000^{2d/N}}\right), \tag{7}$$

The projected matrix $\mathbf{X}_p \in \mathbb{R}^{L_s\times N}$ then serves as the input sequence for N_b attention blocks. Each block contains a multi-head attention layer, a residual layer (add & layer norm), and a feed-forward layer, and a second residual layer. The core of the transformer lies in the self-attention mechanism, allowing the model to weight the significance of distinct time steps. The multi-head self-attention layer is composed of several independent attention blocks. The first block has three learnable weight matrices that linearly map $\mathbf{X}_p$ into query $\mathbf{Q}_1$ and key $\mathbf{K}_1$ of the dimension $L_s \times d_k$ and value $\mathbf{V}_1$ of the dimension $L_s \times d_v$:

$$\mathbf{Q}_1 = \mathbf{X}_p\mathbf{W}_{\mathbf{Q}_1}, \quad \mathbf{K}_1 = \mathbf{X}_p\mathbf{W}_{\mathbf{K}_1}, \quad \mathbf{V}_1 = \mathbf{X}_p\mathbf{W}_{\mathbf{V}_1}, \tag{8}$$

where $\mathbf{W}_{\mathbf{Q}_1} \in \mathbb{R}^{N\times d_k}$, $\mathbf{W}_{\mathbf{K}_1} \in \mathbb{R}^{N\times d_k}$, and $\mathbf{W}_{\mathbf{V}_1} \in \mathbb{R}^{N\times d_v}$ are the trainable weight matrices, d_k is the dimension of the queries and keys, and d_v is the dimension of the values. A convenient choice is $d_k = d_v = N$. The attention scores between the query $\mathbf{Q}_1$ and the key $\mathbf{K}_1$ are calculated by a scaled multiplication, followed by a softmax function:

$$\mathbf{A}_{\mathbf{Q}_1,\mathbf{K}_1} = \text{softmax}\left(\frac{\mathbf{Q}_1\mathbf{K}_1^\mathsf{T}}{\sqrt{d_k}}\right), \tag{9}$$

where $\mathbf{A}_{\mathbf{Q}_1,\mathbf{K}_1} \in \mathbb{R}^{L_s\times L_s}$. The softmax function normalizes the data with $\text{softmax}(x_i) = \exp(x_i)/\sum_j \exp(x_j)$, and the $\sqrt{d_k}$ factor mitigates the enlargement of standard deviation due to matrix multiplication. For the first head (in the first block), the attention matrix is computed as a dot product between $\mathbf{A}_{\mathbf{Q}_1,\mathbf{K}_1}$ and $\mathbf{V}_1$:

$$\begin{aligned}\mathbf{O}_{11} &= \text{Attention}(\mathbf{Q}_1, \mathbf{K}_1, \mathbf{V}_1), \\ &= \mathbf{A}_{\mathbf{Q}_1,\mathbf{K}_1}\mathbf{V}_1 = \text{softmax}\left(\frac{\mathbf{Q}_1\mathbf{K}_1^\mathsf{T}}{\sqrt{d_k}}\right)\mathbf{V}_1,\end{aligned} \tag{10}$$

where $\mathbf{O}_{11} \in \mathbb{R}^{L_s \times d_v}$. The transformer employs multiple (h) attention heads to capture information from different subspaces. The resulting attention heads $\mathbf{O}_{1i}$ ($i = 1, \ldots, h$) are concatenated and mapped into a sequence $\mathbf{O}_1 \in \mathbb{R}^{L_s \times N}$, described as:

$$\mathbf{O}_1 = \mathcal{C}(\mathbf{O}_{11}, \mathbf{O}_{12}, \cdots \mathbf{O}_{1h})\mathbf{W}_{o1}, \tag{11}$$

where $\mathcal{C}$ is the concatenation operation, h is the number of heads, and $\mathbf{W}_{o1} \in \mathbb{R}^{hd_v \times N}$ is an additional matrix for linear transformation for performance enhancement. The output of the attention layer undergoes a residual connection and layer normalization, producing $\mathbf{X}_{R1}$ as follows:

$$\mathbf{X}_{R1} = \text{LayerNorm}(\mathbf{X}_\mathbf{p} + \text{Dropout}(\mathbf{O}_1)) \tag{12}$$

A feed-forward layer then processes this data matrix, generating output $\mathbf{X}_{F1} \in \mathbb{R}^{L_s \times N}$ as:

$$\mathbf{X}_{F1} = \max(0, \mathbf{X}_{R1}\mathbf{W}_{F_a} + \mathbf{b}_a)\mathbf{W}_{F_b} + \mathbf{b}_b, \tag{13}$$

where $\mathbf{W}_{F_a} \in \mathbb{R}^{N \times d_f}$, $\mathbf{W}_{F_b} \in \mathbb{R}^{d_f \times N}$, $\mathbf{b}_a$ and $\mathbf{b}_b$ are biases, and $\max(0, \cdot)$ denotes a ReLU activation function. This output is again subjected to a residual connection and layer normalization.

The output of the first block operation is used as the input to the second block. The same procedure is repeated for each of the remaining $N_b - 1$ blocks. The final output passes through a feed-forward layer to generate the prediction. Overall, the whole process can be represented as $\mathbf{Y} = \mathcal{F}(\tilde{\mathbf{X}})$.

The second component of our hybrid machine-learning framework is reservoir computing, which takes the output of the transformer as the input to reconstruct the long-term "climate" or attractor of the target system. A detailed description of reservoir computing used in this context and its hyperparameters optimization are presented in Supplementary Notes 4 and 5.

3.2 Machine learning loss

To evaluate the reliability of the generated output, we minimize a combined loss function with two components: (1) a mean squared error (MSE) loss that measures absolute error between the output and ground truth, and (2) a smoothness loss that ensures the output maintains appropriate continuity. The loss function is given by

$$\mathcal{L} = \alpha_1 \mathcal{L}_{\text{mse}} + \alpha_2 \mathcal{L}_{\text{smooth}}, \tag{14}$$

where α_1 and α_2 are scalar weights controlling the trade-off between the two loss terms. The first component $\mathcal{L}_{\text{mse}}$ measures the absolute error between the predictions and the ground truth:

$$\mathcal{L}_{\text{mse}} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \tag{15}$$

with n being the total number of data points, y_i and $\hat{y}_i$ denoting the ground truth and predicted value at time point i, respectively. The second component $\mathcal{L}_{\text{smooth}}$ of the loss function consists of two terms: Laplacian regularization and total variation regularization, which penalize the second-order differences and absolute differences, respectively, between consecutive predictions. The two terms are given by:

$$\mathcal{L}_{\text{laplacian}} = \frac{1}{n-2}\sum_{i=2}^{n-1}(\hat{y}_{i-1} + \hat{y}_{i+1} - 2\hat{y})^2, \tag{16}$$

and

$$\mathcal{L}_{\text{tv}} = \frac{1}{n-1}\sum_{i=1}^{n-1}|\hat{y}_i - \hat{y}_{i+1}|. \tag{17}$$

We assign the same weights to the two penalties, so the final combined loss function to be minimized is

$$\mathcal{L} = \mathcal{L}_{\text{mse}} + \alpha_s(\mathcal{L}_{\text{laplacian}} + \mathcal{L}_{\text{tv}}). \tag{18}$$

We set $\alpha_s = 0.1$. It is worth noting that the smoothness penalty is a crucial hyperparameter that should be carefully selected. Excessive smoothness leads the model to learn overly coarse-grained dynamics, while absence of a smoothness penalty causes the reconstructed curves to exhibit poor smoothness (Supplementary Note 5).

3.3 Computational setting

Unless otherwise stated, the following computational settings for machine learning are used. Given a target system, time series are generated numerically by integrating the system with time step $dt = 0.01$. The initial states of both the dynamical process and the neural network are randomly set from a uniform distribution. An initial phase of the time series is removed to ensure that the trajectory has reached the attractor. The training and testing data are obtained by sampling the time series at the interval Δ_s chosen to ensure an acceptable generation. Specifically, for the chaotic food chain, Lorenz and Lotka-Volterra systems, we set $\Delta_s = 1$, $\Delta_s = 0.02$, and $\Delta_s = 0.1$ respectively, corresponding to approximately 1 over 30 $\sim$ 50 cycles of oscillation. A similar procedure is also applied to other synthetic chaotic systems (See Table S3 for Δ_s values for each system). The time series data are preprocessed by using min-max normalization so that they are in the range [0,1]. The complete data length for each system is 1,500,000 (about 30,000 cycles of oscillation), which is divided into segments with randomly chosen sequence lengths L_s and sparsity S_r. For the transformer, we use a maximum sequence length of 3,000 (corresponding to about 60 cycles of oscillation) - the limitation of input time series length. We apply Bayesian optimization [53] and a random search algorithm [54] to systematically explore and identify the optimal set of various hyperparameters. Two chaotic Sprott systems - Sprott_0 and Sprott_1 - are used to find the optimal hyperparameters, ensuring no data leakage from the testing systems. The optimized hyperparameters for the transformer are listed in Table 1.

Table 1 Optimal transformer hyperparameter values

Hyperparameters	Descriptions
$D_l = 1,500,000$	Training length for each system
$d_f = 512$	Number of Feedforward neurons
$h = 4$	Number of Transformer heads
$N_b = 4$	Number of Transformer blocks
$N = 128$	input embedded dimension
$\sigma = 0.05$	Measurement noise level
$L_s^{\max} = 3000$	Maximum input sequence length
$b_s = 16$	Batch size
epoch $= 50$	Epoch
$lr = 0.001$	Learning rate
$P_{drop} = 0.2$	Dropout rate

All simulations are run using Python on computers with six RTX A6000 NVIDIA GPUs. A single training run of our framework typically takes about 30 minutes using one of the GPUs.

3.4 Prediction stability

The prediction stability describe the probability that the transformer generates stable predictions, which is defined as the probability that the MSE is below a predefined stable threshold $\mathrm{MSE_c}$:

$$R_s(\mathrm{MSE_c}) = \frac{1}{n}\sum_{i=1}^{n}[\mathrm{MSE} < \mathrm{MSE_c}], \tag{19}$$

where n is the number of iterations and $[\cdot] = 1$ if the statement inside is true and zero otherwise.

3.5 Deviation value

For a three-dimensional target system, we divide the three-dimensional phase space into a uniform cubic lattice with the cell size $\Delta = 0.05$ and count the number of trajectory points in each cell, for both the predicted and true attractors in a fixed time interval. The DV measure is defined as [21]

$$\mathrm{DV} \equiv \sum_{i=1}^{m_x}\sum_{j=1}^{m_y}\sum_{k=1}^{m_z}\sqrt{(f_{i,j,k} - \hat{f}_{i,j,k})^2}, \tag{20}$$

where m_x, m_y, and m_z are the total numbers of cells in the x, y, and z directions, respectively, $f_{i,j,k}$ and $\hat{f}_{i,j,k}$ are the frequencies of visit to the cell (i, j, k) by the predicted and true trajectories, respectively. If the predicted trajectory leaves the phase space boundary, we count it as if it has landed in the boundary cells where the true trajectory never goes.

3.6 Noise implementation

We study how two types of noise affect the dynamics reconstruction in this work: multiplicative and additive noise. We use normally distributed stochastic processes of zero mean and standard deviation σ, while the former perturbs the observational points x to $x + x \cdot \xi$ after normalization and the latter perturbs x to $x + \xi$. Note that multiplicative (demographic) noise is common in ecological systems.

3.7 Entropy rate estimation

We estimate the entropy rate of the dynamical system using the Kolmogorov-Sinai (KS) entropy, denoted as h_{KS}. This quantity measures the average number of nats of information produced per unit time by the system and reflects its intrinsic unpredictability. Since the governing equations of systems are assumed to be unknown, we apply a recurrence-based approach that is both data-driven and robust to sparse observability [36]. This method is based on the distribution of first recurrence times, i.e., the time it takes for the system to return close to a previous state. Given time series $\mathbf{X} \in \mathbb{R}^{L_s \times D}$, where $\mathbf{x}(t)$ is its state vector at time t, we compute the pairwise Euclidean distances. Specifically, for each time point t, we identify the smallest time lag τ such that:

$$||\mathbf{x}(t) - \mathbf{x}(t + \tau)|| < r,$$

where r is a fixed recurrence threshold. Through this process, a set of recurrence times $\tau(k)$ can be obtained, which defines the empirical distribution $\rho(\tau)$. The entropy rate is then estimated as:

$$h_{KS} \approx \frac{1}{\tau_{\min}} \sum_{\tau} \rho(\tau) \log(\frac{1}{\rho(\tau)}),$$

where $\tau_{\min}$ is the minimum observed recurrence time. In our work, we set the recurrence threshold $r = 0.5$ after normalizing the data to unit variance.

4 Data availability

The data is publicly available via Zenodo at https://doi.org/10.5281/zenodo.14014974 [55].

5 Code availability

The code is publicly available via Zenodo at https://doi.org/10.5281/zenodo.14279347 [56] and via GitHub at https://github.com/Zheng-Meng/Dynamics-Reconstruction-ML.

References

[1] Wang, W.-X., Yang, R., Lai, Y.-C., Kovanis, V., Grebogi, C.: Predicting catastrophes in nonlinear dynamical systems by compressive sensing. Phys. Rev. Lett. **106**(15), 154101 (2011)

[2] Lai, Y.-C.: Finding nonlinear system equations and complex network structures from data: A sparse optimization approach. Chaos **31**, 082101 (2021)

[3] Haynes, N.D., Soriano, M.C., Rosin, D.P., Fischer, I., Gauthier, D.J.: Reservoir computing with a single time-delay autonomous Boolean node. Phys. Rev. E **91**, 020801 (2015) https://doi.org/10.1103/PhysRevE.91.020801

[4] Larger, L., Baylón-Fuentes, A., Martinenghi, R., Udaltsov, V.S., Chembo, Y.K., Jacquot, M.: High-speed photonic reservoir computing using a time-delay-based architecture: Million words per second classification. Phys. Rev. X **7**, 011015 (2017) https://doi.org/10.1103/PhysRevX.7.011015

[5] Pathak, J., Lu, Z., Hunt, B., Girvan, M., Ott, E.: Using machine learning to replicate chaotic attractors and calculate Lyapunov exponents from data. Chaos **27**, 121102 (2017)

[6] Lu, Z., Pathak, J., Hunt, B., Girvan, M., Brockett, R., Ott, E.: Reservoir observers: Model-free inference of unmeasured variables in chaotic systems. Chaos **27**, 041102 (2017)

[7] Pathak, J., Hunt, B., Girvan, M., Lu, Z., Ott, E.: Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. Phys. Rev. Lett. **120**, 024102 (2018) https://doi.org/10.1103/PhysRevLett.120.024102

[8] Carroll, T.L.: Using reservoir computers to distinguish chaotic signals. Phys. Rev. E **98**, 052209 (2018) https://doi.org/10.1103/PhysRevE.98.052209

[9] Nakai, K., Saiki, Y.: Machine-learning inference of fluid variables from data using reservoir computing. Phys. Rev. E **98**, 023111 (2018) https://doi.org/10.1103/PhysRevE.98.023111

[10] Roland, Z.S., Parlitz, U.: Observing spatio-temporal dynamics of excitable media using reservoir computing. Chaos **28**, 043118 (2018)

[11] Griffith, A., Pomerance, A., Gauthier, D.J.: Forecasting chaotic systems with very low connectivity reservoir computers. Chaos **29**, 123108 (2019)

[12] Tanaka, G., Yamane, T., Héroux, J.B., Nakane, R., Kanazawa, N., Takeda, S., Numata, H., Nakano, D., Hirose, A.: Recent advances in physical reservoir computing: A review. Neu. Net. **115**, 100–123 (2019)

[13] Fan, H., Jiang, J., Zhang, C., Wang, X., Lai, Y.-C.: Long-term prediction of chaotic systems with machine learning. Phys. Rev. Res. **2**, 012080 (2020) https://doi.org/10.1103/PhysRevResearch.2.012080

[14] Klos, C., Kossio, Y.F.K., Goedeke, S., Gilra, A., Memmesheimer, R.-M.: Dynamical learning of dynamics. Phys. Rev. Lett. **125**(8), 088103 (2020)

[15] Chen, P., Liu, R., Aihara, K., Chen, L.: Autoreservoir computing for multistep ahead prediction based on the spatiotemporal information transformation. Nat. Commun. **11**(1), 4568 (2020)

[16] Kong, L.-W., Fan, H.-W., Grebogi, C., Lai, Y.-C.: Machine learning prediction of critical transition and system collapse. Phys. Rev. Res. **3**, 013090 (2021)

[17] Patel, D., Canaday, D., Girvan, M., Pomerance, A., Ott, E.: Using machine learning to predict statistical properties of non-stationary dynamical processes: System climate, regime transitions, and the effect of stochasticity. Chaos **31**(3), 033149 (2021)

[18] Kim, J.Z., Lu, Z., Nozari, E., Pappas, G.J., Bassett, D.S.: Teaching recurrent neural networks to infer global temporal structure from local examples. Nat. Machine Intell. **3**(4), 316–323 (2021)

[19] Bollt, E.: On explaining the surprising success of reservoir computing forecaster of chaos? The universal machine learning dynamical system with contrast to VAR and DMD. Chaos **31**(1), 013108 (2021)

[20] Gauthier, D.J., Bollt, E., Griffith, A., Barbosa, W.A.: Next generation reservoir computing. Nat. Commun. **12**(1), 1–8 (2021)

[21] Zhai, Z.-M., Kong, L.-W., Lai, Y.-C.: Emergence of a resonance in machine learning. Phys. Rev. Res. **5**, 033127 (2023) https://doi.org/10.1103/PhysRevResearch.5.033127

[22] Yan, M., Huang, C., Bienstman, P., Tino, P., Lin, W., Sun, J.: Emerging opportunities and challenges for the future of reservoir computing. Nat. Commun. **15**(1), 2056 (2024)

[23] Zhai, Z.-M., Moradi, M., Kong, L.-W., Glaz, B., Haile, M., Lai, Y.-C.: Model-free tracking control of complex dynamical trajectories with machine learning. Nat. Commun. **14**(1), 5698 (2023)

[24] Zhai, Z.-M., Moradi, M., Kong, L.-W., Lai, Y.-C.: Detecting weak physical signal from noise: A machine-learning approach with applications to magnetic-anomaly-guided navigation. Phys. Rev. Appl. **19**(3), 034030 (2023)

[25] Liu, Z., Wang, Y., Vaidya, S., Ruehle, F., Halverson, J., Soljačić, M.,

Hou, T.Y., Tegmark, M.: Kan: Kolmogorov-Arnold networks. arXiv preprint arXiv:2404.19756 (2024)

[26] Moradi, M., Panahi, S., Bollt, E.M., Lai, Y.-C.: Data-driven model discovery with Kolmogorov-Arnold networks. arXiv preprint arXiv:2409.15167 (2024)

[27] Sonnewald, M., Lguensat, R., Jones, D.C., Dueben, P.D., Brajard, J., Balaji, V.: Bridging observations, theory and numerical simulation of the ocean using machine learning. Environ. Res. Lett. **16**(7), 073008 (2021)

[28] Cismondi, F., Fialho, A.S., Vieira, S.M., Reti, S.R., Sousa, J.M., Finkelstein, S.N.: Missing data in medical databases: Impute, delete or classify? Artif. Intell. Med. **58**(1), 63–72 (2013)

[29] Yeo, K.: Data-driven reconstruction of nonlinear dynamics from sparse observation. J. Comput. Phys. **395**, 671–689 (2019)

[30] McCann, K., Yodzis, P.: Nonlinear dynamics and population disappearances. Am. Nat. **144**(5), 873–879 (1994)

[31] Lorenz, E.N.: Deterministic nonperiodic flow. J. Atmos. Sci. **20**(2), 130–141 (1963)

[32] Vano, J., Wildenberg, J., Anderson, M., Noel, J., Sprott, J.: Chaos in low-dimensional Lotka-Volterra models of competition. Nonlinearity **19**(10), 2391 (2006)

[33] Kaplan, J., McCandlish, S., Henighan, T., Brown, T.B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., Amodei, D.: Scaling laws for neural language models. arXiv preprint arXiv:2001.08361 (2020)

[34] Gilpin, W.: Chaos as an interpretable benchmark for forecasting and data-driven modelling. arXiv preprint arXiv:2110.05266 (2021)

[35] Latora, V., Baranger, M.: Kolmogorov-Sinai entropy rate versus physical entropy. Phys. Rev. Lett. **82**(3), 520 (1999)

[36] Baptista, M., Ngamga, E., Pinto, P.R., Brito, M., Kurths, J.: Kolmogorov-Sinai entropy from recurrence times. Phys. Lett. A **374**(9), 1135–1140 (2010)

[37] Donoho, D.L.: Compressed sensing. IEEE Trans. Inf. Theory **52**(4), 1289–1306 (2006)

[38] Duarte, M.F., Eldar, Y.C.: Structured compressed sensing: From theory to applications. IEEE Trans. Signal Process. **59**(9), 4053–4085 (2011)

[39] Kim, J.Z., Bassett, D.S.: A neural machine code and programming framework for the reservoir computer. Nat. Mach. Intell. **5**(6), 622–630 (2023)

[40] Liu, J., Shen, Z., He, Y., Zhang, X., Xu, R., Yu, H., Cui, P.: Towards out-of-distribution generalization: A survey. arXiv preprint arXiv:2108.13624 (2021)

[41] Kong, L.-W., Brewer, G.A., Lai, Y.-C.: Reservoir-computing based associative memory and itinerancy for complex dynamical attractors. Nat. Commun. **15**(1), 4840 (2024)

[42] Du, Y., Luo, H., Guo, J., Xiao, J., Yu, Y., Wang, X.: Multi-functional reservoir computing. arXiv preprint arXiv:2409.16719 (2024)

[43] Zhai, Z.-M., Glaz, B., Haile, M., Lai, Y.-C.: Learning to learn ecosystems from limited data - a meta-learning approach. arXiv preprint arXiv:2410.07368 (2024)

[44] Zhang, Y., Gilpin, W.: Zero-shot forecasting of chaotic systems. arXiv preprint arXiv:2409.15771 (2024)

[45] Panahi, S., Lai, Y.-C.: Adaptable reservoir computing: A paradigm for model-free data-driven prediction of critical transitions in nonlinear dynamical systems. Chaos **34**, 051501 (2024)

[46] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Adv. Neural Inf. Process. Syst. **30** (2017)

[47] Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., Zhang, W.: Informer: Beyond efficient transformer for long sequence time-series forecasting. In: Proc. AAAI Conf. Artif. Intell., vol. 35, pp. 11106–11115 (2021)

[48] Wen, Q., Zhou, T., Zhang, C., Chen, W., Ma, Z., Yan, J., Sun, L.: Transformers in time series: A survey. arXiv preprint arXiv:2202.07125 (2022)

[49] Liu, Y., Hu, T., Zhang, H., Wu, H., Wang, S., Ma, L., Long, M.: itransformer: Inverted transformers are effective for time series forecasting. arXiv preprint arXiv:2310.06625 (2023)

[50] Xu, J., Wu, H., Wang, J., Long, M.: Anomaly transformer: Time series anomaly detection with association discrepancy. arXiv preprint arXiv:2110.02642 (2021)

[51] Foumani, N.M., Tan, C.W., Webb, G.I., Salehi, M.: Improving position encoding of transformers for multivariate time series classification. Data Min. Knowl. Discov. **38**(1), 22–48 (2024)

[52] Yıldız, A.Y., Koç, E., Koç, A.: Multivariate time series imputation with transformers. IEEE Signal Process. Lett. **29**, 2517–2521 (2022)

[53] Nogueira, F.: Bayesian Optimization: Open source constrained global optimization tool for Python (2014–). https://github.com/bayesian-optimization/BayesianOptimization

[54] Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. J. Mach. Learn. Res. **13**(2) (2012)

[55] Zhai, Z.-M.: Time series of chaotic systems. Zenodo. https://doi.org/10.5281/zenodo.14014974 (2023). https://doi.org/10.5281/zenodo.14014974

[56] Zhai, Z.-M.: Dynamics Reconstruction ML. Zenodo. https://doi.org/10.5281/zenodo.14279347 (2023). https://doi.org/10.5281/zenodo.14279347

6 Acknowledgment

We thank J.-Y. Huang for discussions. This work was supported by the Air Force Office of Scientific Research under Grant No. FA9550-21-1-0438 and by the Office of Naval Research under Grant No. N00014-24-1-2548.

7 Author contributions

Z.-M.Z., B.D.S., and Y.-C.L. designed the research project, the models, and methods. Z.-M.Z. performed the computations. Z.-M.Z., B.D.S., and Y.-C.L. analyzed the data. Z.-M.Z. and Y.-C.L. wrote the paper. Y.-C.L. edited the manuscript.

8 Competing interests

The authors declare no competing interests.

9 Additional information

Supplementary information available at Supplementary Information.

10 Correspondence

Correspondence and requests for materials should be addressed to Ying-Cheng Lai.