nature communications

Article

https://doi.org/10.1038/s41467-025-63019-8

Bridging known and unknown dynamics by transformer-based machine-learning inference from sparse observations

Received: 28 January 2025

Accepted: 7 August 2025

Published online: 28 August 2025

Check for updates

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

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 proposed hybrid machine-learning framework is tested using various prototypical nonlinear systems, demonstrating that the dynamics can be faithfully reconstructed from reasonably sparse data. The framework provides a paradigm of reconstructing complex and nonlinear dynamics in the 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[3–23], control[24], and signal detection[25]. Quite recently, Kolmogorov–Arnold networks (KANs)[26], 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[27] 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[28], 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[29,30]. 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

[1]School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ, USA. [2]Doctor of Physical Therapy Program, Tufts University School of Medicine, Phoenix, AZ, USA. [3]Department of Physics, Arizona State University, Tempe, AZ, USA. e-mail: Ying-Cheng.Lai@asu.edu

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[31] (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 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 smooth continuous time series. 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 quite 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 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. Specifically, the norm of the first derivative of the chaotic signals is ensured to stay below a predefined small threshold. 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 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:

$$S_r = \frac{L_s - L_s^O}{L_s - L_s^N}, \tag{1}$$

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 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 transformer-based 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[32] 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, providing a viable framework for reconstructing complex and nonlinear dynamics in situations where training data from the target system do not exist and the observations or measurements are insufficient.

Figure 1 highlights the challenge of reconstructing the dynamics from sparse data without training data. In particular, Fig. 1a 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 Fig. 1b, c. The time interval contains approximately two periods of oscillation, which defines a local frequency denoted as $f_{\text{local}} = 2$. As the signal is chaotic or random, the cutoff frequency $f_{\max}$ in the power spectrum can be higher than the frequency represented by the two oscillation cycles as shown. As a concrete example, we assume $f_{\max} = 3f_{\text{local}}$, so the Nyquist frequency is $f_{\text{Nyquist}} = 6f_{\text{local}}$. If the signal is sampled at the corresponding Nyquist time interval $\Delta T = 1/f_{\text{Nyquist}} = 1/12$, 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[31], as illustrated in Fig. 1b. 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 Fig. 1d–f.

Results

We test our approach on three nonlinear dynamical systems in the deployment phase: a three-species chaotic food-chain system[33], the

https://doi.org/10.1038/s41467-025-63019-8

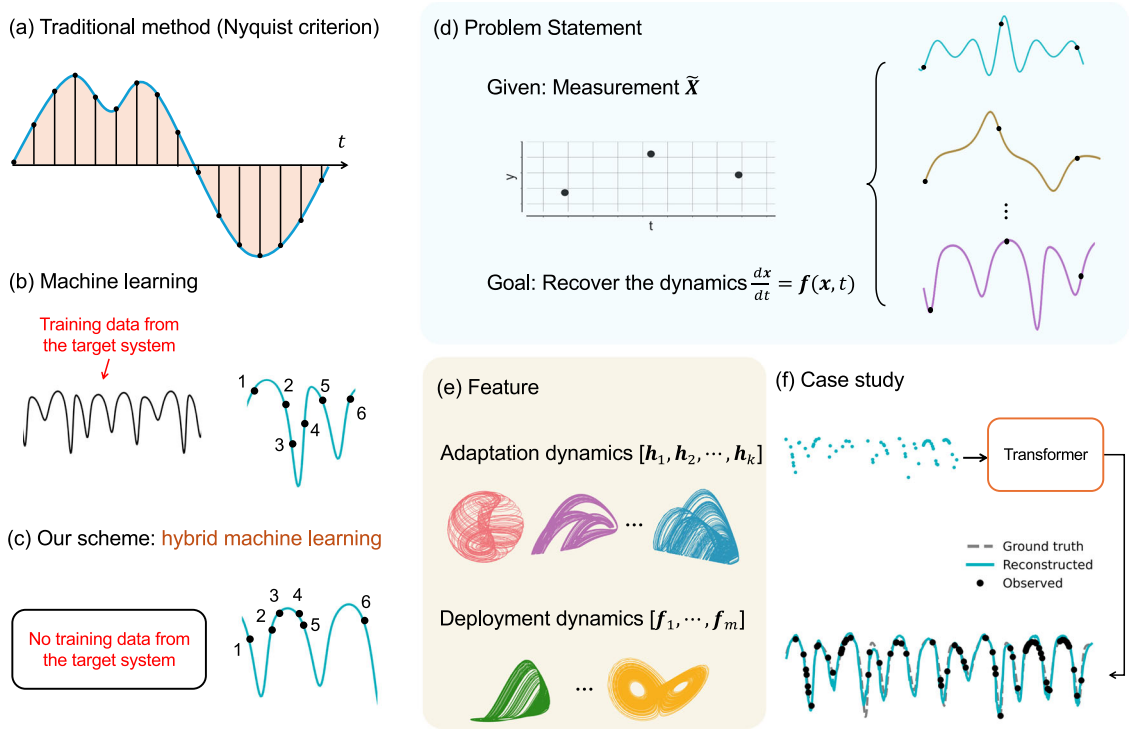

Fig. 1 | Dynamics reconstruction from random and sparse data. a The textbook case of a random time series sampled at a frequency higher than the Nyquist frequency. **b** Training data from the target system (left) and a segment of time series of six data points in a time interval containing approximately 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[31]. **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.

classic chaotic Lorenz system[34], and Lotka–Volterra system[35]. 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 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, 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 2a 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).

Dynamics reconstruction

The three species food-chain system[33] is described by

$$\begin{aligned}\frac{dR}{dt} &= R\left(1-\frac{R}{\mathrm{K}}\right)-\frac{\mathrm{x_c y_c} CR}{R+\mathrm{R_0}},\\ \frac{dC}{dt} &= \mathrm{x_c} C\left(\frac{\mathrm{y_c} R}{R+\mathrm{R_0}}-1\right)-\frac{\mathrm{x_p y_p} PC}{C+\mathrm{C_0}},\\ \frac{dP}{dt} &= \mathrm{x_p} P\left(\frac{\mathrm{y_p} C}{C+\mathrm{C_0}}-1\right),\end{aligned} \tag{2}$$

where R, C, and P are the population densities of the resource, consumer, and predator species, respectively. The system has seven parameters: K, $\mathrm{x_c}$, $\mathrm{y_c}$, $\mathrm{x_p}$, $\mathrm{y_p}$, $\mathrm{R_0}$, $\mathrm{C_0} > 0$. Figure 2b 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. 2b 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

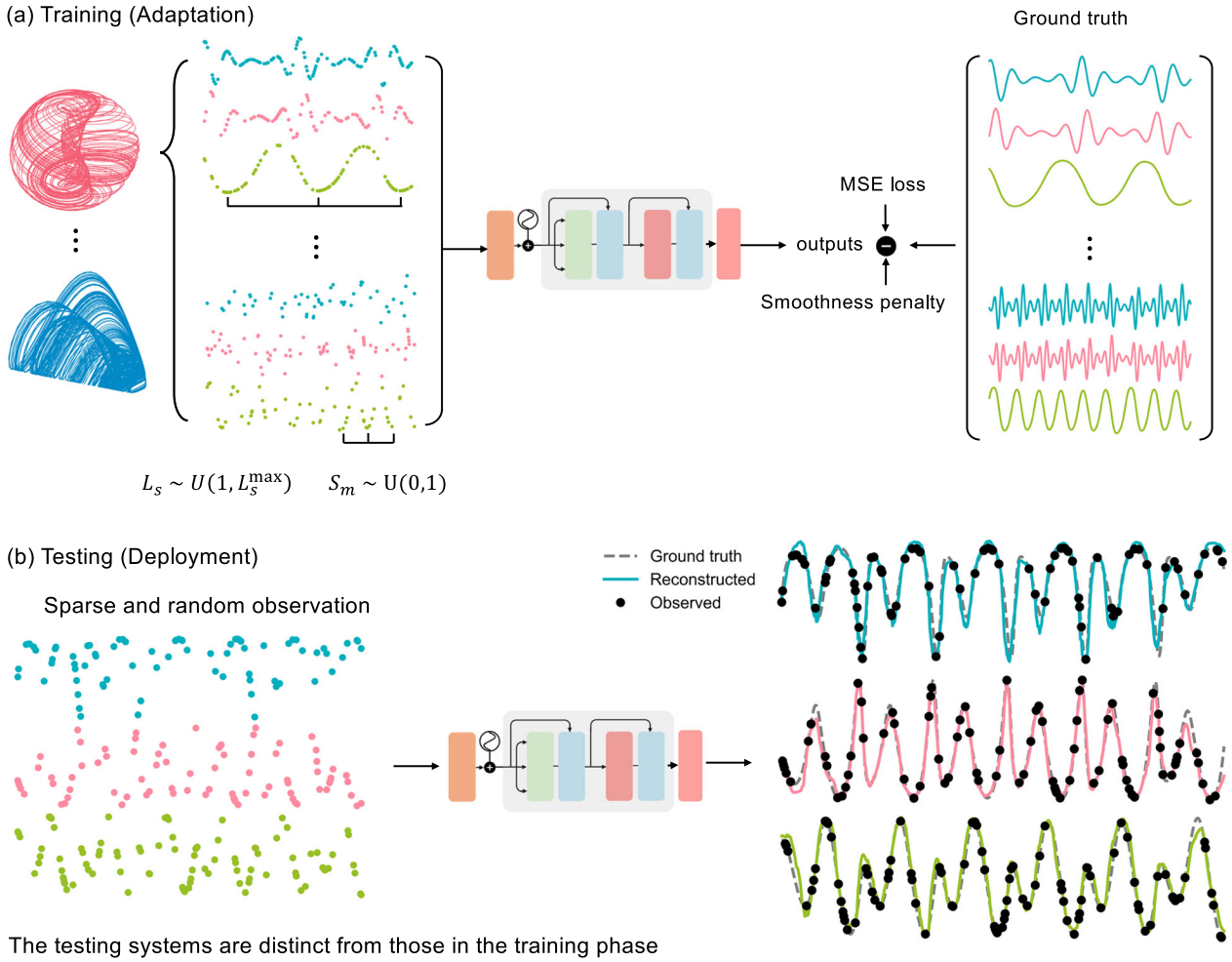

Fig. 2 | Illustration of the transformer-based dynamics reconstruction framework. a Training (adaptation) phase, where the model is trained on various synthetic chaotic systems, each divided into segments with uniformly distributed sequence lengths L_s and sparsity measure S_m. The data is masked before being input into the transformer, and the ground truth is used to minimize the MSE (mean squared error) 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.

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.

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 3a 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. Figure 3b, 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 3d 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.

Key features of dynamics reconstruction

Transformer has the ability to reconstruct the time series of previously unknown dynamical systems, particularly under high sparsity. This capability stems from the generalizability of the transformer during its training on sufficiently diverse chaotic systems with large data. 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 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. 4a, the MSE decreases with increasing k following a power-law trend with saturation, demonstrating that training our transformer-based framework on a

https://doi.org/10.1038/s41467-025-63019-8

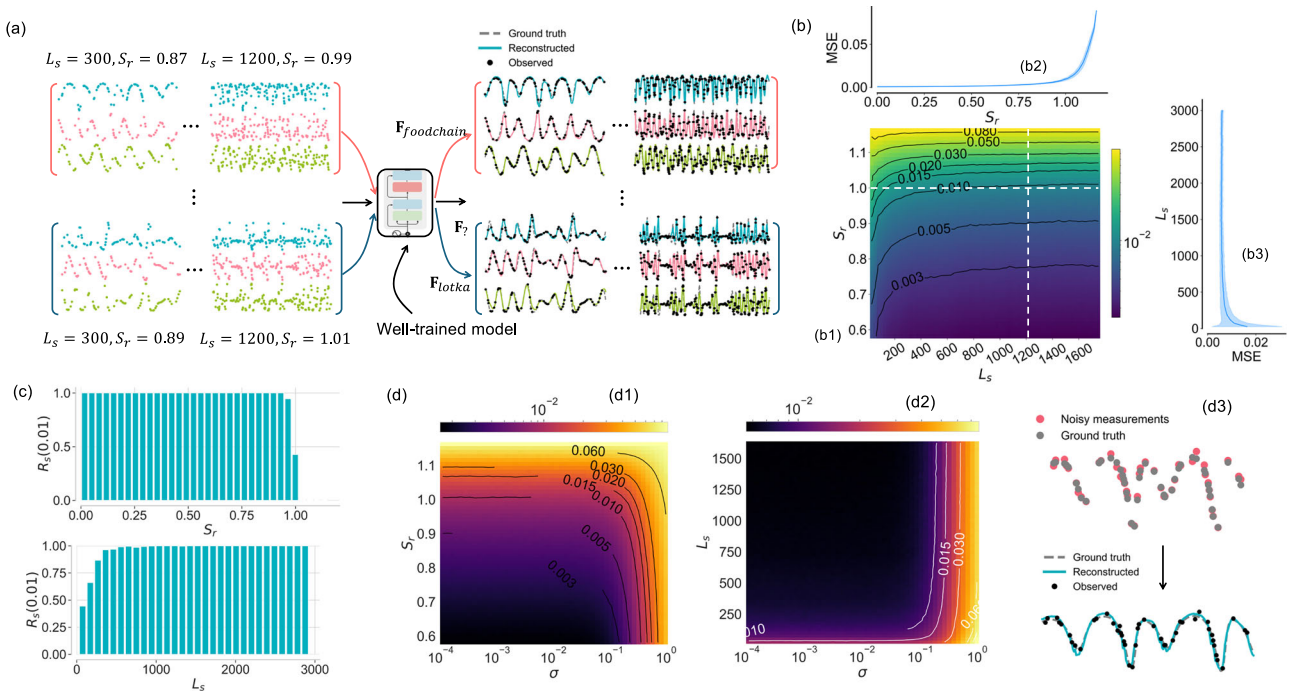

Fig. 3 | Performance of dynamics reconstruction. a Illustration of reconstruction results for the chaotic food-chain and Lotka–Volterra systems as the testing targets that the transformer has never been exposed to. For each target system, two sets of sparse measurements of different length L_s and sparsity 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.

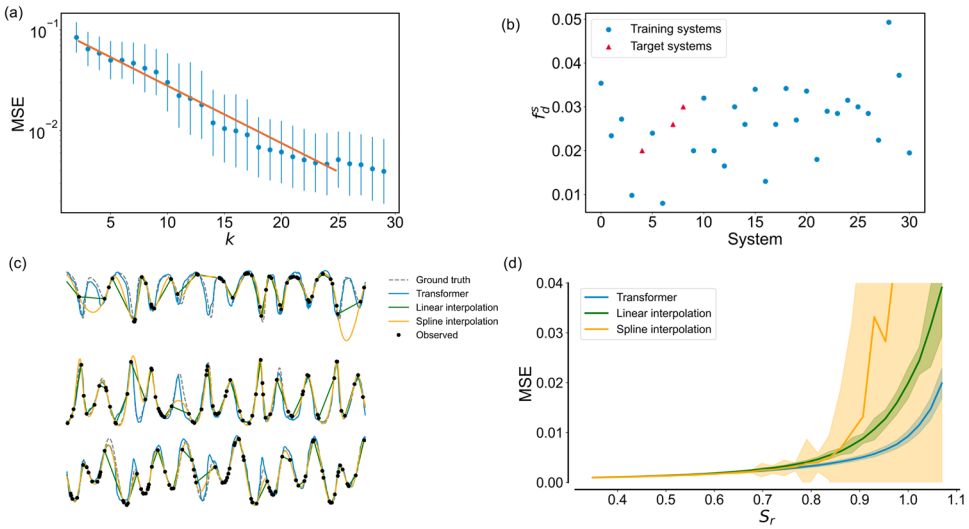

Fig. 4 | Demonstration of the capabilities of the transformer-based dynamics reconstruction. a Power-law decrease of MSE as the number of training systems k increase. **b** Scaled frequency f_s^d of training and target systems. **c** Example of a time series reconstructed by the transformer, compared 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. Error bars and shaded areas represent standard deviations across these realizations.

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 new systems from sparse observations. To demonstrate this, we have shown that our trained transformer performs well on 28 additional unseen target systems[36] (Supplementary Note 12).

To assess the frequency properties of the training and target systems more systematically, we analyze the power spectral density (PSD) for each system. The calculated PSDs are shown in Supplementary Note 3. Since each chaotic system is assigned a different sampling time Δs, chosen to ensure a smooth attractor while limiting the number of sample points, it is important to also compare the experimental

https://doi.org/10.1038/s41467-025-63019-8

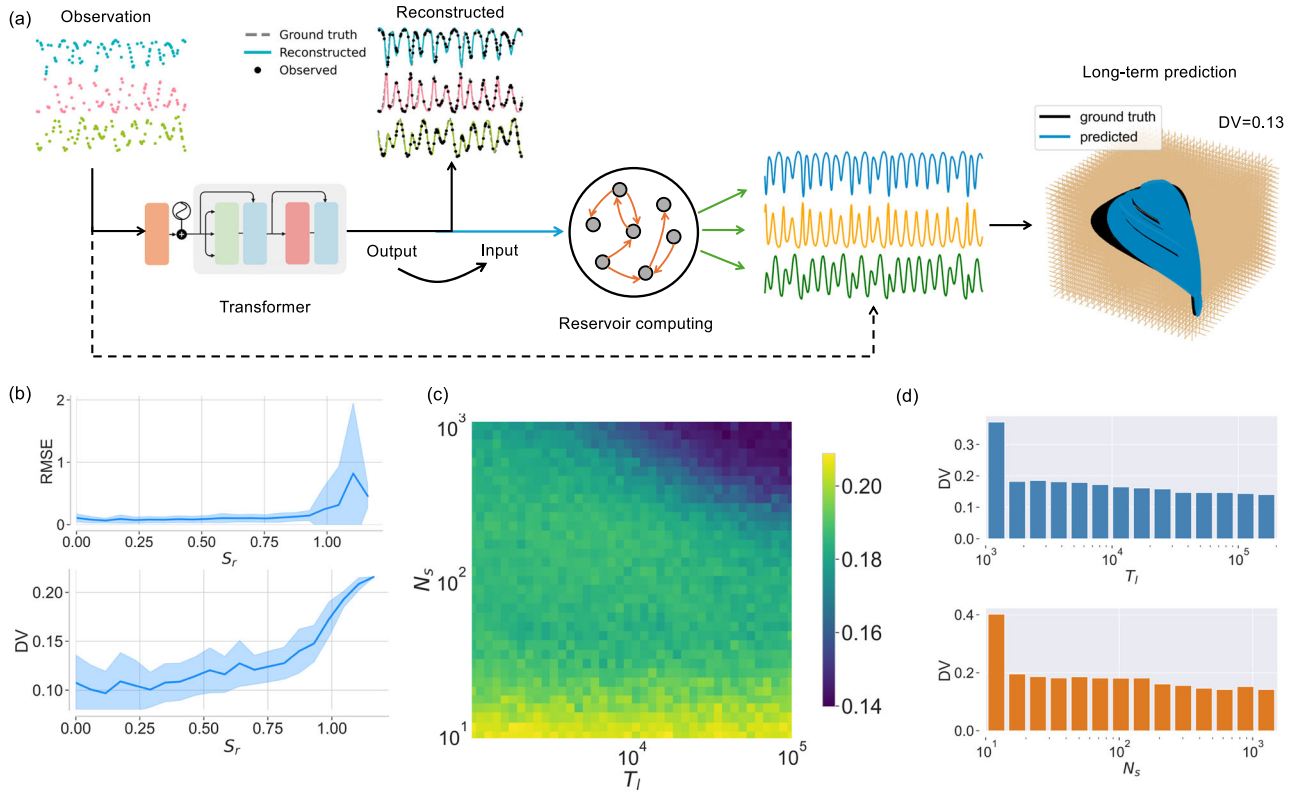

Fig. 5 | Reservoir-computing based long-term dynamics prediction. a An illustration of hybrid transformer/reservoir-computing framework. The time series reconstructed by the transformer is used to train the reservoir computer that generates time series of the target system of arbitrary length, leading to a reconstructed attractor that agrees with the ground truth. **b** RMSE and DV versus the sparsity parameter. Shaded areas represent the standard deviation. **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.

frequencies in a consistent manner. We normalize both the dominant frequency by the sampling time to achieve this. Specifically, we compute the scaled dominant frequency as $f_d^s = f_d \cdot \Delta s$ for each system, ensuring a fair comparison across the systems. As depicted in Fig. 4b, f_d^s exhibits a broad distribution across the systems. It can be seen that the target systems fall within the wide range of the training systems in both measures, contributing to the strong generalizability observed.

While our method is applicable across a wide range of sparsity, traditional techniques such as linear and spline interpolations 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 4c 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 4d shows the reconstruction performance across a range of sparsity levels for a fixed sequence length, $L_s = 2000$ (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[39,40]. 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 low to moderate, CS performs better than the transformer. However, when the observational sparsity is high, our hybrid machine-learning framework outperforms CS significantly (Supplementary Note 10). Hereafter, the term high sparsity in this study is used to mean not only a limited number of available observations but also the failure to reconstruct the system accurately at this level of sparsity by the traditional methods such as linear and spline interpolation, CS, and conventional machine-learning techniques.

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 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. The trained reservoir computer can predict or generate any length of time series of the target system, as exemplified in Fig. 5a. 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 5b 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. Figure 5c, 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.

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[24,41]. 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[42]. 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[43,44]. 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[45]. In addition, a quite recent work used well-defined, pretrained 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[46].

We have developed a hybrid transformer-based machine-learning framework to construct the dynamics of target systems, under two limitations: (1) the available observational data are random and sparse and (2) no training data from the system are available. We have addressed 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 have implemented 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 were 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 learning by reservoir computing, 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 (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., non-autonomous. 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 CS (Supplementary Note 10). Additional results from chaotic systems are presented in Supplementary Note 12. While the proposed machine-learning framework demonstrates promising performance on most of the target systems, we note that there are few cases where the transformer struggles to produce

accurate reconstruction due to the particularly complex dynamics and high sparsity observation (Supplementary Notes 11 and 12).

A question is whether our proposed framework can function as a universal. The observed power-law relationship between the reconstruction error and training set diversity suggests its potential to function as a foundation model. In addition, extensive experiments on a broad set of target systems have demonstrated the emergence of extrapolation capabilities, where the model utilizes global patterns in sparse observations to infer the underlying dynamics. However, establishing the theoretical base for this generalization remains challenging. Classical statistical learning theory (e.g., VC dimension, bias-variance trade-off) has proven inadequate in explaining the empirical success of over parameterized deep neural networks[47,48]. As stated in the Standford CRFM report[49], the community currently has a quite limited theoretical understanding of foundation models. Our current insights are empirical and based on physical intuition, warranting the development of a rigorous formalization of the inner mechanisms of the proposed hybrid machine-learning framework.

Moreover, there are constraints related to data segment length and sparsity. Specifically, performance deteriorates if the available data is insufficient or the sparsity exceeds a critical threshold. These thresholds are not universal across different systems, and determining them for a new target system is challenging. A reasonable solution is to supply longer observation windows or more segments when possible. In this work, we have provided such conditions through extensive experiments and the statistical results consistently reveal trends that are likely to generalize to other systems with similar levels of sparsity and sequence length. It is worth mentioning that this challenge in fact emerges commonly in time series analysis. For example, while weather forecasts can be extensively validated using historical data, it remains fundamentally uncertain whether predictions for the coming days will be accurate. Only statistical confidence can be offered.

Overall, our hybrid transformer/reservoir-computing framework has been demonstrated to be effective 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[16,50] 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.

Methods

Hybrid machine learning

Consider a nonlinear dynamical system described by

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{F}(\mathbf{x}(t), t), \quad t \in [0, T], \tag{3}$$

where $\mathbf{x}(t) \in \mathbb{R}^D$ is a D-dimensional state vector and $\mathbf{F}(\cdot)$ is the unknown velocity field. Let $\mathbf{X} = (\mathbf{x}_0, \cdots, \mathbf{x}_{L_s})^\top \in \mathbb{R}^{L_s \times D}$ 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

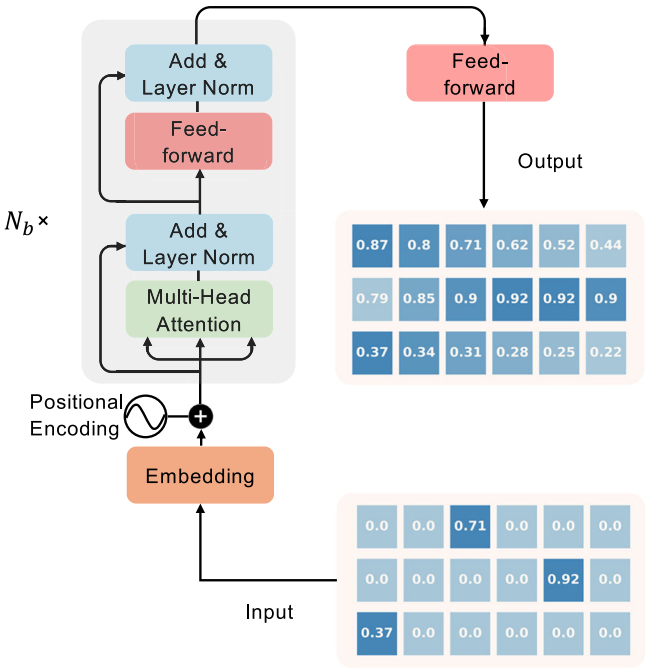

Fig. 6 | Transformer architecture. The transformer receives the sparse and random observation as the input and generates the reconstructed output. N_b refers to the number of transformer blocks. See text for a detailed mathematical description.

observational vector can be expressed as

$$\tilde{\mathbf{X}} = \mathbf{g}_\alpha(\mathbf{X})(1 + \sigma \cdot \Xi), \tag{4}$$

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:

$$X^O_{ij} = g_\alpha(X_{ij}) = \begin{cases} X_{ij}, & \text{if } X_{ij} \text{ is observed,} \\ 0, & \text{otherwise,} \end{cases} \tag{5}$$

with α representing the probability of matrix element X_{ij} being observed. In Eq. (4), 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 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[32], 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[51–53], anomaly detection[54], and classification[55]. 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[56]:

$$\mathbf{X}_p = \tilde{\mathbf{X}}\mathbf{W}_p + \mathbf{W}_b + \mathbf{PE}, \tag{6}$$

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 $1 \le \mathrm{pos} \le L_s^{max}$ and dimension $1 \le d \le D$, the encoding is given by

$$\mathbf{PE}_{pos,2d} = \sin\left(\frac{\mathrm{pos}}{10000^{2d/N}}\right), \tag{7}$$

$$\mathbf{PE}_{pos,2d+1} = \cos\left(\frac{\mathrm{pos}}{10000^{2d/N}}\right), \tag{8}$$

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 $L_s \times d_v$:

$$\mathbf{Q}_1 = \mathbf{X}_p\mathbf{W}_{\mathbf{Q}_1}, \quad \mathbf{K}_1 = \mathbf{X}_p\mathbf{W}_{\mathbf{K}_1}, \quad \mathbf{V}_1 = \mathbf{X}_p\mathbf{W}_{\mathbf{V}_1}, \tag{9}$$

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:

$$\mathbf{A}_{\mathbf{Q}_1,\mathbf{K}_1} = \mathrm{softmax}\left(\frac{\mathbf{Q}_1\mathbf{K}_1^{\mathrm{T}}}{\sqrt{d_k}}\right), \tag{10}$$

where $\mathbf{A}_{\mathbf{Q}_1,\mathbf{K}_1} \in \mathbb{R}^{L_s\times L_s}$. The softmax function normalizes the data with $\mathrm{softmax}(x_i) = \exp(x_i)/\sum_j \exp(x_j)$, 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 $\mathbf{V}_1$:

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

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:

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

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:

$$\mathbf{X}_{R1} = \mathrm{LayerNorm}(\mathbf{X_p} + \mathrm{Dropout}(\mathbf{O}_1)) \tag{13}$$

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

$$\mathbf{X}_{F1} = \max\left(0, \mathbf{X}_{R1}\mathbf{W}_{F_a} + \mathbf{b}_a\right)\mathbf{W}_{F_b} + \mathbf{b}_b, \tag{14}$$

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.

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

$$\mathcal{L} = \alpha_1\mathcal{L}_{\mathrm{mse}} + \alpha_2\mathcal{L}_{\mathrm{smooth}}, \tag{15}$$

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

$$\mathcal{L}_{\mathrm{mse}} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \tag{16}$$

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 $\mathcal{L}_{\mathrm{smooth}}$ 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:

$$\mathcal{L}_{\mathrm{laplacian}} = \frac{1}{n-2}\sum_{i=2}^{n-1}(\hat{y}_{i-1} + \hat{y}_{i+1} - 2\hat{y})^2, \tag{17}$$

and

$$\mathcal{L}_{\mathrm{tv}} = \frac{1}{n-1}\sum_{i=1}^{n-1}|\hat{y}_i - \hat{y}_{i+1}|. \tag{18}$$

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

$$\mathcal{L} = \mathcal{L}_{\mathrm{mse}} + \alpha_s(\mathcal{L}_{\mathrm{laplacian}} + \mathcal{L}_{\mathrm{tv}}). \tag{19}$$

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).

Computational setting

Unless otherwise stated, the following computational settings for machine learning are used. Given a target system, time series are

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

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 $\Delta_s = 1$ respectively, corresponding to approximately 1 over 30 ~ 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 3000 (corresponding to about 60 cycles of oscillation)—the limitation of input time series length. We apply Bayesian optimization[57] and a random search algorithm[58] to systematically explore and identify the optimal set of various hyperparameters. Two chaotic Sprott systems—Sprott_0 and Sprott_1—are used as validation systems to find the optimal hyperparameters and to train the final model weights, ensuring no data leakage from the testing systems. The optimized hyperparameters for the transformer are listed in Table 1. 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 min using one of the GPUs.

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 MSE_c:

$$R_s(\text{MSE}_c) = \frac{1}{n}\sum_{i=1}^{n}[\text{MSE} < \text{MSE}_c], \tag{20}$$

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

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]

$$\text{DV} \equiv \sum_{i=1}^{m_x}\sum_{j=1}^{m_y}\sum_{k=1}^{m_z}\sqrt{\left(f_{i,j,k} - \hat{f}_{i,j,k}\right)^2}, \tag{21}$$

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.

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.

Reporting summary

Further information on research design is available in the Nature Portfolio Reporting Summary linked to this article.

Data availability

The data generated in this study is publicly available via Zenodo at https://doi.org/10.5281/zenodo.14014974[59].

Code availability

The code is publicly available via Zenodo at https://doi.org/10.5281/zenodo.14279347[60] 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**, 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).
4. Larger, L. et al. High-speed photonic reservoir computing using a time-delay-based architecture: million words per second classification. *Phys. Rev. X* **7**, 011015 (2017).
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. et al. 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).
8. Carroll, T. L. Using reservoir computers to distinguish chaotic signals. *Phys. Rev. E* **98**, 052209 (2018).
9. Nakai, K. & Saiki, Y. Machine-learning inference of fluid variables from data using reservoir computing. *Phys. Rev. E* **98**, 023111 (2018).
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. et al. 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).
14. Klos, C., Kossio, Y. F. K., Goedeke, S., Gilra, A. & Memmesheimer, R.-M. Dynamical learning of dynamics. *Phys. Rev. Lett.* **125**, 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**, 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**, 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**, 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**, 013108 (2021).
20. Gauthier, D. J., Bollt, E., Griffith, A. & Barbosa, W. A. Next generation reservoir computing. *Nat. Commun.* **12**, 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).
22. Yan, M. et al. Emerging opportunities and challenges for the future of reservoir computing. *Nat. Commun.* **15**, 2056 (2024).
23. Panahi, S. et al. Machine learning prediction of tipping in complex dynamical systems. *Phys. Rev. Res.* **6**, 043194 (2024).
24. Zhai, Z.-M. et al. Model-free tracking control of complex dynamical trajectories with machine learning. *Nat. Commun.* **14**, 5698 (2023).
25. 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**, 034030 (2023).
26. Liu, Z. et al. KAN: Kolmogorov-Arnold networks. Preprint at https://arxiv.org/abs/2404.19756 (2024).
27. Panahi, S., Moradi, M., Bollt, E. M. & Lai, Y.-C. Data-driven model discovery with Kolmogorov-Arnold networks. *Phys. Rev. Res.* **7**, 023037 (2025).
28. Tan, L. & Jiang, J. *Digital Signal Processing: Fundamentals and Applications*, 3rd edn. (Academic Press, Inc., USA, 2018).
29. Sonnewald, M. et al. Bridging observations, theory and numerical simulation of the ocean using machine learning. *Environ. Res. Lett.* **16**, 073008 (2021).
30. Cismondi, F. et al. Missing data in medical databases: impute, delete or classify? *Artif. Intell. Med.* **58**, 63–72 (2013).
31. Yeo, K. Data-driven reconstruction of nonlinear dynamics from sparse observation. *J. Comput. Phys.* **395**, 671–689 (2019).
32. Vaswani, A. et al. Attention is all you need. *Advances in Neural Information Processing Systems* **30** (2017).
33. McCann, K. & Yodzis, P. Nonlinear dynamics and population disappearances. *Am. Nat.* **144**, 873–879 (1994).
34. Lorenz, E. N. Deterministic nonperiodic flow. *J. Atmos. Sci.* **20**, 130–141 (1963).
35. Vano, J., Wildenberg, J., Anderson, M., Noel, J. & Sprott, J. Chaos in low-dimensional Lotka-Volterra models of competition. *Nonlinearity* **19**, 2391 (2006).
36. Gilpin, W. Chaos as an interpretable benchmark for forecasting and data-driven modelling. Preprint at https://arxiv.org/abs/2110.05266 (2021).
37. Salgado, C.M., Azevedo, C., Proença, H. & Vieira, S.M. Missing data. *Second. Anal. Electron. Health Rec.* 143–162 (2016).
38. Junninen, H., Niska, H., Tuppurainen, K., Ruuskanen, J. & Kolehmainen, M. Methods for imputation of missing values in air quality data sets. *Atmos. Environ.* **38**, 2895–2907 (2004).
39. Donoho, D. L. Compressed sensing. *IEEE Trans. Inf. Theory* **52**, 1289–1306 (2006).
40. Duarte, M. F. & Eldar, Y. C. Structured compressed sensing: from theory to applications. *IEEE Trans. Signal Process.* **59**, 4053–4085 (2011).
41. Kim, J. Z. & Bassett, D. S. A neural machine code and programming framework for the reservoir computer. *Nat. Mach. Intell.* **5**, 622–630 (2023).
42. Liu, J. et al. Towards out-of-distribution generalization: a survey. Preprint at https://arxiv.org/abs/2108.13624 (2021).
43. Kong, L.-W., Brewer, G. A. & Lai, Y.-C. Reservoir-computing based associative memory and itinerancy for complex dynamical attractors. *Nat. Commun.* **15**, 4840 (2024).
44. Du, Y. et al. Multi-functional reservoir computing. *Phys. Rev. E* **111**, 035303 (2025).
45. Zhai, Z.-M., Glaz, B., Haile, M. & Lai, Y.-C. Learning to learn ecosystems from limited data - a meta-learning approach. Preprint at https://arxiv.org/abs/2410.07368 (2024).
46. Zhang, Y. & Gilpin, W. Zero-shot forecasting of chaotic systems. Preprint at https://arxiv.org/abs/2409.15771 (2024).
47. Zhang, C., Bengio, S., Hardt, M., Recht, B. & Vinyals, O. Understanding deep learning requires rethinking generalization. Preprint at https://arxiv.org/abs/1611.03530 (2016).
48. Belkin, M., Hsu, D., Ma, S. & Mandal, S. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proc. Natl. Acad. Sci. USA* **116**, 15849–15854 (2019).
49. Bommasani, R. et al. On the opportunities and risks of foundation models. Preprint at https://arxiv.org/abs/2108.07258 (2021).
50. 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).
51. Zhou, H. et al. Informer: beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35, 11106–11115 (AAAI Press, 2021).
52. Wen, Q. et al. Transformers in time series: a survey. In *Proc. 32nd International Joint Conference on Artificial Intelligence (IJCAI)* (2023).
53. Liu, Y. et al. iTransformer: inverted transformers are effective for time series forecasting. Preprint at https://arxiv.org/abs/2310.06625 (2023).
54. Xu, J., Wu, H., Wang, J. & Long, M. Anomaly transformer: time series anomaly detection with association discrepancy. In *Proc. International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=LzQQ89U1qm_ (2022).
55. 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**, 22–48 (2024).
56. Yıldız, A. Y., Koç, E. & Koç, A. Multivariate time series imputation with transformers. *IEEE Signal Process. Lett.* **29**, 2517–2521 (2022).
57. Nogueira, F. Bayesian optimization: open source constrained global optimization tool for Python. https://github.com/bayesian-optimization/BayesianOptimization (2014).
58. Bergstra, J. & Bengio, Y. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.* **13**, 281–305 (2012).
59. Zhai, Z.-M. Time series of chaotic systems. Zenodo. https://doi.org/10.5281/zenodo.14014974 (2023).
60. Zhai, Z.-M. Dynamics reconstruction ML. Zenodo. https://doi.org/10.5281/zenodo.14279347 (2023).

Acknowledgements

We thank J.-Y. Huang for some discussion. This work was supported by the Air Force Office of Scientific Research under Grant No.

FA9550-21-1-0438, by the Office of Naval Research under Grant No. N00014-24-1-2548, and by Tufts CTSI NIH Clinical and Translational Science Award (UM1TR004398).

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.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41467-025-63019-8.

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

Peer review information *Nature Communications* thanks the anonymous reviewers for their contribution to the peer review of this work. A peer review file is available.

Supplementary Information for

Bridging known and unknown dynamics by transformer-based machine-learning inference from sparse observations

Zheng-Meng Zhai, Benjamin D. Stern, and Ying-Cheng Lai

Corresponding author: Ying-Cheng Lai (Ying-Cheng.Lai@asu.edu)

CONTENTS

Supplementary Note 1: Additional background 2

Supplementary Note 2: Adaptation (training) and deployment (testing) systems 3

Supplementary Note 3: Effective bandwidth of chaotic systems 5

Supplementary Note 4: Reservoir computing 8

Supplementary Note 5: Hyperparameter optimization 9
Transformer hyperparameter optimization 9
Reservoir computing hyperparameter optimization 12

Supplementary Note 6: Further demonstration of dynamics reconstruction 13
Additional examples of dynamics reconstruction 13
Performance of system dynamics reconstruction 13
Performance of long-term "climate" reconstruction 14

Supplementary Note 7: Robustness test 16

Supplementary Note 8: A counter example 20

Supplementary Note 9: Nonautonomous systems 22

Supplementary Note 10: Reconstructing dynamics with traditional methods 25

Supplementary Note 11: Reconstructing dynamics of diverse chaotic systems 27

Supplementary Note 12: Reconstructing dynamics of additional chaotic systems 27

References 37

SUPPLEMENTARY NOTE 1: ADDITIONAL BACKGROUND

For dynamical systems reconstruction, machine-learning methods are basically complex approximators mapping inputs to the outputs, or fit relationships among the variables. Deep learning, in particular, is a purely data-driven method that relies on extensive datasets to discover the input-output relationships. This process inherently causes the model to overfit to a specific domain distributed about the training data. As a result, models trained on one dataset often struggle to adapt to unseen data from new distributions, a phenomenon known as out-of-distribution generalization or distribution (covariate) shift [1]. This limitation is particularly pronounced when learning nonlinear dynamics. The challenge is compounded not only by the nonstationary nature of the dynamics but also by changes in the system parameters, which can cause distribution shifts. More extreme shifts can occur when switching between different systems.

Recent years have witnessed a growth of interest in machine learning methods leading to predictive frameworks to estimate data values using unsupervised or supervised learning, due to their nonlinearity, flexibility, and ability to capture useful information embedded in the observed data. Traditional methods include K-nearest neighbor (KNN) [2], support vector machine (SVM) [3], and MissForest [4], etc., but deep learning models are able to deliver more accurate predictions, which include three major categories: recurrent neural network (RNN)-based, generative models, and self-attention based [5]. RNN-based methods [6], featured by their recurrent neural-network structure, are time-consuming and memory-constrained, making it difficult to capture the long-term dependencies within the time series. In addition, the RNN methods are commonly iterative, which inevitably introduces compounding errors through the process. Generative models include generative adversarial network (GAN) [7], variational autoencoder (VAE) [8], and diffusion models [9]. For example, the GAN-based method GAIN [7] takes a generator and a discriminator to predict the data values, where the former imputes the missing values conditioned on the observed data and outputs a reconstructed complete time series, and the latter determines which values are predicted and which belong to the observed data. Generative models provide an adversarial point of view to improve the performance of the generator, but it may suffer from non-convergence or mode collapse due to the loss formulation, and thus are difficult to train [10]. Self-attention-based methods, e.g., transformer models [11], focus on capturing the long-range dependencies and complex relationships within the time series. They excel at handling multi-dimensional data and at tackling input and output sequences of arbitrary length [12]. However, transformer models require substantial amounts of data for effective training and for avoiding overfitting. For instance, a vision transformer (ViT) usually yields low accuracies on mid-size datasets but attains high accuracies when trained on sufficiently large datasets (e.g., 14M-300M images) [13]. Transformer model is thus preferable when the resources are adequate due to its computational efficiency and scalability. Overall, most previous works focused on predicting unobserved values by assembling observable points in various ways, while marginalizing the underlying dynamics, which can be effective for specialized applications but is difficult to be generalized to complex dynamical systems.

It is worth mentioning the related problem of missing data imputation. Handling missing data can be approached through deletion or imputation. Deletion entails removing all entries with missing values [14]. This method is simple and effective when the missing data rate is low, typically less than 10% or 15%, so the removal procedure does not significantly affect the analysis [15]. However, as the missing rate increases, the deletion method becomes less effective and can lead to biased conclusions [16]. In contrast, imputation replaces missing data with estimated values us-

ing statistical or machine learning techniques [17]. Simple imputation methods, such as replacing missing values with the mean or median of the available values, can be readily implemented and are often used during data preprocessing. However, these methods can produce biased or unrealistic results, especially for high-dimensional datasets. Regression is another common imputation technique, where missing values are predicted using a regression model built from complete observations [18]. While effective, this method requires a large amount of data and does not account for any inherent variability. A more advanced approach is the Bayesian method, which treats missing values as unknown parameters drawn from an appropriate probability distribution [19]. This approach allows for the incorporation of prior knowledge about the data distribution and specifies a probabilistic model that captures the relationship between the observed and missing values.

SUPPLEMENTARY NOTE 2: ADAPTATION (TRAINING) AND DEPLOYMENT (TESTING) SYSTEMS

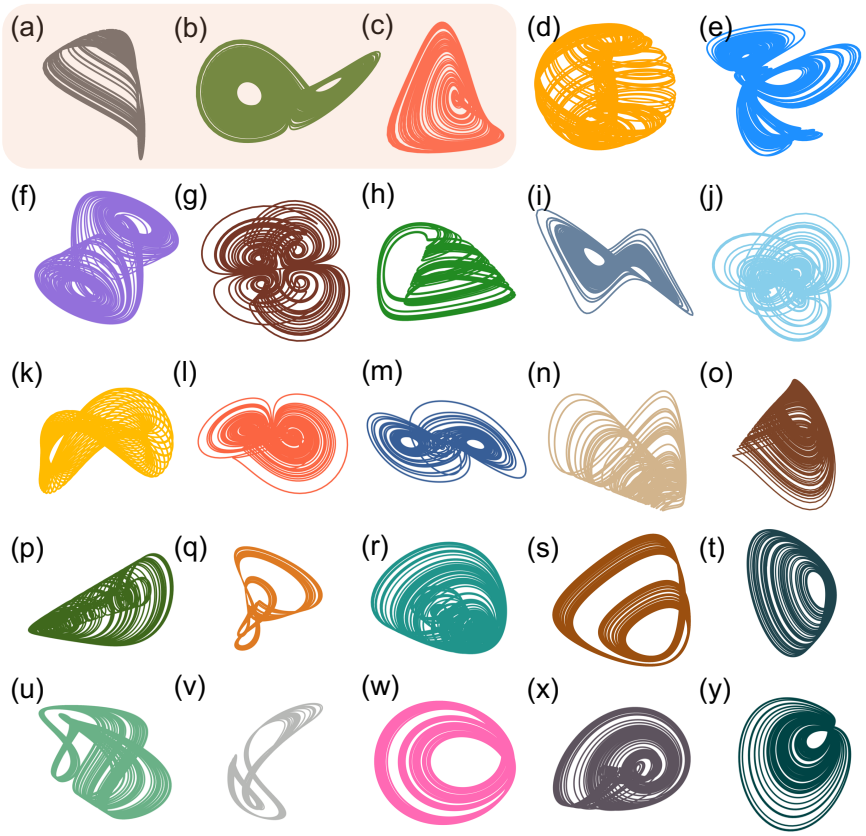

FIG. S1. The chaotic systems for adaptation learning during the training and the target systems for testing. (a-c) Target testing systems: food-chain, Lorenz, and Lotka-Volterra systems, respectively. Systems for learning: (d) Aizawa, (e) Bouali, (f) Chua, (g) Four wing, (h) Hastings-Powell, (i) Rikitake, (j) Wang, and (k-y) Sprott systems.

We provide the details of the systems employed during both the adaptation and deployment phases of our hybrid machine-learning framework as in Fig. S1, some of which are from Ref. [20].

The equations and parameters used to simulate the time series are listed in Tabs. S1 and S2. For the target testing systems, in addition to the three-species food chain system discussed in detail in the main text, two other systems are tested: the classic chaotic Lorenz and Lotka-Volterra systems.

The Lorenz system is described by

$$\begin{aligned} \frac{dx}{dt} &= \sigma_{lo}(y - x), \\ \frac{dy}{dt} &= x(\rho_{lo} - z) - y, \\ \frac{dz}{dt} &= xy - \beta_{lo} z, \end{aligned} \tag{S1}$$

where $\sigma_{lo} = 10$, $\rho_{lo} = 28$, and $\beta_{lo} = 2.67$ are parameters. The Lotka-Volterra system was originated from the modeling of predator-prey interactions and of certain chemical reactions [21]. For a system of N_c species with population P_i ($i = 1, \ldots, N_c$) competing for finite resources, the equations are given by

$$\frac{dP_i}{d} = r_i P_i (1 - \sum_{j=1}^{N_c} a_{ij} P_j), \tag{S2}$$

where r_i is the growth rate of species i and a_{ij} represents characterizes the interaction between species j and i. We use the four-species Lotka-Volterra model [21], with the parameter values:

$$r_i = \begin{bmatrix} 1 \\ 0.72 \\ 1.53 \\ 1.27 \end{bmatrix}, \quad a_{ij} = \begin{bmatrix} 1 & 1.09 & 1.52 & 0 \\ 0 & 1 & 0.44 & 1.36 \\ 2.33 & 0 & 1 & 0.47 \\ 1.21 & 0.51 & 0.35 & 1 \end{bmatrix}, \tag{S3}$$

While transformer has the advantage of flexible input data length, the dimension of the input vector needs to be fixed. Since the trained systems are three-dimensional, the target systems should have the same dimension. For the generated four-dimensional Lotka-Volterra time series, we take only data from the first three variables as the target. Since this operation discards the information of the fourth dynamical variable, the reconstruction task becomes even more challenging.

When obtaining time series data from different dynamical systems, it is necessary to carefully choose the sampling rate. The goal of our work is to reconstruct the dynamics of new target systems that are smooth dynamical systems. To maintain the smoothness, selecting an appropriate sampling rate Δs for data generation is essential. An excessively large Δs can produce jagged attractors that inadequately represent the long-term dynamics, while some too-small sampling rate leads to oversampling. We determine the appropriate Δs values for each system by ensuring the attractors remain smooth while limiting the number of sampled points, as listed in Tab. S3. Figure S2 presents examples of the attractors of the simulated target systems under varying sampling rates, where the orange-colored attractors represent the data ultimately utilized in training our hybrid machine-learning framework. Note that that the orange-colored attractors represent the ground-truth trajectories, i.e., the target smooth signals we aim to recover. They are not the input data to the machine: the machine-learning model does not receive these orange trajectories as input (unless the data is fully observed, i.e., with zero sparsity). Instead, the inputs typically

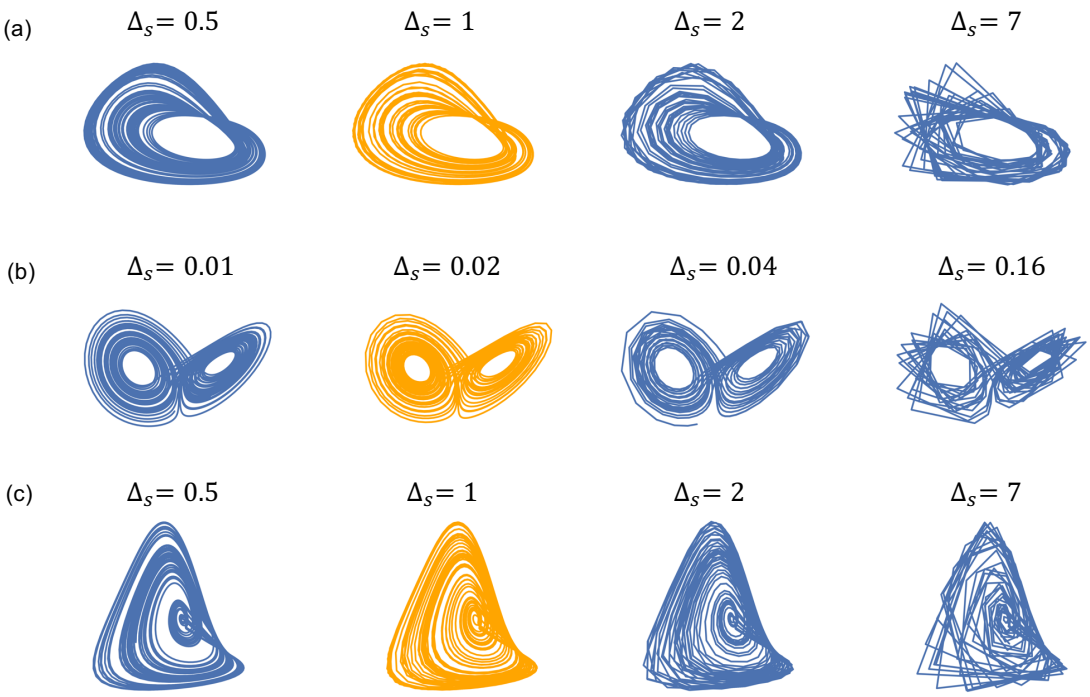

FIG. S2. Simulated attractors of several target systems with varying sampling rate Δs. (a-c) Target testing systems: chaotic food-chain, Lorenz, and Lotka-Volterra systems, respectively. The orange attractors are those reconstructed (ground-truth) using the sampling rate stated in the main text, which ensures that the attractors remain sufficiently smooth while maintaining a limited number of sampled points. The attractors in the right most column are simulated according to the Nyquist criterion.

consist of randomly and sparsely sampled (and visually jagged) trajectories, and our framework is designed to reconstruct the smooth ground-truth trajectory from them.

It is also worth noting that the the third column of Fig. S2 corresponds to doubling the selected sampling rate, which reduces the data availability to approximately 50%. The last column shows that a sampling rate 7-8 times higher than the selected rate yields about 14% data availability, which is consistent with the Nyquist criterion ($S_r = 1$). This demonstrates a nonlinear (concave) relationship between the sampling rate and sparsity. For example, in the chaotic food chain system, increasing Δs from 1 to 2 reduces the data availability to 50% (corresponding to $S_r = 0.58$), whereas further increasing it to 7 results in $S_r = 1$ (only a 0.42 increment).

SUPPLEMENTARY NOTE 3: EFFECTIVE BANDWIDTH OF CHAOTIC SYSTEMS

The Nyquist sampling theorem in signal processing establishes the conditions for reconstructing a continuous time band-limited signal from its discrete samples. Specifically, to perfectly reconstruct a band-limited signal, the sampling rate must be at least twice the highest frequency component present in the signal; this minimum sampling rate is known as the Nyqusit sampling rate. Mathematically, if a continuous-time signal $x(t)$ contains no frequency components above $f_{\max}$ Hz, it can be faithfully represented by discrete samples taken at a rate $f_s > 2f_{\max}$ Hz. It is

worth noting that the theorem requires perfect uniformly sampling with no noise in the measurement with ideal reconstruction filters.

For periodic signals such as sinusoidal waves, the bandwidth determination is straightforward, as the highest harmonic component determines the Nyquist rate. Chaotic systems present unique challenges for bandwidth determination due to their complex and broad spectra. In particular, chaotic systems typically contain continuous, broadband power spectra rather than discrete frequency components, and the power spectral density decays with the frequency, theoretically extending to infinite frequencies with diminishing power. In addition, different chaotic systems exhibit distinct spectral distributions. Since chaotic signals can contain energy across an infinitely wide frequency range in principle, a strict application of the Nyquist theorem would require an infinitely high sampling rate. However, this is neither practical nor necessary for applications. To address this challenge, we define the "effective bandwidth" based on the cumulative power spectral density, which defines a practical frequency limit that captures the significant dynamical information of the system, while excluding negligible high-frequency components.

Specifically, we define a "cutoff" frequency $f_{\max}$ as the frequency below which 98% of the total signal power is contained. We calculate the power spectral density (PSD) by employing Welch's method [22], where the time series $x(t)$ is divided into overlapping segments and a window function is applied to each segment. The PSD is then computed and averaged as:

$$P_{xx}(f) = \frac{1}{K}\sum_{i=1}^{K} |\sum_{n=0}^{N_x-1} w(n)x_i(n)e^{-j2\pi fn}|^2,$$

where K is the number of segments, N_x is the segment length, $w(n)$ is the window function, and $x_i(n)$ is the i-th segment of the signal. The dominant frequency f_d is computed as the maximum frequency across all dimensions, where in each dimension, it is identified as the frequency corresponding to the maximum value in PSD. Afterward, we compute the cumulative power distribution by integrating the PSD from zero frequency to each frequency point:

$$C(f) = \frac{\int_0^f P_{xx}(\xi)d\xi}{\int_0^\infty P_{xx}(\xi)d\xi}.$$

We identify the frequency $f_{\max}$ at which the cumulative power reaches our predefined threshold, i.e., $C(f_{\max}) = 0.98$. Table S3 shows the dominant frequency f_d, effective bandwidth $f_{\max}$, and the sampling rate Δs for each chaotic system in our study. The quantity L_s^N defined in the main text, where $L_s^N = 2f_{\max} \cdot T$, can be calculated as $L_s^N = 2f_{\max} \cdot L_s \cdot \Delta s$.

We analyze the power spectral density for each system, to assess the frequency properties of the training and target systems more systematically. The calculated PSDs are shown in Fig. S3, with the black dashed vertical line in each panel indicating the dominant frequency f_d. The three colors in each panel represent the PSDs of the three state variables of the system. A common feature across all systems in both the training and testing phases is that, due to their nonlinear and chaotic nature, they exhibit broadband spectra with no dominant single frequency. We find that the target systems share similar frequency components with the training systems, providing an intuitive explanation of the ability of the transformer to generalize to the target systems.

In addition, we emphasize that, for chaotic systems that typically exhibit broad and continuous power spectral densities, if one were to apply a sliding window and approximate the dominant frequency within each window, the instantaneous frequency content would vary significantly across

time. In high-frequency windows, sampling several points may not be sufficient, and random sampling worsens the situation. This variability, in combination with the random sparsity patterns, make Fourier interpolation or other classical methods inappropriate, providing further justification for our proposed framework.

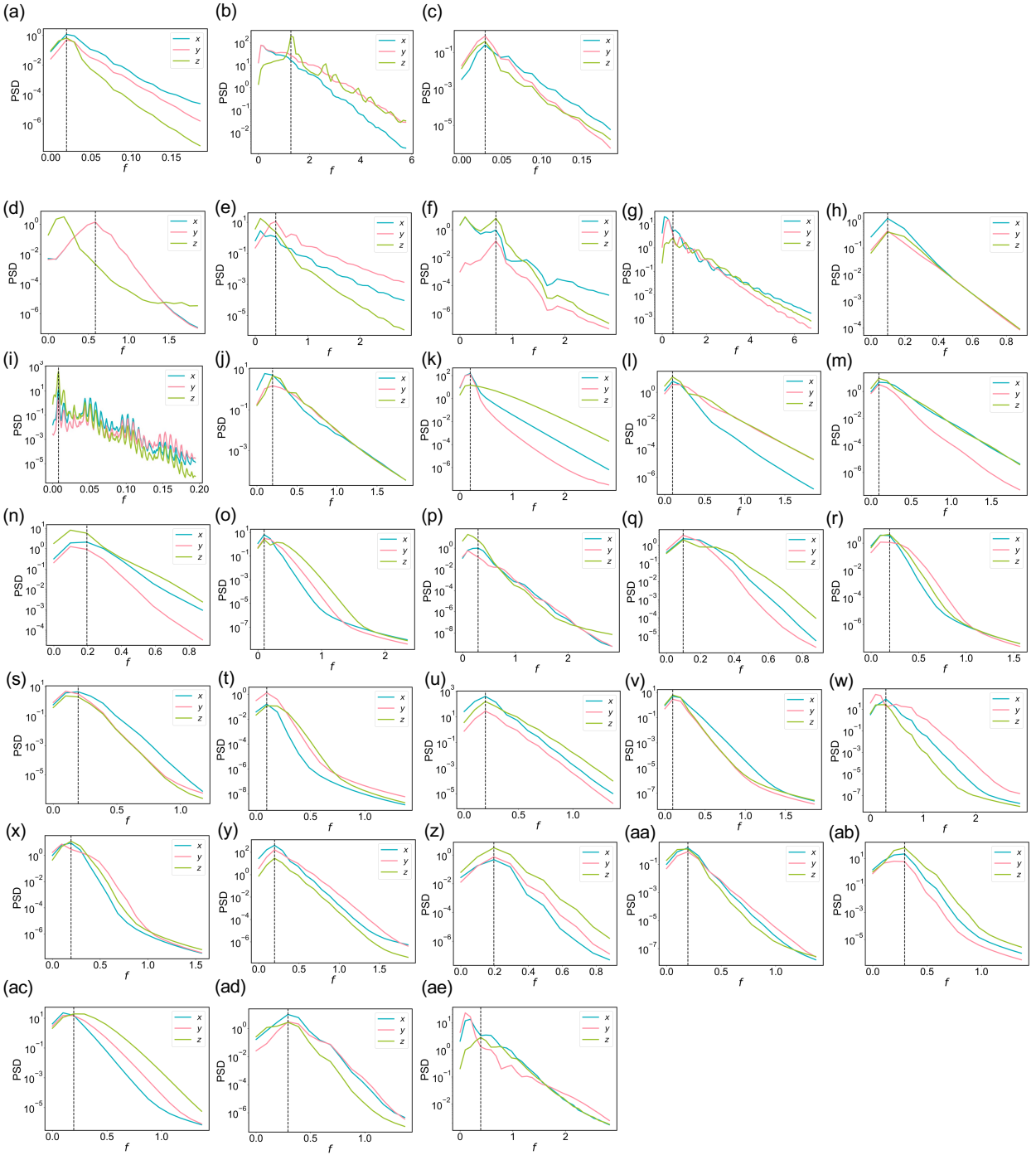

FIG. S3. Power spectral density for the training and the target systems for testing. (a-c) Target testing systems: food-chain, Lorenz, and Lota-Volterra systems, respectively (d) Aizawa, (e) Bouali, (f) Chua, (g) Four wing, (h) Hastings-Powell, (i) Rikitake, (j) Rossler, (k-ad) Sprott, (ae) Wang systems.

SUPPLEMENTARY NOTE 4: RESERVOIR COMPUTING

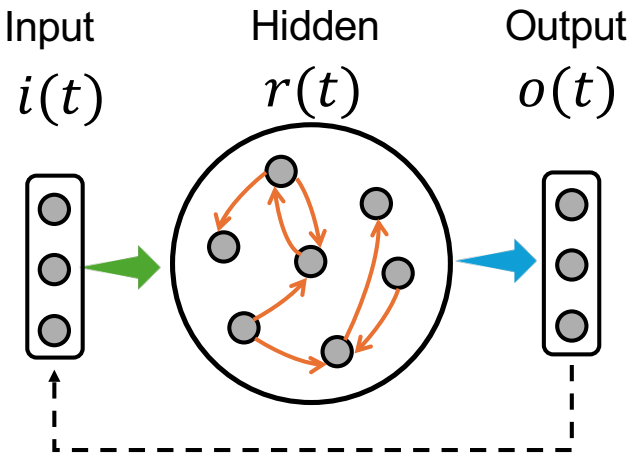

FIG. S4. Basic configuration of reservoir computing.

Reservoir computing, a class of recurrent neural networks, is a computational framework designed for efficient training and dynamical systems modeling. It consists of three layers: an input layer, a hidden recurrent layer, and an output layer. Unlike traditional recurrent neural networks where the weights associated all layers are trained, in reservoir computing the input weights and internal recurrent connections remain fixed after random initialization, while only the output weights are adjusted during training through a linear regression. This setting enables efficient training with performance similar to that of the conventional recurrent neural networks [23].

Figure S4 illustrates the basic structure of reservoir computing, where the input signal $\mathbf{i}(t)$, typically a low-dimensional vector, is mapped into the high-dimensional state space in the hidden layer with N_s nodes by the input matrix $\mathcal{W}_{\text{in}}$. Activated by the sequence of reservoir input signals $[\mathbf{i}(:,1), \mathbf{i}(:,2), \cdots, \mathbf{i}(:,t)]$, the hidden layer state $\mathbf{r}(t)$ is updated step-by-step according to

$$\begin{aligned}\mathbf{r}(t+1) = (1-\alpha_r)\cdot\mathbf{r}(t) + \\ \alpha_r \cdot \tanh\left[\mathcal{A}\cdot\mathbf{r}(t) + \mathcal{W}_{\text{in}}\cdot\mathbf{i}(t)\right],\end{aligned} \tag{S4}$$

where α_r is the leaking parameter that controls the speed of reservoir memory decays and the nonlinear activation function is of the hyperbolic tangent type. The dimensions of the input signal $\mathbf{i}(t)$, hidden state $\mathbf{r}$(t), and output signal $\mathbf{o}$(t) are denoted as D_i, N_s, and D_o, respectively. For three-dimensional chaotic systems, we have $D_o = D_i = 3$. The elements of the input matrix $\mathcal{W}_{\text{in}}$, which has the dimension of $N_s * D_i$, are generated uniformly in the range $[-\gamma_r, \gamma_r]$ prior to training. The elements of the hidden network $\mathcal{A}$ of the dimension $N_s * N_s$, are Gaussian random numbers generated before training, given the network size N_s, network link probability d_r, and spectral radius ρ_r. The reservoir network size N_s is typically much larger than input dimension D_i to ensure that it has sufficient capacity to learn the complex dynamics underlying the input signal. The output matrix $\mathcal{W}_{\text{out}}$ has the dimension of $D_o * N_s$. During the training phase, the reservoir state $\mathbf{r}(t)$ is updated according to Eq. (S4) and is concatenated into a matrix $\mathcal{R}$ of the dimension $N_s * T_l$, where T_l is the total training length. With the corresponding input concatenated matrix $\mathcal{U}$, the output matrix $\mathcal{W}_{\text{out}}$ can be obtained by Tikhonov regularization [24] as

$$\mathcal{W}_{\text{out}} = \mathcal{U}\cdot\mathcal{R}^{\intercal}(\mathcal{R}\cdot\mathcal{R}^{\intercal} + \beta_r\mathcal{I})^{-1}, \tag{S5}$$

where β_r is the regularization coefficient, $\mathcal{I}$ is the identity matrix of the dimension N_s. In the testing phase, the trained model takes the input $\mathbf{i}(t)$, updates the reservoir state $\mathbf{r}(t)$, and predicts the output $\mathbf{o}(t)$ via a linear combination of the reservoir state:

$$\mathbf{o}(\mathbf{t}) = \mathcal{W}_{\text{out}}\mathbf{r}(\mathbf{t}). \tag{S6}$$

As the ground truth data is no longer provided to the reservoir during the testing phase, the output from Eq. (S6) is fed back as the input at the next time step and used to update the hidden states. This process is repeated for any long time until the desired length of dynamical prediction is reached. An assumption of our hybrid machine-learning framework is 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 then predict the attractor or generate arbitrarily long time series of the target system.

SUPPLEMENTARY NOTE 5: HYPERPARAMETER OPTIMIZATION

Hyperparameter values are essential for machine-learning methods, which can have a significant effect on the performance. We optimize the hyperparameters of transformer and reservoir computing through random search and Bayesian optimization, respectively.

Transformer hyperparameter optimization

Random search is a simple yet effective method for hyperparameter optimization. Unlike grid search where hyperparameters are tested exhaustively over each possible combination, random search allows sampling hyperparameter values in a larger space. Each time, it randomly selects a combination of the hyperparameters from the search space, evaluates the performance on the validation dataset, and chooses the best performance combination. Due to the large number of hyperparameters the transformer contains, random search is practically effective in terms of the trade-off between the computational cost and performance.

We optimize eleven hyperparameters for transformer through random search, with their values listed in Table. I in the main text. To gain an intuitive understanding of how these hyperparameter values impact the performance, we test the target systems as they are not used in training and therefore are not used during the hyperparameter optimization process. We compare the prediction performance between the two cases where the hyperparameters are optimized and not. Two chaotic Sprott systems, Sprott0 and Sprott1, are used to find optimal hyperparameters. As transformer is typically "data-greedy," we not only prepare multiple systems, but for each system, we also provide a sufficient amount of data for training. To demonstrate statistically how much data is required from each training system, we use varying training data length D_l and evaluate the model performance on target systems. Representative results are shown in Fig. S5. It can be seen that the MSEs are large when D_l is small, but after reaching a threshold about $D_l = 10^5$, the performance improves dramatically. Notably, the sparsity affects the model performance as well: if it is large, increasing D_l hardly yields any improvement. This is reasonable, as merely increasing the training data for a specific system does not enhance the transformer's ability to generalize. In fact, it may even hinder the capacity of the model to learn new dynamics due to overfitting on a specific system.

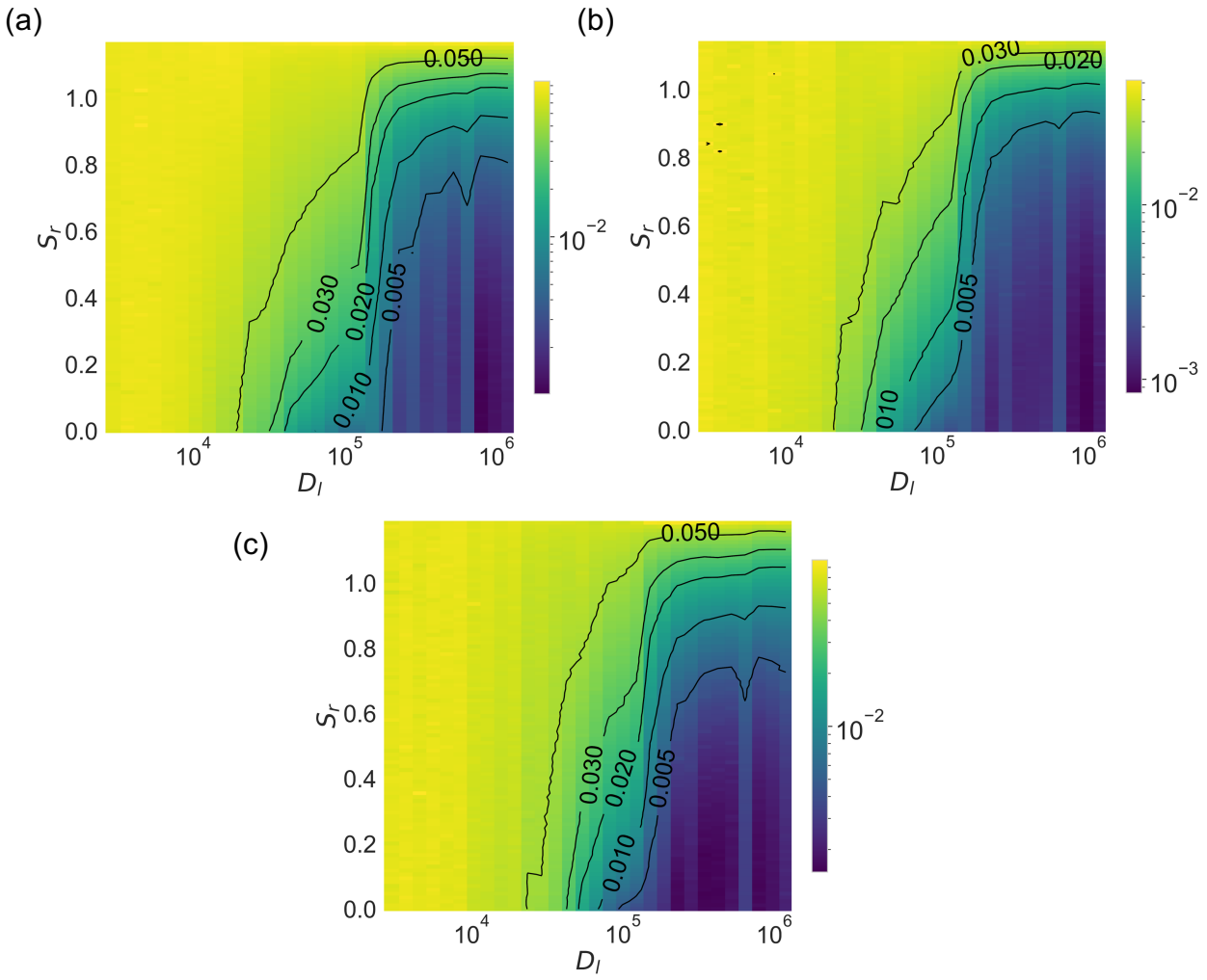

FIG. S5. Effects of the training length D_l on dynamics reconstruction performance, for different sparsity S_r. (a-c) Color-coded ensemble-averaged MSE values in the plane of (D_l, S_r) for the food-chain, Lorenz, and Lotka-Volterra systems, respectively. Each value of the averaged MSE is obtained using 50 statistical realizations. Increasing the training length can often dramatically improve the prediction performance.

To demonstrate the effectiveness of random search in finding optimal or near-optimal hyperparameter sets, we present the results on the effects of typical hyperparameters on the performance, as depicted in Fig. S6, where all hyperparameters are fixed except one. We train the transformer multiple times to evaluate the effect of varying this hyperparameter on the averaged MSE across trained systems. The results show that in most cases, the hyperparameters determined by random search are indeed optimal or near-optimal for training the transformer.

Smoothness penalty α_s is also a crucial hyperparameter in the training procedure. We study the effect of this penalty on reconstruction quality through two examples in Figs. S7(a) and S7(b), where training with smoothness yields smoother reconstructions. To evaluate the impact of α_s on the reconstruction accuracy, we test four values: $\alpha_s = 0$ (no smoothness), $\alpha_s = 0.1$ (our setting), $\alpha_s = 0.5$, and a large penalty $\alpha_s = 1$. In each case, the dynamics reconstruction framework is trained 50 times and evaluated on the target system. The results in Figs. S7(c) and S7(d) show that excessive smoothness leads the model to learn overly coarse-grained dynamics, resulting in a high

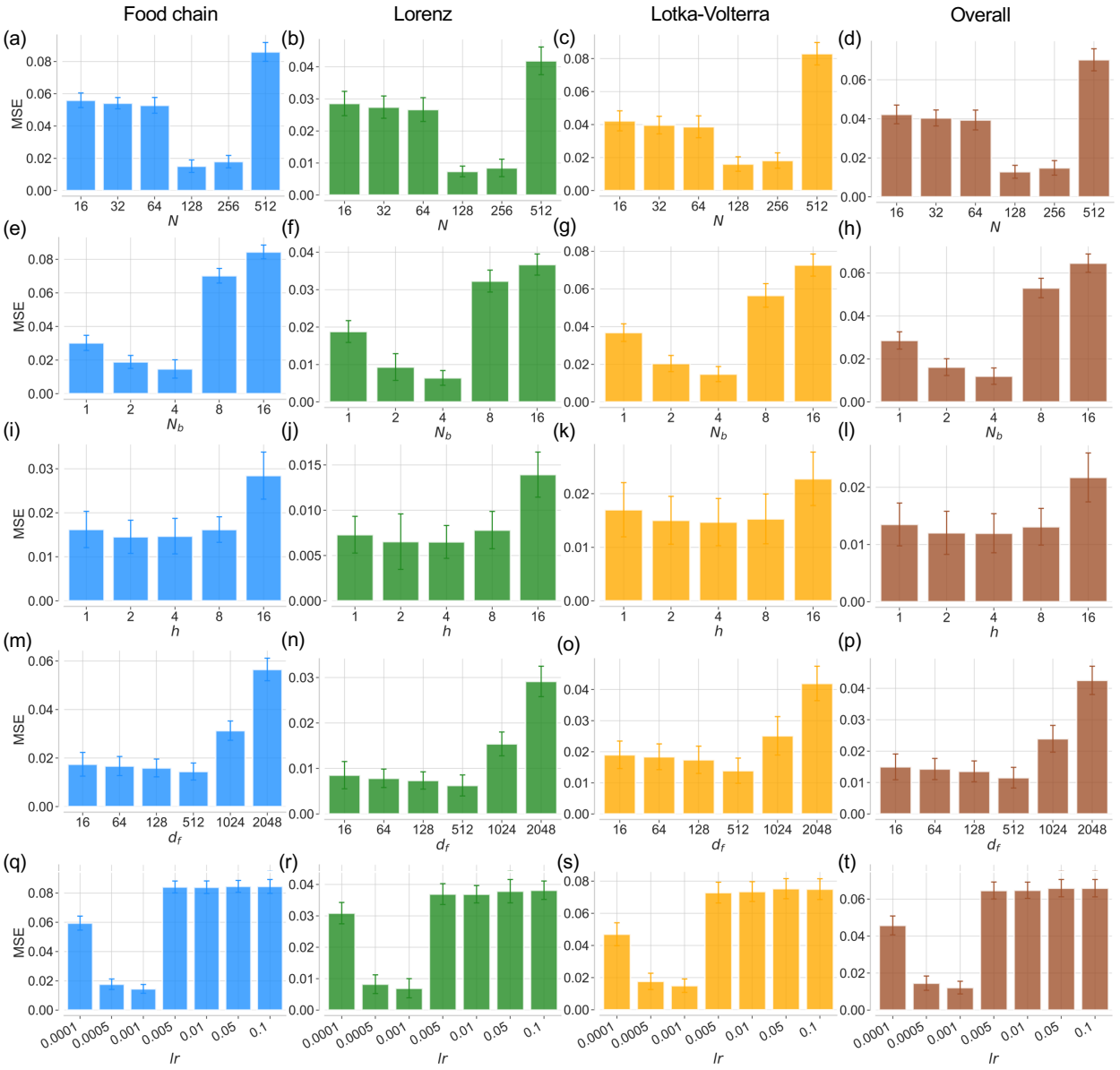

FIG. S6. Effects of representative hyperparameters on performance. Columns 1-4 are the results for the food chain, Lorenz, Lotka-Volterra systems, and the averaged performance of these systems, respectively. Rows 1-5 present the performance with respect to the following hyperparameters: input embedding dimension N, number N_b of transformer blocks, transformer heads h, feedforward neurons d_f, and learning rate lr, respectively. To reduce the statistical fluctuations, 50 transformer models are trained to obtain the averaged MSE, and error bars represent standard deviation across them.

MSE. Conversely, when the smoothness is small or absent, the MSE remains similarly low. We conclude that the smoothness penalty should be chosen appropriately to ensure both accurate and smooth reconstruction.

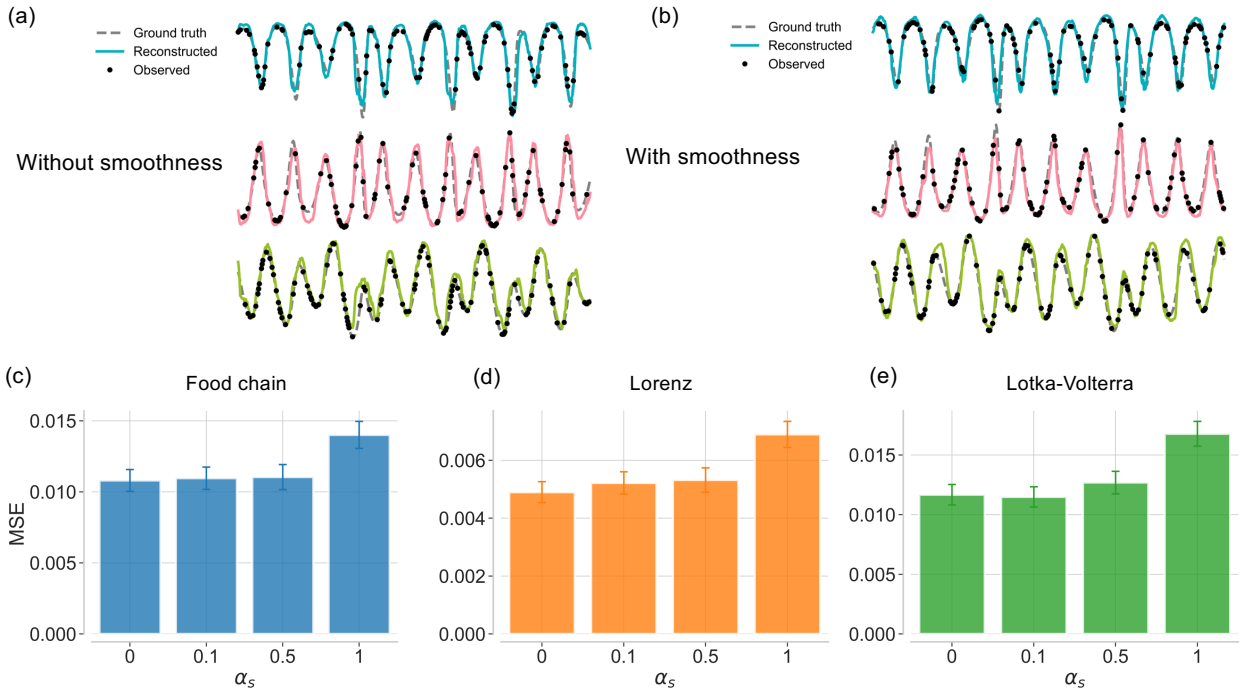

FIG. S7. Effects of the smoothness penalty α_s on dynamics reconstruction performance. (a) An example of training without smoothness and (b) an example of training with smoothness. (c-e) Performance for the food chain, Lorenz, and Lotka-Volterra systems, respectively, with respect to the smoothness penalty parameter α_s. To reduce the statistical fluctuations, 50 transformer models are trained to obtain the averaged MSE, and error bars represent standard deviations. The smoothness penalty should not be set too large in order to ensure both smooth and accurate reconstruction.

Reservoir computing hyperparameter optimization

A key issue of reservoir computing is that its performance often depends sensitively on the hyperparameters. We apply Bayesian optimization from Python (*bayesian-optimization*) to find the optimal values of the following hyperparameters: leakage parameter α_r, regularization coefficient β_r, input matrix scaling factor γ_r, spectral radius ρ_r, link probability d_r determining the network matrix $\mathcal{A}$, and the amplitude σ_r of the noise added to the input signal. The optimized hyperparameters are listed in Tab. S4. We evaluate the performance of the reservoir-computing based long-term "climate" or attractor reconstruction with respect to varying the training length T_l and network size N_s. Figures S8(a) and S8(b) show the performance versus the two hyperparameters for the Lorenz and Lotka-Volterra systems, respectively. It can be seen that a combination of larger network and longer training length will lead to better reconstruction performance, while increasing the network size for small T_l will degrade the performance.

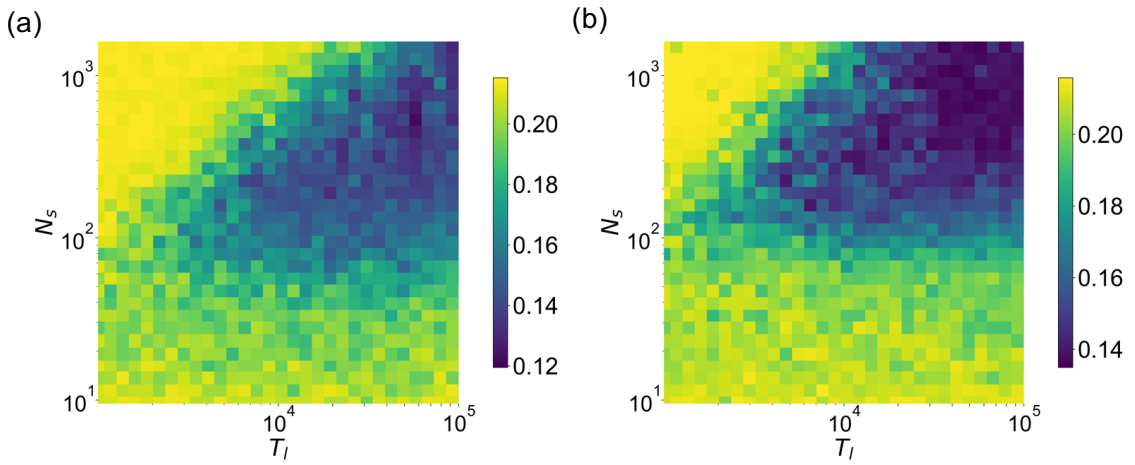

FIG. S8. Effects of reservoir-computing training length and network size on performance. (a,b) Ensemble-averaged MSE obtained from 50 realizations versus changes in the training length T_l and network size N_s for the chaotic Lorenz and Lotka-Volterra systems, respectively. Appropriate parameter ranges lead to satisfactory performance.

SUPPLEMENTARY NOTE 6: FURTHER DEMONSTRATION OF DYNAMICS RECONSTRUCTION

Additional examples of dynamics reconstruction

In the main text, we presented examples of the dynamics reconstruction of the food chain system. To demonstrate that the transformer-based framework performs for varying sparsity S_r and sequence length L_s, we present additional examples for the food chain system, as well as the Lorenz and Lotka-Volterra systems, as shown in Figs. S9 and S10, with results comparable to those in the main text: as L_s increases and S_r decreases, the performance of the model on target systems continuously improves.

Performance of system dynamics reconstruction

In addition to the food chain system in the main text, we further evaluate the performance of the other two target systems. Figure S11 illustrates the reconstruction performance for the Lorenz and Lotka-Volterra as characterized by MSE [see Eq. 16] and reconstruction stability [Eq. 20]. For reconstruction stability analyses, L_s is fixed at 1,200 (about 25 cycles of oscillation) when varying S_r, and S_r is set to 0.93 when varying L_s. For the Lotka-Volterra system, we set S_r to 0.89 to ensure reconstruction stability reaching one given sufficient L_s. These results further demonstrate the ability of the transformer to reconstruct the dynamics of new systems from sparse observational data.

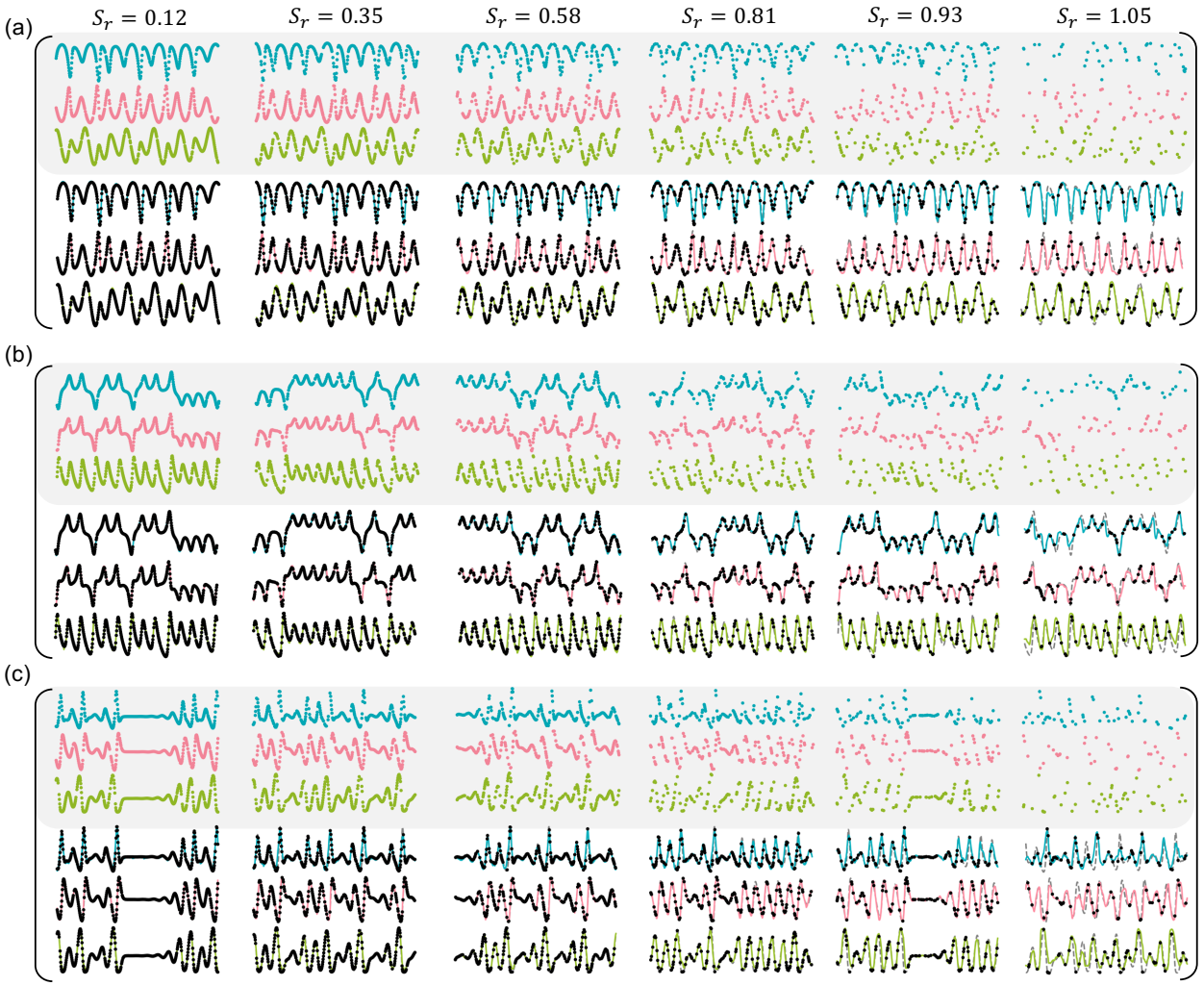

FIG. S9. Additional examples of dynamics reconstruction by transformer for different levels of data sparsity. (a-c) Reconstructed time series for the food chain, Lorenz, and Lotka-Volterra systems, respectively, where the shaded areas represent observable points and the unshaded areas show the corresponding reconstruction results. The sequence length is fixed at $L_s = 1200$ (about 25 cycles of oscillation), but only the first 500 data points are displayed for clarity of presentation.

Performance of long-term "climate" reconstruction

Reservoir computing utilizes and trains on the reconstructed time series from the transformer and is able to generate arbitrarily long time series with the same statistical properties as those of the original system. Here we present results from long-term reconstruction of the Lorenz and Lotka-Volterra systems, where the time series reconstructed from the transformer are for $S_r = 0.93$ and 0.89, respectively. Figures S12(a) and S12(b) show segments of the ground truth for the Lorenz and Lotka-Volterra systems, respectively. Figures S12(c) and S12(d) show the time series generated from reservoir computing for the respective systems. The reconstructed (predicted) and ground truth chaotic attractors of the two systems are depicted in Figs. S12(e) and S12(f), respectively. It can be seen that the reservoir computer can capture the "climate" through the transformer output, providing indirect confirmation that the transformer has successfully reconstructed the

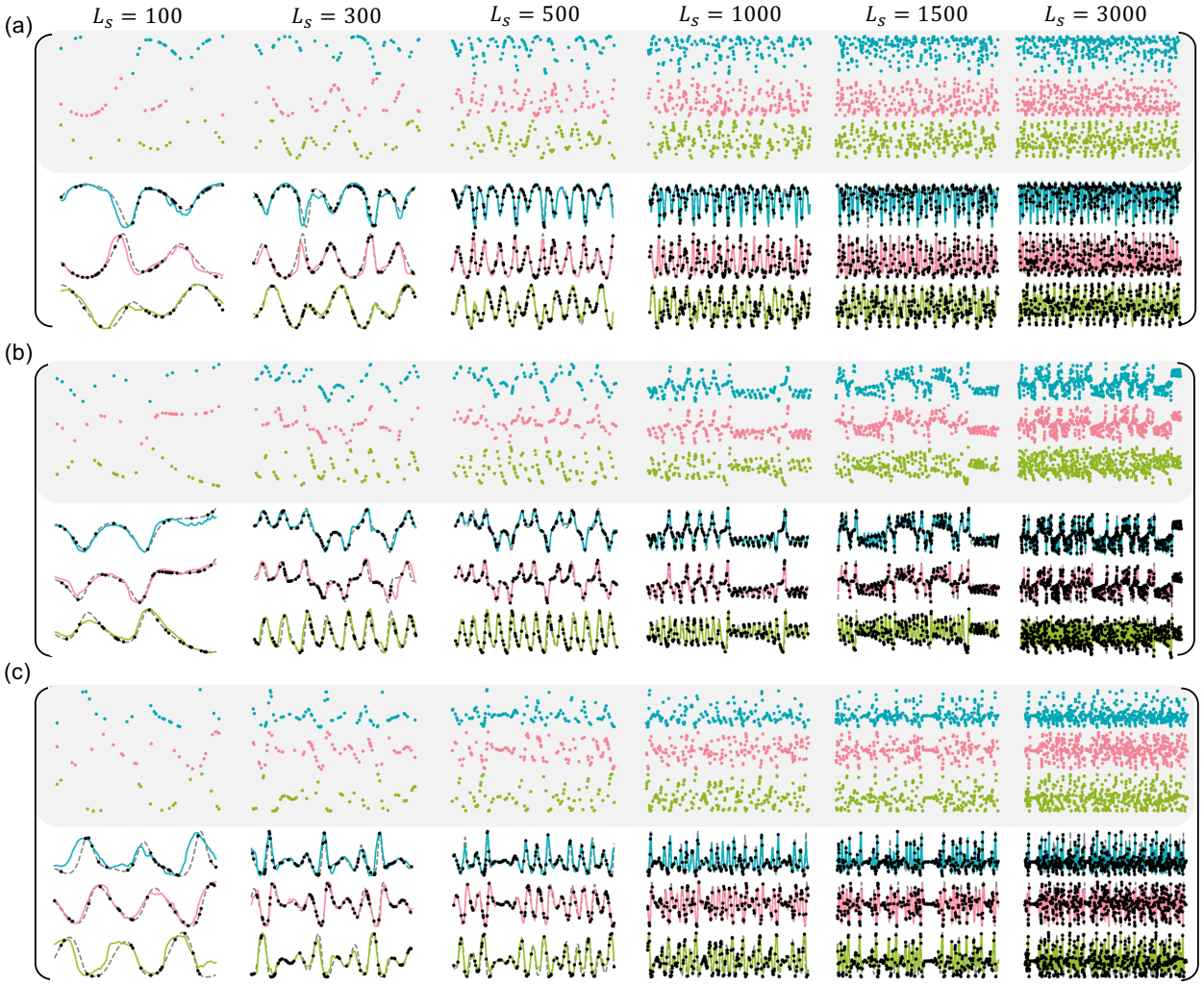

FIG. S10. Additional examples of dynamics reconstruction with respect to varying input sequence length L_s. (a-c) Reconstructed time series for the food chain, Lorenz, and Lotka-Volterra systems, respectively, where the shaded areas represent the observable points and the unshaded areas show the corresponding reconstruction. The sparsity value is fixed at around $S_r = 0.93$.

dynamics. Figures S12(g) and S12(h) show the DV versus varying S_r. In general, as the observations become more sparse, the attractor reconstruction performance as measured by DV gradually degrades. For $S_r > 1.0$, the performance deteriorates rapidly.

In addition, the reservoir computer plays a crucial role in our hybrid machine learning framework, whose roles are: (1) validating the reliability of reconstructed time series by the transformer, (2) refining the output dynamics of the transformer, and (3) generating arbitrarily long time series of the target system. This is particularly necessary when the observational data are highly sparse, where the transformer alone becomes insufficient to accurately reconstruct the full dynamics. In such cases, the reservoir computer learns from the transformer's output, refines and consolidates the reconstructed dynamics.

Figure S13 presents examples of dynamics reconstruction for three target systems: the chaotic food chain, Lorenz, and Lotka-Volterra systems, all under high sparsity conditions. The observed sparse data from these systems are processed by a transformer which is well trained on data from

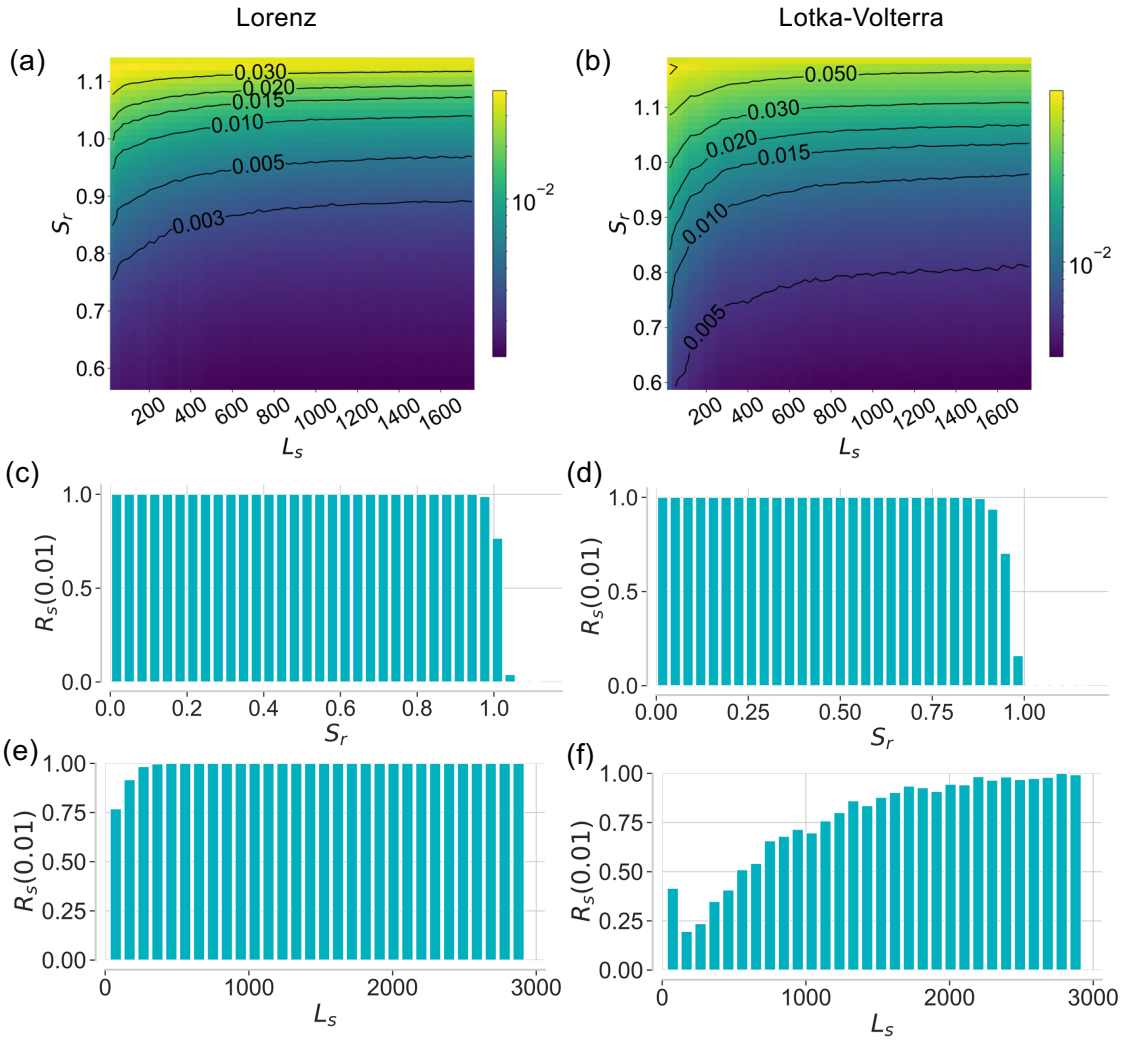

FIG. S11. Dynamics reconstruction performance. The two columns are the results from the Lorenz and Lotka-Volterra systems, respectively. The three rows from top to down present the color-coded averaged MSE in the parameter plane (L_s, S_r), the reconstruction stability R_s versus S_r and L_s for MSE threshold 0.01, respectively. Reconstruction stability and averaged MSE are calculated from 400,000 data points (corresponding to around 8,000 cycles of oscillation) in each case.

other systems. While the output of the transformer approaches the ground truth, there is a mismatch in the fine structural details, as shown in Fig. S13. To overcome this difficulty, we train an independent reservoir computer for each target system, using the transformer-reconstructed time series as the training data.

SUPPLEMENTARY NOTE 7: ROBUSTNESS TEST

We present the results of robustness test against multiplicative and additive noise, for the three target systems, as shown in Figs. S14 and S15. It can be seen that the transformer based dynamics reconstruction framework is robust against multiplicative and additive noise of amplitude below 10^{-1} and $10^{-1.5}$, respectively. The sequence length is $L_s = 1200$ (around 25 cycles of oscillation)

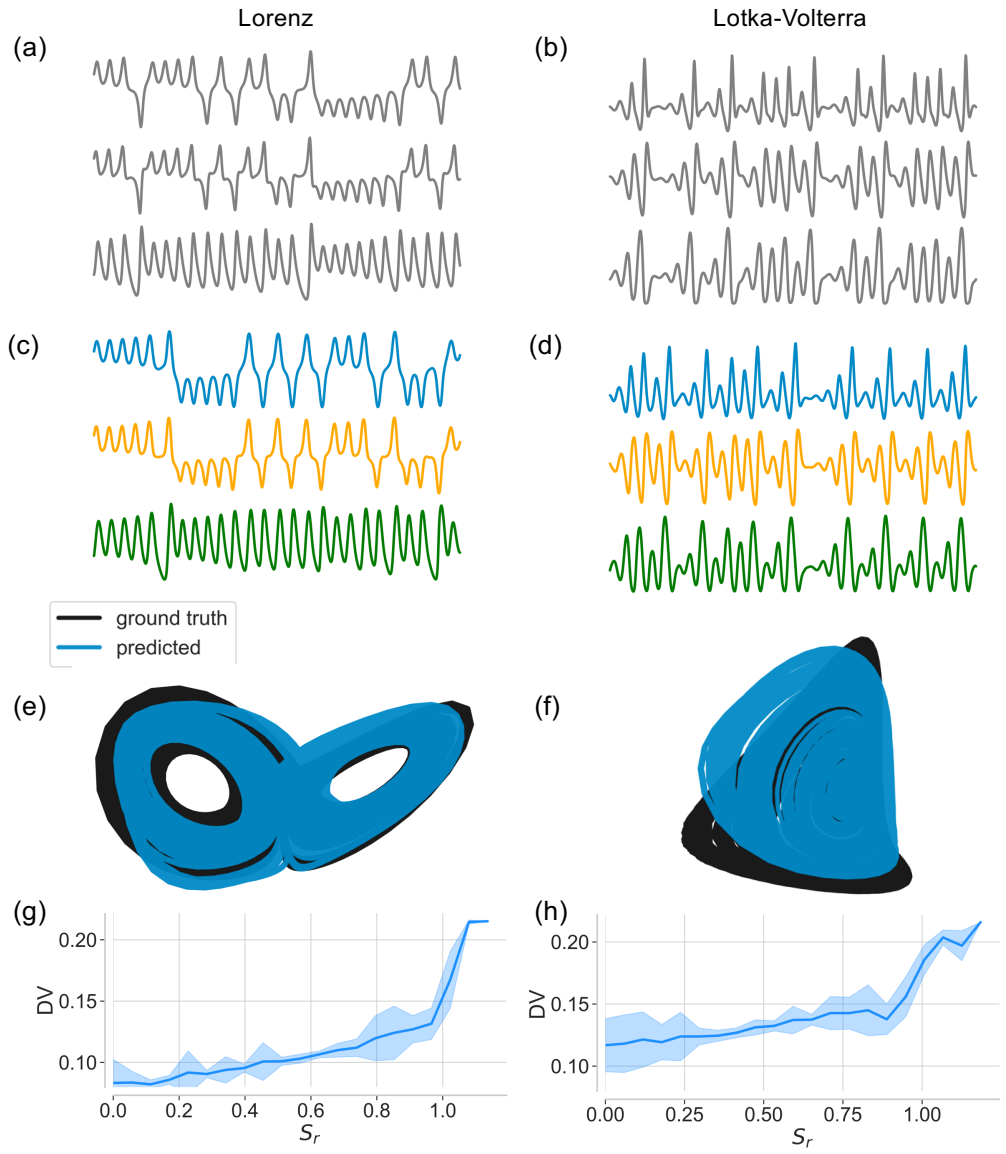

FIG. S12. Attractor reconstruction by reservoir computing. The first and second columns are for the Lorenz and Lotka-Volterra systems, respectively. (a,b) Segments of ground-truth time series, (c,d) short-term predictions by reservoir computing, (e,f) reconstructed long-term attractors as compared with the true attractors, and (g,h) ensemble-averaged DV versus sparsity S_r, obtained from 50 independent realizations.

for varying S_r and noise level σ, while $S_r = 0.93$ for varying L_s.

A pertinent question is, can the proposed framework recover correctly similar dynamics from nearby values of the bifurcation parameter? To address this question, we consider the chaotic food chain system with a varying bifurcation parameter as an example. Specifically, the environmental carrying capacity parameter K in Eq. (2) in the main text is a commonly used bifurcation parameterr [25]. An example of the bifurcation diagram is shown in Fig. S16(a). We use three K values: 0.96, 0.97, 0.98, collect high sparsity data with $S_r = 0.93$ from each of the parameter values, and reconstruct the time series through the transformer. The reconstructed time series for each parameter value are then fed into a reservoir computer to generate the corresponding attrac-

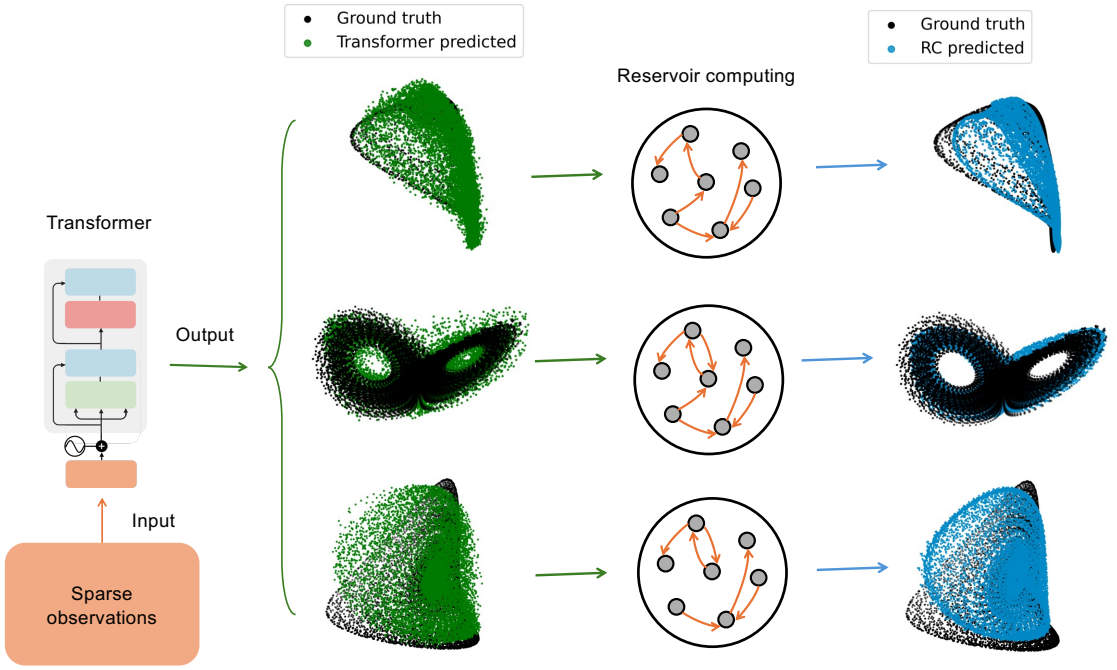

FIG. S13. Attractor reconstruction by the transformer alone and the hybrid transformer/reservoir computing framework. Sparse observations from target systems are provided to the transformer, which reconstruct the time series and present the attractors as green dots. The reconstructed time series by the transformer are then used to train independent reservoir computers, generating attractor predictions shown as the blue dots. The attractors, from top to bottom, are from the chaotic food chain, Lorenz, and Lotka-Volterra systems, at the sparsity level 0.93.

tors for comparison with the ground truth. The results are shown using the confusion matrix in Fig. S16(b), where the diagonal entries exhibit the smallest values in each row, indicating that each predicted attractor closely aligns with the ground-truth attractor at the same parameter, while the remaining entries characterize the distinction from the attractors at the other parameter values. The results demonstrate that our framework is capable of successfully recovering the dynamics from the target system at different values of the bifurcation parameter. Nonetheless, as expected, if the parameter difference is too small, the framework would fail in recovering the similar dynamics, because the underlying dynamics are statistically indistinguishable. For instance, the dynamics from $\mathrm{K} = 0.98$ to $\mathrm{K} = 0.981$ are nearly identical. In such cases, even if the full dynamics were observable, it would be practically difficult for any machine-learning model to distinguish between them, especially with sparse data.

The observational noise considered in this Section does not alter the dynamics of the system. In contrast, variations in the bifurcation parameter can result in changes in the underlying dynamics. Is our framework robust enough to reconstruct the correct dynamics in these scenarios? To address this question, we consider three chaotic food chain systems: the first with $\mathrm{K} = 0.98$ (an example in main text), the second also with $\mathrm{K} = 0.98$ but under noise of a moderate level σ, and the third with $\mathrm{K} = 0.97$. For each system, we apply our transformer reservoir-computing framework to reconstruct the attractor and to compare it with the ground truth. The performance is conveniently characterized by the 3 by 3 confusion matrix, where we compare each predicted attractor with

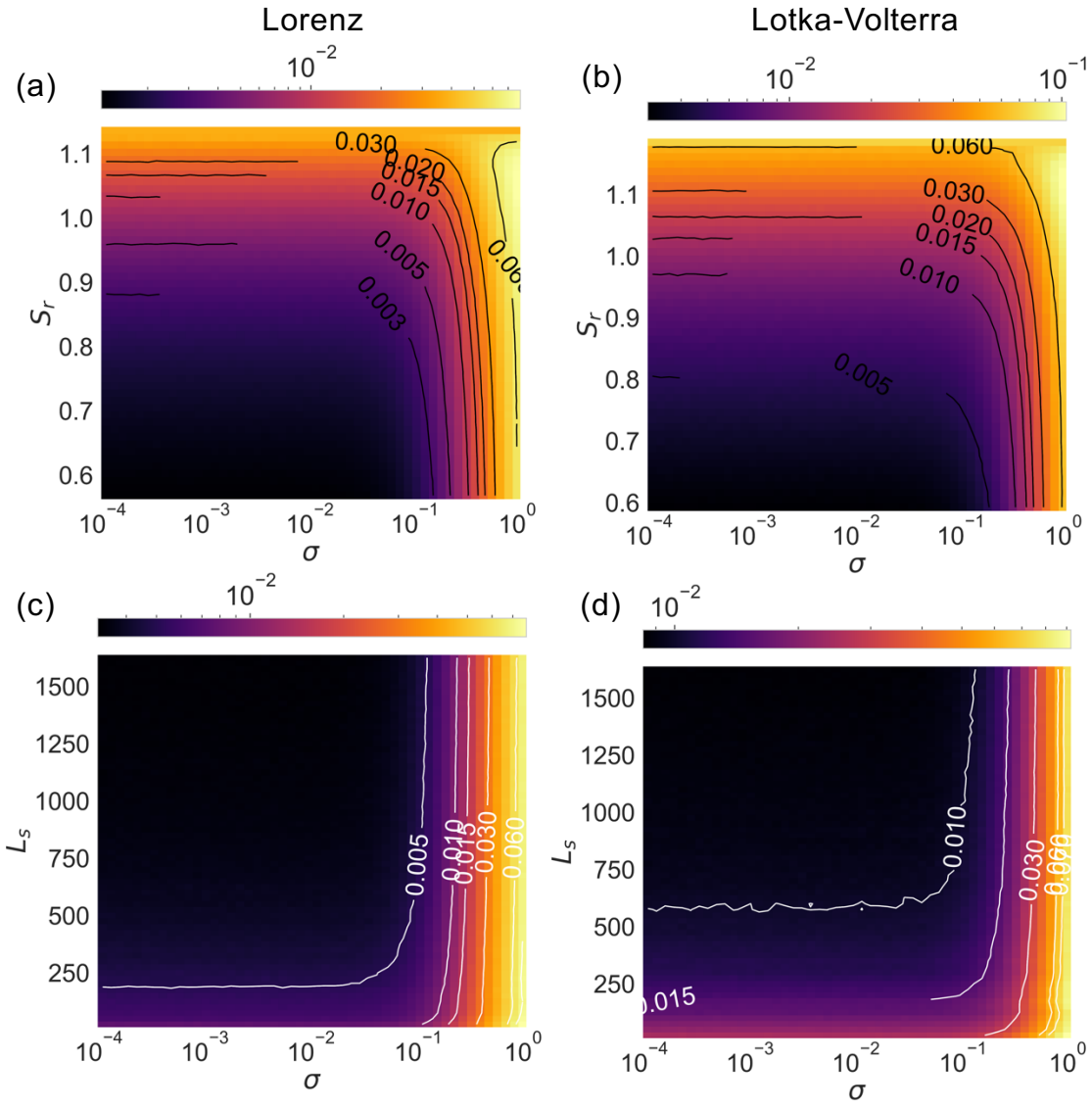

FIG. S14. Robustness against multiplicative noise. Two columns represent results from the Lorenz and Lotka-Volterra systems, respectively. The two rows show ensemble-averaged, color-coded MSE from 50 independent realizations, over the parameter planes (σ, S_r) and (σ, L_s), respectively. The results indicate that the transformer-based dynamics-reconstruction framework is robust against multiplicative noise.

the ground-truth attractor. Figures S17(a) and S17(b) present examples of the reconstructed time series and the corresponding confusion matrix, respectively. The results show that the predicted attractors for $\mathrm{K} = 0.97$ and $\mathrm{K} = 0.98$ align most closely with their corresponding ground-truth attractors, consistent with our previous findings. However, when noise is added to the observations at $\mathrm{K} = 0.98$, the predicted attractor closely matches the ground-truth attractor of the noiseless $\mathrm{K} = 0.98$ system, rather than the attractor at $\mathrm{K} = 0.97$ or the noisy attractor at $\mathrm{K} = 0.98$. The results demonstrate a key distinction between parameter perturbation and observational noise: while noise perturbs the observed data, the framework is robust enough to reconstruct the underlying true dynamics. In contrast, a change in the bifurcation parameter alters the system's governing equations and thereby the dynamics themselves, which our framework correctly identifies as dis-

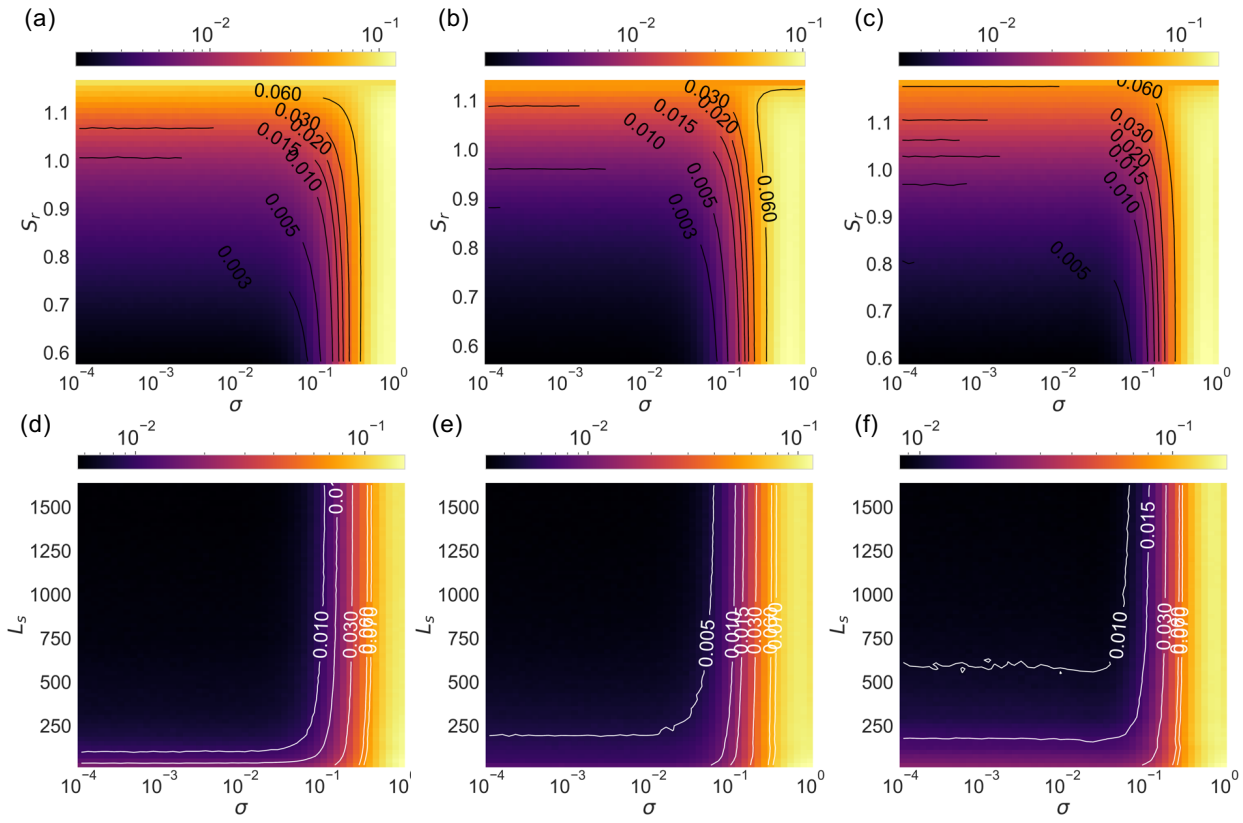

FIG. S15. Robustness against additive noise. The three columns represent results for the food chain, Lorenz, and Lotka-Volterra systems, respectively. The two rows show ensemble-averaged, color-coded MSE from 50 independent realizations, over the parameter planes (σ, S_r) and (σ, L_s), respectively. The transformer-based dynamics reconstruction framework is robust against additive noise.

tinct.

SUPPLEMENTARY NOTE 8: A COUNTER EXAMPLE

For testing, three target systems are presented as examples of reconstructing the system dynamics, assuming that the transformer has never been exposed to these dynamics. This indicates that the transformer trained using data from a large number of synthetic dynamical systems already possesses the ability to find the dynamics behind the sparse data from the target system. To verify that this is indeed the case, we study two scenarios for comparison: (1) the transformer has such a "dynamics-adaptable" ability and (2) it does not have the ability. For the first scenario, we use stochastic signals for testing. Specifically, we generate uniformly distributed noise independently for the three dimensions. A Gaussian filter with the standard deviation $\sigma_g = 9$ of the Gaussian kernel is applied to smooth out the generated noisy data. The resulting stochastic signals are normalized and collected as the testing dataset, as shown in Fig. S18(a). We then take a well-trained transformer adapted for other chaotic systems and test it with the stochastic signal, as shown in Fig. S18(b). For comparison, performance with the food-chain system is shown in Fig. S18(c). For the same sparsity S_r and sequence length L_s, the transformer successfully reconstruct the dynam-

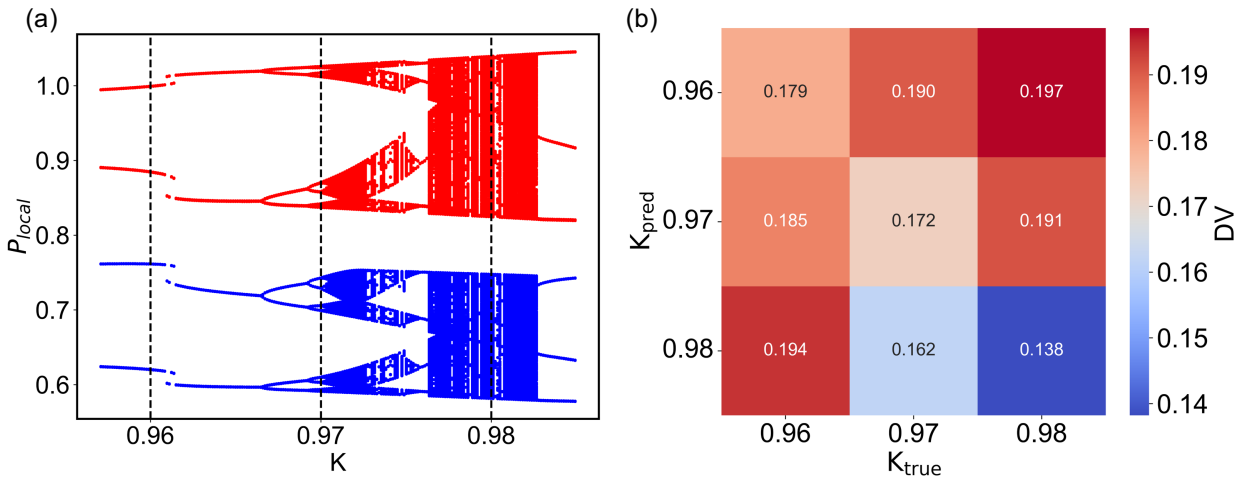

FIG. S16. Robustness against parameter variations. (a) Bifurcation diagram of the chaotic food chain system, with K as the bifurcation parameter. Red and blue dots represent the local maxima and minima, respectively. (b) Confusion matrix of the predicted and ground truth attractors, with DV averaged over 50 independent realizations. The predicted attractor for each K closely matches the corresponding ground truth attractor, while remaining distinct from attractors generated with other parameters.

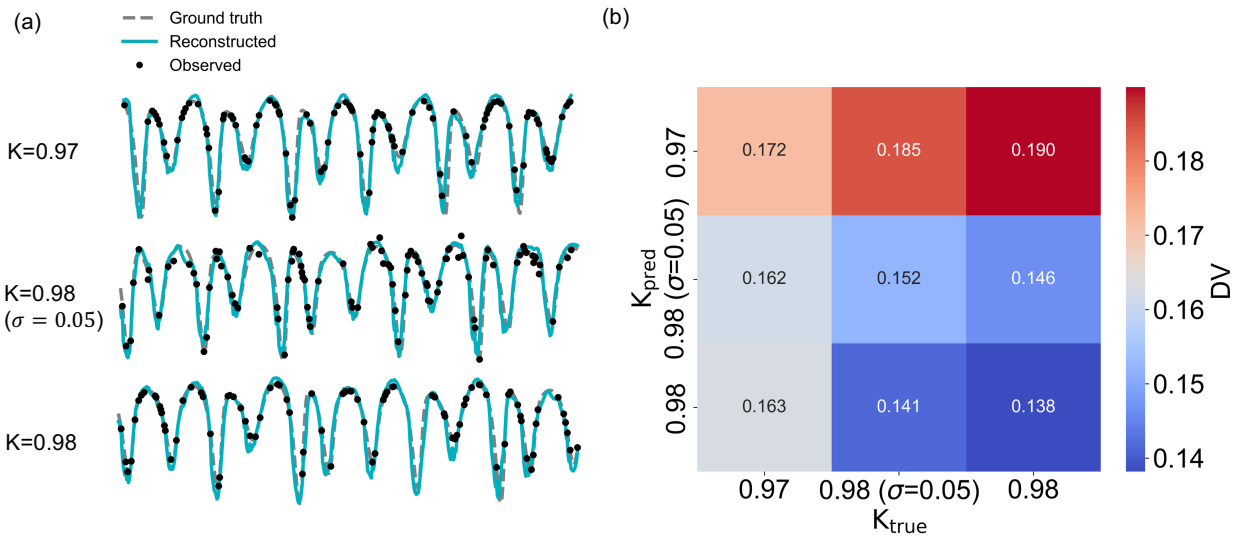

FIG. S17. Distinguishing the dynamics with parameter variations and observational noise. (a) Examples of the dynamics reconstruction of the chaotic food chain system, from top to bottom: $\mathrm{K} = 0.97$, $\mathrm{K} = 0.98$ with multiplicative noise of noise level $\sigma = 0.05$, $\mathrm{K} = 0.98$. (b) Confusion matrix of the predicted and ground-truth attractors, with DV averaged over 50 independent realizations. The predicted noisy attractor for $\mathrm{K} = 0.98$ is closest to the noiseless ground-truth attractor for $\mathrm{K} = 0.98$.

ics of the food chain system but completely fails to reconstruct the stochastic signal. The general conclusion is that, the sparse data presented to the transformer need to be associated with some deterministic, nonlinear dynamical process to achieve successful reconstruction.

We further study the performance of the transformer using the stochastic signals and the dy-

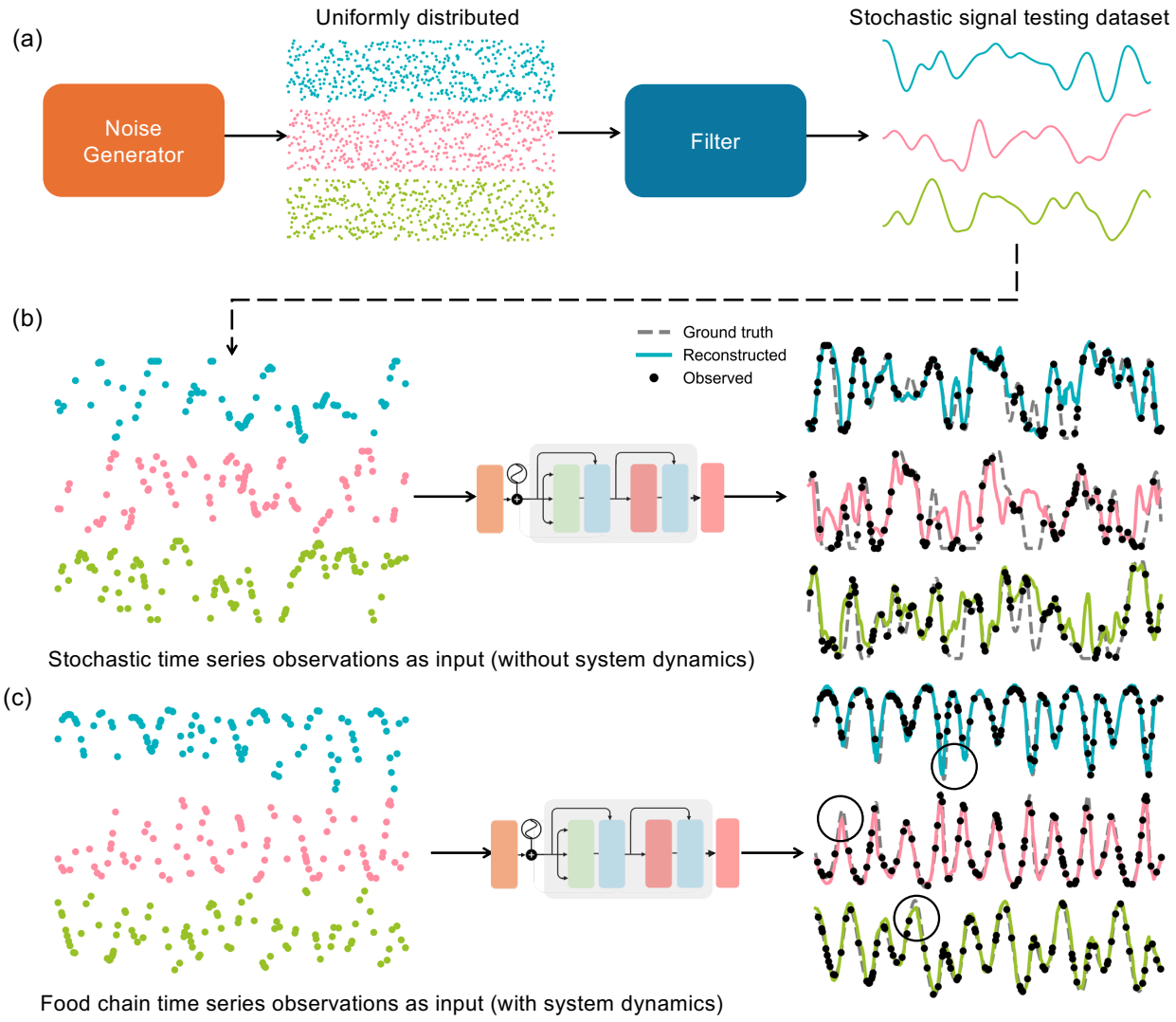

FIG. S18. Illustration of stochastic signal reconstruction. (a) Generation of the stochastic signal dataset. (b,c) Examples of reconstructing time series without and with underlying system dynamics by using a well-trained transformer for $L_s = 1200$ (about 25 cycles of oscillation) and $S_r = 0.93$.

namical systems, as shown Fig. S19. It can be seen that, regardless of the values of S_r and L_s, the performance on the target dynamical systems consistently surpasses that on stochastic signals. A performance comparison for $S_r = 0.58$ is shown in Fig. S19(f). Even with this low sparsity, the transformer is unable to reconstruct the stochastic signal.

SUPPLEMENTARY NOTE 9: NONAUTONOMOUS SYSTEMS

In our framework, both the training systems and the target unseen systems are autonomous, where the transformer is trained using data from a diverse set of dynamical systems and is then evaluated by recovering the underlying dynamics from sparse observations on the target system. When the target system is nonautonomous, training solely on autonomous systems may not be sufficient for reconstructing the target dynamics. To evaluate the capability of our framework in

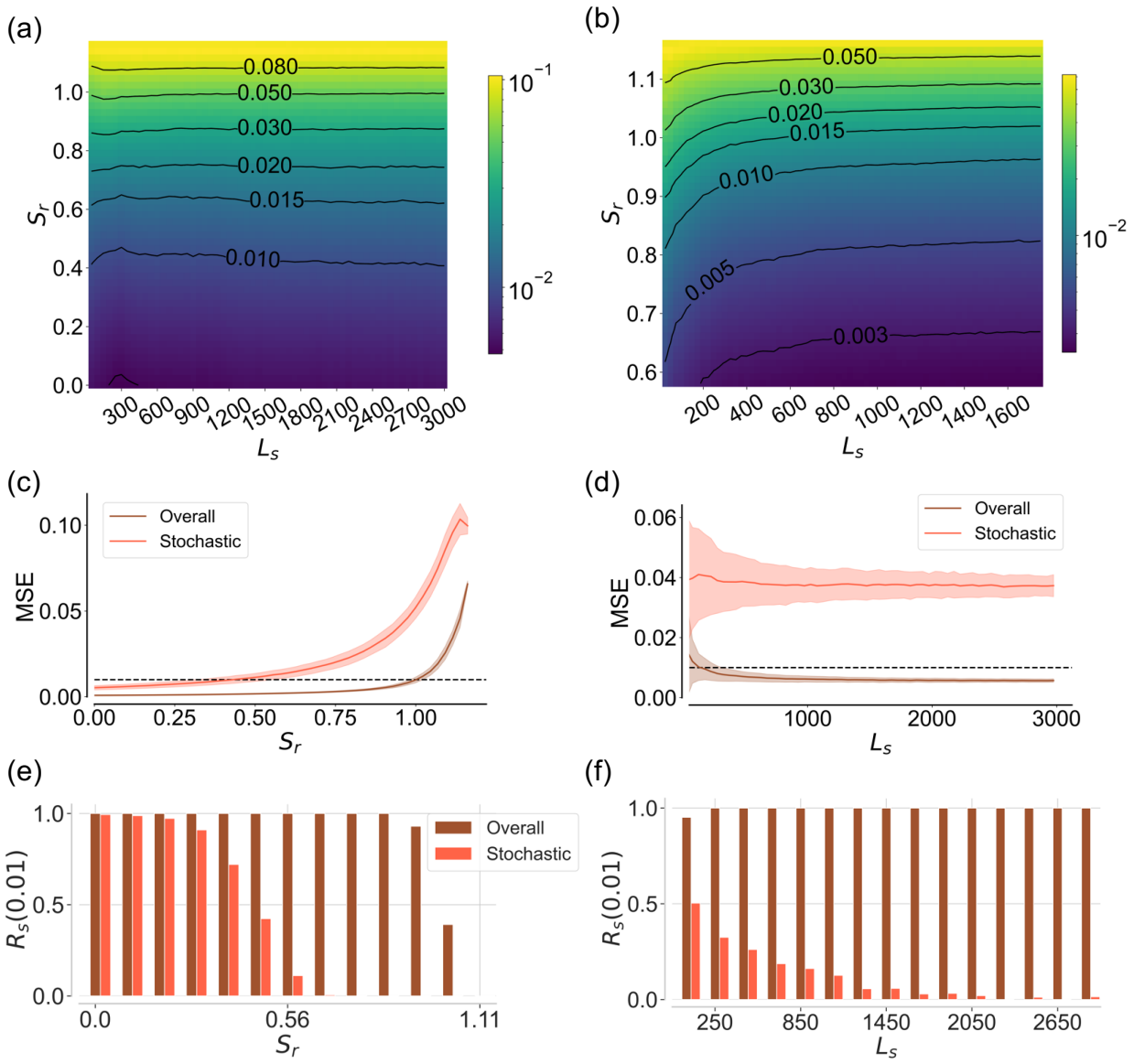

FIG. S19. Performance of transformer on stochastic signals and target systems. (a,b) Ensemble-averaged color-coded MSE in the parameter plane (L_s, S_r) for the stochastic and target dynamical systems, respectively, (c) MSE versus S_r for $L_s = 1200$ (around 25 cycles), (d) MSE versus L_s for $S_r = 0.93$, (e) reconstruction stability $R_s(\cdot)$ versus S_r for $L_s = 1200$, (f) reconstruction stability $R_s(\cdot)$ versus L_s for $S_r = 0.58$. The reconstruction stability and the averaged MSE are calculated from 400,000 data points (corresponding to around 8,000 cycles) in each case.

dealing with nonautonomous systems, we collect four such systems: the forced Rössler, forced Lorenz, mega extreme [26], and memristive laser [27] systems. Concretely, we use the first three systems as part of the training set and treat the laser system as an unseen target. The equations for the four nonautonomous systems are listed in Table S5.

We compare two training strategies: (1) mixed training that includes the three nonautonomous systems and k-3 autonomous systems, for a total of k systems, and (2) purely autonomous training without any nonautonomous systems, using k autonomous systems. Both strategies are trained 50

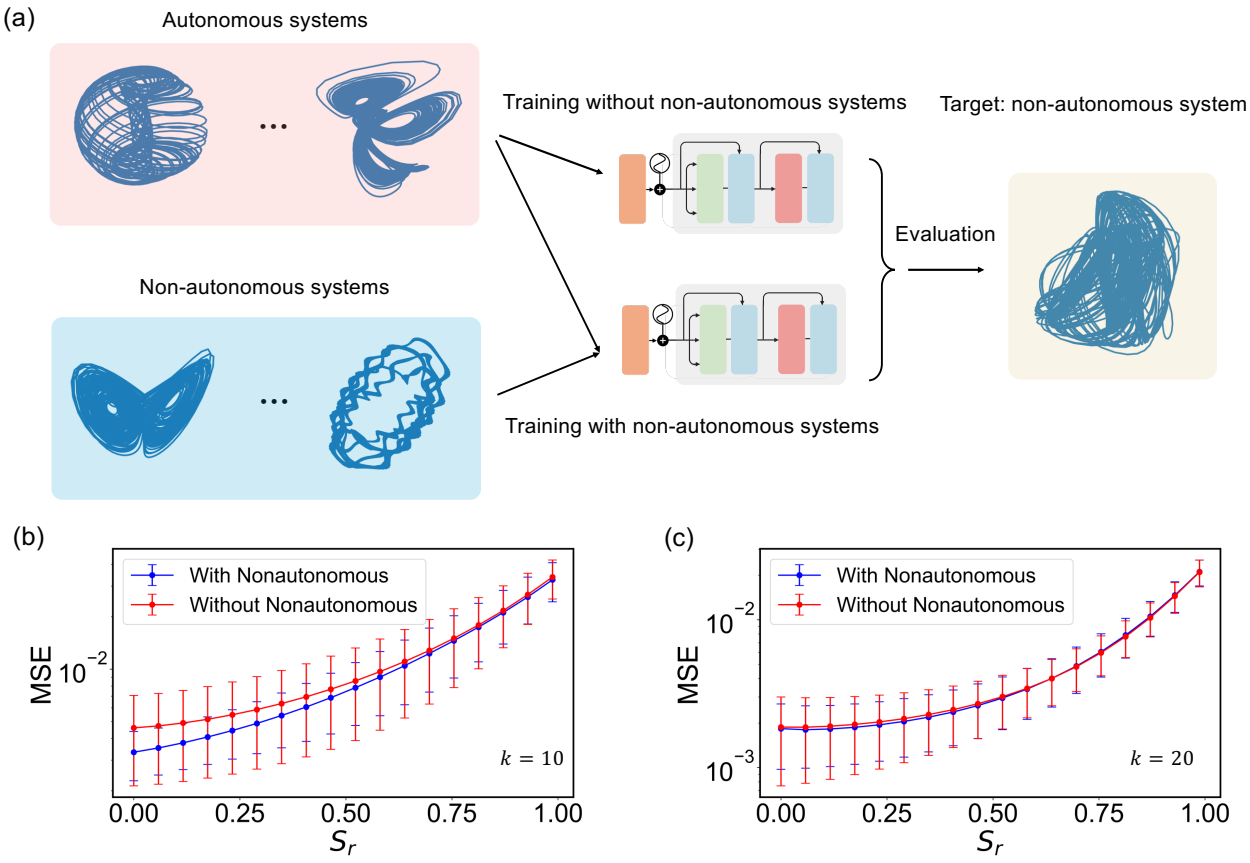

FIG. S20. Comparison of dynamic reconstruction training on autonomous and nonautonomous systems. (a) Illustration of two training strategies: training without and with nonautonomous systems. (b,c) Number of training systems is $k = 10$ and $k = 20$, respectively. In both cases, three nonautonomous systems are provided for "with nonautonomous" strategy. To reduce the statistical fluctuations, 50 independent models are trained to obtain the averaged MSE, and error bars represent standard deviations among them.

times and evaluated on the nonautonomous laser system. The structure of the training with and without the nonautonomous systems and testing on the nonautonomous system is illustrated in Fig. S20(a). As the performance of transformer scales with the number of training systems in a power-law fashion, including a sufficient number of training systems can be crucial: we test this by setting $k = 10$ and $k = 20$ for comparison. Since the total number of systems in our training pool is larger than k, we randomly select the required number of systems from the pool for each training event. As shown in Fig. S20(b), when the number of training systems is small ($k = 10$), the fraction of nonautonomous systems ($3/10$) is relatively high, and the model trained with the mixed strategy outperforms the one trained solely on autonomous systems. However, for $k = 20$ where the ratio of nonautonomous systems is $3/20$, the performance difference between the two strategies becomes negligible.

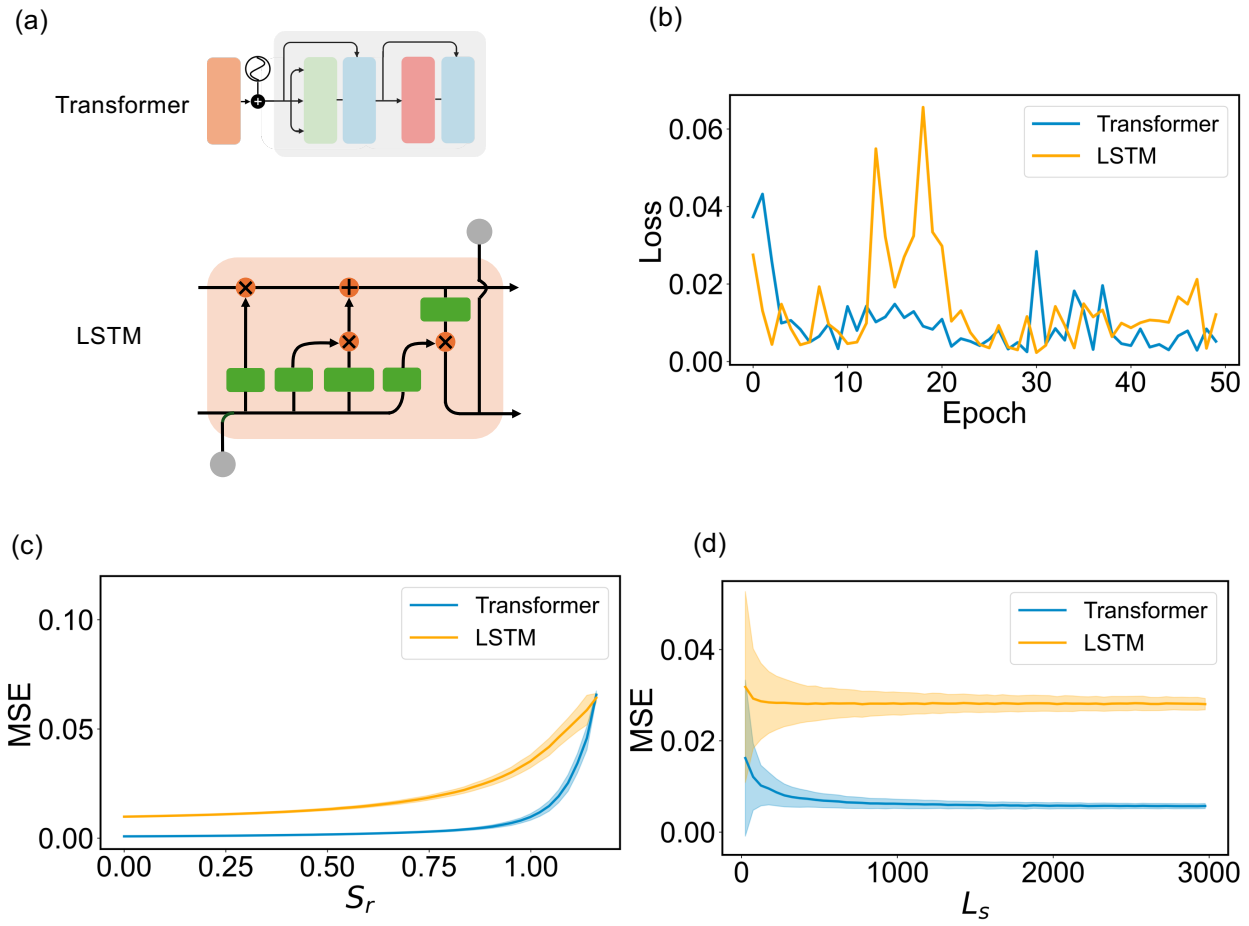

FIG. S21. Comparison of dynamics reconstruction performance between transformer and RNNs. (a) Basic structures of the three machine learning frameworks. (b) Training loss of transformer and LSTM. (c) MSE versus S_r for transformer and LSTM for $L_s = 1200$ (around 25 cycles). (d) MSE versus L_S for $S_r = 0.93$. Averaged MSEs are calculated from 400,000 data points (corresponding to about 8,000 cycles of oscillation) in each case, for the food chain, Lorenz, and Lotka-Volterra systems.

SUPPLEMENTARY NOTE 10: RECONSTRUCTING DYNAMICS WITH TRADITIONAL METHODS

Long short-term memory (LSTM) is a type of recurrent neural networks designed to capture the long-term dependencies in sequential data [28]. LSTM has demonstrated remarkable success in time series forecasting, natural language processing, and other tasks, due to its unique cell structure comprising input, forget, and output gates, which regulate the flow of information through the network. The relevant information is retained over extended time steps in LSTM networks, helping them capture temporal dependencies. The basic structure of LSTM is shown in Fig. S21(a).

We investigate whether LSTM, as representative of RNNs, can be appropriate substitutes for the transformer for dynamics reconstruction from sparse data. To ensure a fair comparison, we keep the hyperparameters as consistent as possible. For example, the hidden network size is set to 512, and the number of layers is set to 4. We train both the LSTMs and transformer using the same setup for each training instance, with randomly chosen sequence lengths L_s and spar-

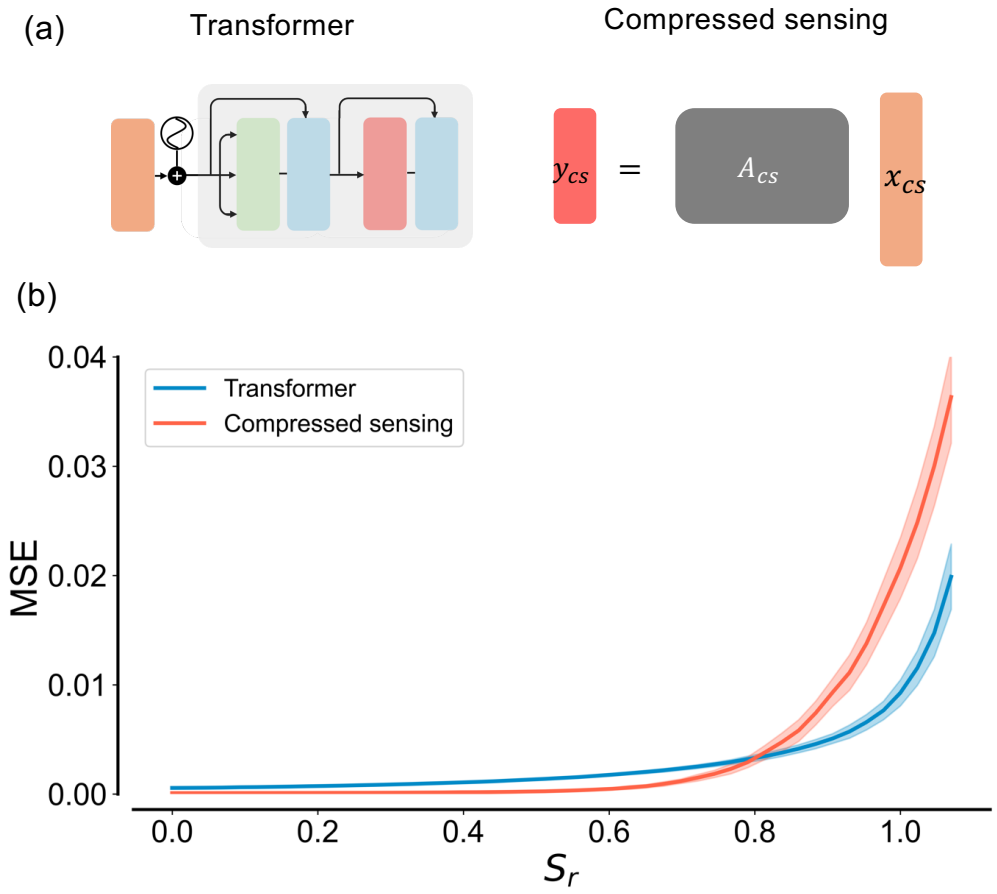

FIG. S22. Comparison of dynamics reconstruction performance by transformer and compressed sensing. (a) Basic structures of the two frameworks. (b) MSE versus S_r for transformer and LSTM for $L_s = 2000$ (around 40 cycles). Averaged MSEs are calculated from 400,000 data points (corresponding to about 8,000 cycles of oscillation) in each case, for the food chain, Lorenz, and Lotka-Volterra systems.

sity S_r, using a batch size of 16 and 50 epochs. The training loss versus the epoch is depicted in Fig. S21(b), indicating that, after the training, the three machine-learning frameworks exhibit similar performance on these training systems. However, for testing, the transformer yields much lower MSE values, as shown in Figs. S21(c) and S21(d). These results suggest that, for dynamics reconstruction from sparse data of unseen dynamical systems, transformer is more effective than traditional recurrent neural networks. We demonstrate that RNNs, such as LSTM, capture dynamics differently from the transformer. The former store information in their hidden states and rely on memory of past inputs to make predictions. As a result, they tend to overfit to the specific systems seen during training, particularly the most recent ones, exhibiting poor generalizability. In contrast, the transformer does not rely on the memory but instead learn to reconstruct dynamics by identifying the correlations among observed points through its attention mechanism. This allows them to capture more fundamental and system-independent properties.

Compressed sensing (CS) is a signal reconstruction framework, which recovers signals from limited observations, provided that the signal is sparse or compressible in a known basis. It is important to note that the definition of "sparse" in CS differs from what is used in our work - it is referred to as the signal having only a few non-zero components when expressed in an appropriate

basis (e.g., Fourier or wavelet). Let y_{cs} be the observed measurements, i.e., the compressed or sampled data. The goal of CS is to recover the full signal x_{cs}, assumed to be sparse in a known basis, by solving the optimization problem:

$$\min ||x_{cs}||_1 \quad \text{subject to} \quad y_{cs} = A_{cs} x_{cs}, \tag{S7}$$

where A_{cs} is a sensing matrix determined by the sampling scheme and basis. In our study, we apply CS in the discrete cosine transform (DCT) domain to reconstruct time series, from randomly sampled observations. The basic framework of CS is illustrated in Fig. S22(a).

We evaluate the performance of CS in reconstructing dynamical systems from sparse and randomly sampled data, as tested for the transformer. Figure S22(b) demonstrates the MSE of CS and transformer reconstruction with varying sparsity S_r. It can be seen that CS can outperform the transformer under low sparsity conditions. However, as the sparsity level increases, the transformer yields more accurate reconstructions, demonstrating its stronger generalizability and recoverability under extreme observational limitations.

SUPPLEMENTARY NOTE 11: RECONSTRUCTING DYNAMICS OF DIVERSE CHAOTIC SYSTEMS

In the main text, we have used 28 chaotic systems for training the transformer and three specific systems for testing. Altogether, 31 chaotic systems have been used. Here we demonstrate that the transformer can also reconstruct arbitrary systems. In particular, each time we choose several systems as the targets for testing, which are excluded from the list of training systems. The remaining chaotic systems are used for training. After training, sparse data from the chosen target system are presented to the transformer for reconstruction. Repeating this process, each and every system in the list of 31 systems can be tested as the target. Representative reconstruction results are shown in Fig. S23, where four systems are chosen as the targets each time. For most systems, the transformer can faithfully reconstruct the target dynamics for $S_r = 0.93$ and $L_s = 1200$. However, for certain target systems, there is reduced performance. For instance, the construction of the third dimension of the chaotic Rössler system is unsatisfactory, in spite of the excellent reconstruction of the first and second dimensions, as shown in Fig. S23(h). The reason is that the third dynamical variable of the Rössler system exhibits a kind of impulsive behavior. Figure S24 shows the performance of dynamics reconstruction of the 28 different chaotic systems in terms of MSE (a) and reconstruction stability $R_s(\cdot)$ (b) for $S_r = 0.93$. The overall MSE is low the reconstruction is highly stable across all the 28 systems, demonstrating the power and generalizability of the transformer based reconstruction framework.

SUPPLEMENTARY NOTE 12: RECONSTRUCTING DYNAMICS OF ADDITIONAL CHAOTIC SYSTEMS

To further demonstrate that the transformer can reconstruct new unseen systems, we evaluate the well-trained transformer (described in the main text) on 28 additional chaotic systems. Specifically, the transformer was previously trained on 28 systems and evaluated on three target systems: the food chain, Lorenz, and Lotka-Volterra systems. Now we introduce 28 new chaotic systems as targets for an extensive assessment. Representative reconstruction results are shown in Fig. S25.

For most systems, the transformer can faithfully reconstruct the target dynamics for $S_r = 0.93$ and $L_s = 2000$. Figure S26 presents the performance of dynamics reconstruction of the 28 additional chaotic systems in terms of MSE (a) and reconstruction stability $R_s(\cdot)$ (b) for $S_r = 0.93$. The overall MSE is low and the reconstruction is highly stable across all 28 systems, demonstrating the power and generalizability of our transformer based reconstruction framework.

TABLE S1. Chaotic systems

Systems	Equations	Parameters
Aizawa	$\dot{x} = (z-b)x - dy$ $\dot{y} = dx + (z-b)y$ $\dot{z} = c + az - z^3/3 - (x^2+y^2)(1+e^z) + fzx^3$	$a = 0.95, b = 0.7, c = 0.6$ $d = 3.5, e = 0.25, f = 0.1$
Bouali	$\dot{x} = x(a-y) + \alpha z$ $\dot{y} = -y(b-x^2)$ $\dot{z} = -x(c-sz) - \beta z$	$\alpha = 0.3, \beta = 0.05$ $a = 4, b = 1, c = 1.5, s = 1$
Chua	$\dot{x} = \alpha(y - x - ht)$ $\dot{y} = \gamma(x - y + z)$ $\dot{z} = -\beta y$	$\alpha = 15.6, \gamma = 1, \beta = 28$ $\mu_0 = -1.143, \mu_1 = -0.714$ $ht = \mu_1 x + 0.5(\mu_0 - \mu_1)(\lvert x+1\rvert - \lvert x-1\rvert)$
Dadras	$\dot{x} = y - ax + byz$ $\dot{y} = cy - xz + z$ $\dot{z} = dxy - ez$	$a = 3, b = 2.7$ $c = 1.7, d = 2, e = 9$
Four wing	$\dot{x} = ax + yz$ $\dot{y} = bx + cy - xz$ $\dot{z} = -z - xy$	$a = 0.2, b = 0.01, c = -0.4$
Hastings-Powell	$\dot{V} = V(1-V) - a_1 VH/(b_1 V + 1)$ $\dot{H} = a_1 VH/(b_1 V + 1) - a_2 HP/(b_2 H + 1) - d_1 H$ $\dot{P} = a_2 HP/(b_2 H + 1) - d_2 P$	$a_1 = 5, a_2 = 0.1$ $b_1 = 3, b_2 = 2$ $d_1 = 0.4, d_2 = 0.01$
Rikitake	$\dot{x} = -\mu x + zy$ $\dot{y} = -\mu y + x(z-a)$ $\dot{z} = 1 - xy$	$\mu = 0.94, a = 1.7$
Rossler	$\dot{x} = -(y+z)$ $\dot{y} = x + ay$ $\dot{z} = b + z(x-c)$	$a = 0.2, b = 0.2, c = 5.7$
Wang	$\dot{x} = x - yz$ $\dot{y} = x - y + xz$ $\dot{z} = -az + xy$	$a = 3$

TABLE S2. Chaotic Sprott systems

Case	Equations
0	$\dot{x} = y,\ \dot{y} = -x + yz,\ \dot{z} = 1 - y^2$
1	$\dot{x} = yz,\ \dot{y} = x - y,\ \dot{z} = 1 - xy$
2	$\dot{x} = yz,\ \dot{y} = x - y,\ \dot{z} = 1 - x^2$
3	$\dot{x} = -y,\ \dot{y} = x + z,\ \dot{z} = xz + 3y^2$
4	$\dot{x} = yz,\ \dot{y} = x^2 - y,\ \dot{z} = 1 - 4x$
5	$\dot{x} = y + z,\ \dot{y} = -x + 0.5y,\ \dot{z} = x^2 - z$
6	$\dot{x} = 0.4x + z,\ \dot{z} = xz - y,\ \dot{y} = -x + y$
7	$\dot{x} = -y + z^2,\ \dot{y} = x + 0.5y,\ \dot{z} = xz$
8	$\dot{x} = -0.2y,\ \dot{y} = x + z,\ \dot{z} = x + y^2 - z$
9	$\dot{x} = 2z,\ \dot{y} = -2y + z,\ \dot{z} = -x + y + y^2$
10	$\dot{x} = xy - z,\ \dot{y} = x - y,\ \dot{z} = x + 0.3z$
11	$\dot{x} = y + 3.9z,\ \dot{y} = 0.9x^2 - y,\ \dot{z} = 1 - x$
12	$\dot{x} = -z,\ \dot{y} = -x^2 - y,\ \dot{z} = 1.7 + 1.7x + y$
13	$\dot{x} = -2y,\ \dot{y} = x + z^2,\ \dot{z} = 1 + y - 2z$
14	$\dot{x} = y,\ \dot{y} = x - z,\ \dot{z} = x + xz + 2.7y$
15	$\dot{x} = 2.7y + z,\ \dot{y} = -x + y^2,\ \dot{z} = x + y$
16	$\dot{x} = -z,\ \dot{y} = x - y,\ \dot{z} = 3.1x + y^2 + 0.5z$
17	$\dot{x} = 0.9 - y,\ \dot{y} = 0.4 + z,\ \dot{z} = xy - z$
18	$\dot{x} = -x - 4y,\ \dot{y} = x + z^2,\ \dot{z} = 1 + x$

TABLE S3. Dominant frequencies and effective bandwidths of chaotic systems

System	f_d (Hz)	$f_{\max}$ (Hz)	Δs
Aizawa	0.59	0.78	0.06
Bouali	0.39	1.17	0.06
Chua	0.68	0.88	0.04
Dadras	0.49	3.13	0.02
Food chain	0.02	0.07	1.00
Four wing	0.10	0.39	0.24
Hastings-Powell	0.008	0.103	1.00
Lorenz	1.30	3.00	0.02
Lotka-Volterra	0.03	0.08	1.00
Rikitake	0.20	0.68	0.10
Rossler	0.20	1.17	0.16
Sprott0	0.10	0.49	0.20
Sprott1	0.11	0.49	0.15
Sprott2	0.20	0.49	0.15
Sprott3	0.20	0.59	0.13
Sprott4	0.20	0.68	0.17
Sprott5	0.10	0.39	0.13
Sprott6	0.20	0.49	0.13
Sprott7	0.19	0.40	0.18
Sprott8	0.09	0.28	0.30
Sprott9	0.21	0.49	0.16
Sprott10	0.10	0.39	0.18
Sprott11	0.29	0.78	0.10
Sprott12	0.19	0.49	0.15
Sprott13	0.21	0.49	0.15
Sprott14	0.20	0.31	0.15
Sprott15	0.19	0.39	0.15
Sprott16	0.28	0.49	0.08
Sprott17	0.29	0.59	0.17
Sprott18	0.31	0.62	0.12
Wang	0.39	1.46	0.05

TABLE S4. Optimal hyperparameter values of reservoir computing for target systems

System	α_r	β_r	γ_r	ρ_r	d_r	σ_r
Food chain	0.36	$10^{-1.25}$	1.16	1.29	0.41	$10^{-4.70}$
Lorenz	0.30	$10^{-5.15}$	1.82	1.30	0.68	$10^{-2.04}$
Lotka-Volterra	0.29	$10^{-6.62}$	0.19	1.72	0.02	$10^{-2.73}$

TABLE S5. Nonautonomous chaotic systems

Systems	Equations	Parameters
Forced Rossler	$\dot{x} = -(y+z) + A\cos(\omega_1 t)$ $\dot{y} = x + ay$ $\dot{z} = b + z(x-c) + A\sin(\omega_2 t)$	$a = 0.2, b = 0.2, c = 5.7$ $A = 0.1, \omega_1 = 1.0, \omega_2 = \sqrt{2}$
Forced Lorenz	$\dot{x} = a(y-x)$ $\dot{y} = cx - xz$ $\dot{z} = xy - bz$	$a = 10 + 2\sin(\omega t)$ $b = 8/3 + 0.5\cos(\omega t)$ $c = 28 + \sin(2\omega t), \omega = 0.5$
Mega Extreme	$\dot{x} = y$ $\dot{y} = z + y\cos(x)$ $\dot{z} = -by + A\omega\cos(\omega t)$	$A = 0.8, b = 0.1, \omega = 0.7$
Memristive Laser	$\dot{x} = -ax + bxy - A\sin(2\pi f t)\lvert z\rvert$ $\dot{y} = -(1 + c + x^2)y + c - 1$ $\dot{z} = A\sin(2\pi f t)$	$a = 55, b = 167, c = 6.9$ $A = 19, f = 5.7$

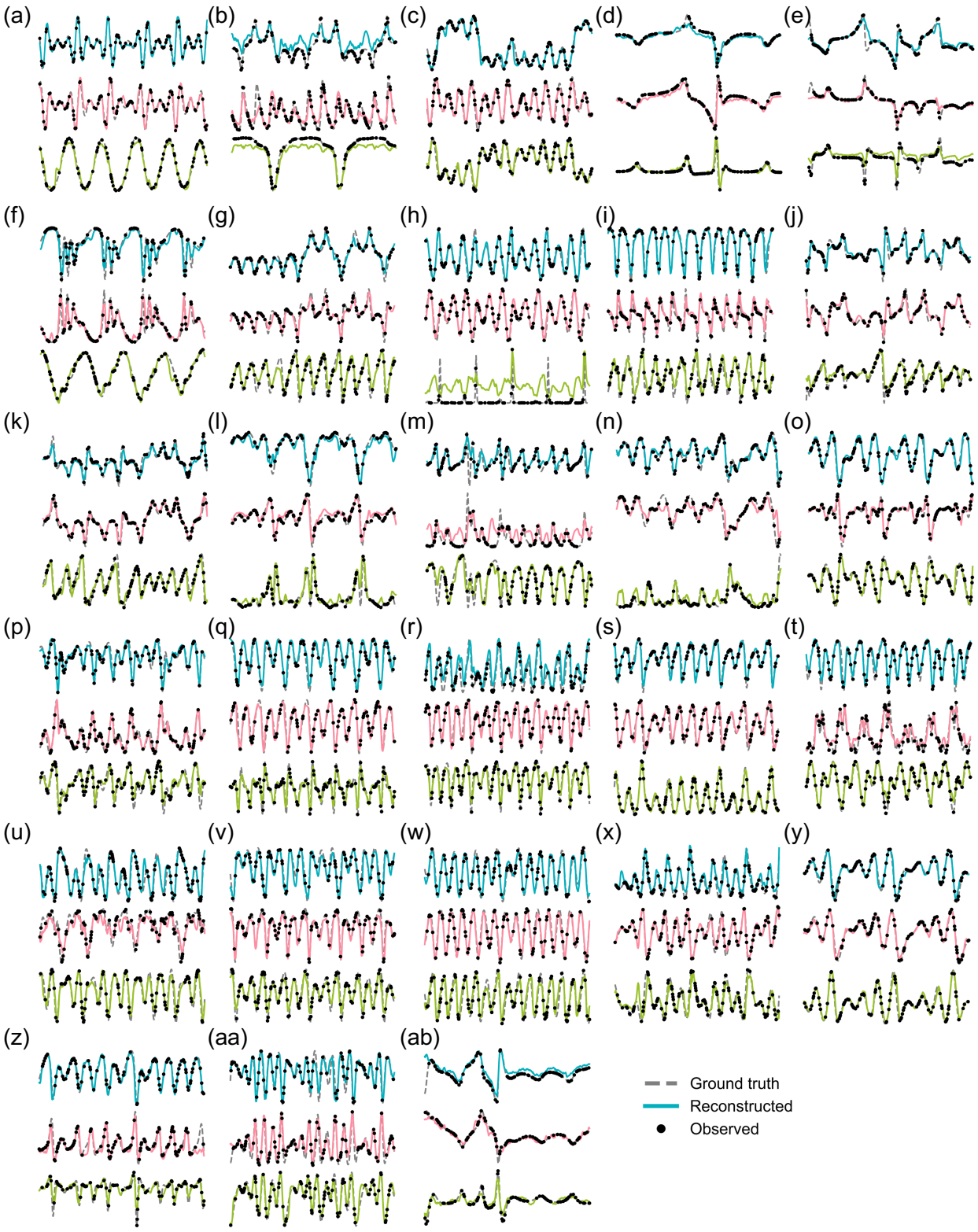

FIG. S23. Dynamics reconstruction of 28 different chaotic systems: (a) Aizawa, (b) Bouali, (c) Chua, (d) Dadras, (e) Four wing, (f) Hastings-Powell, (g) Rikitake, (h) Rossler, (i-aa) Sprott systems. (i-ab) Wang system, for $S_r = 0.93$ and $L_s = 1200$ (about 25 cycles of oscillation).

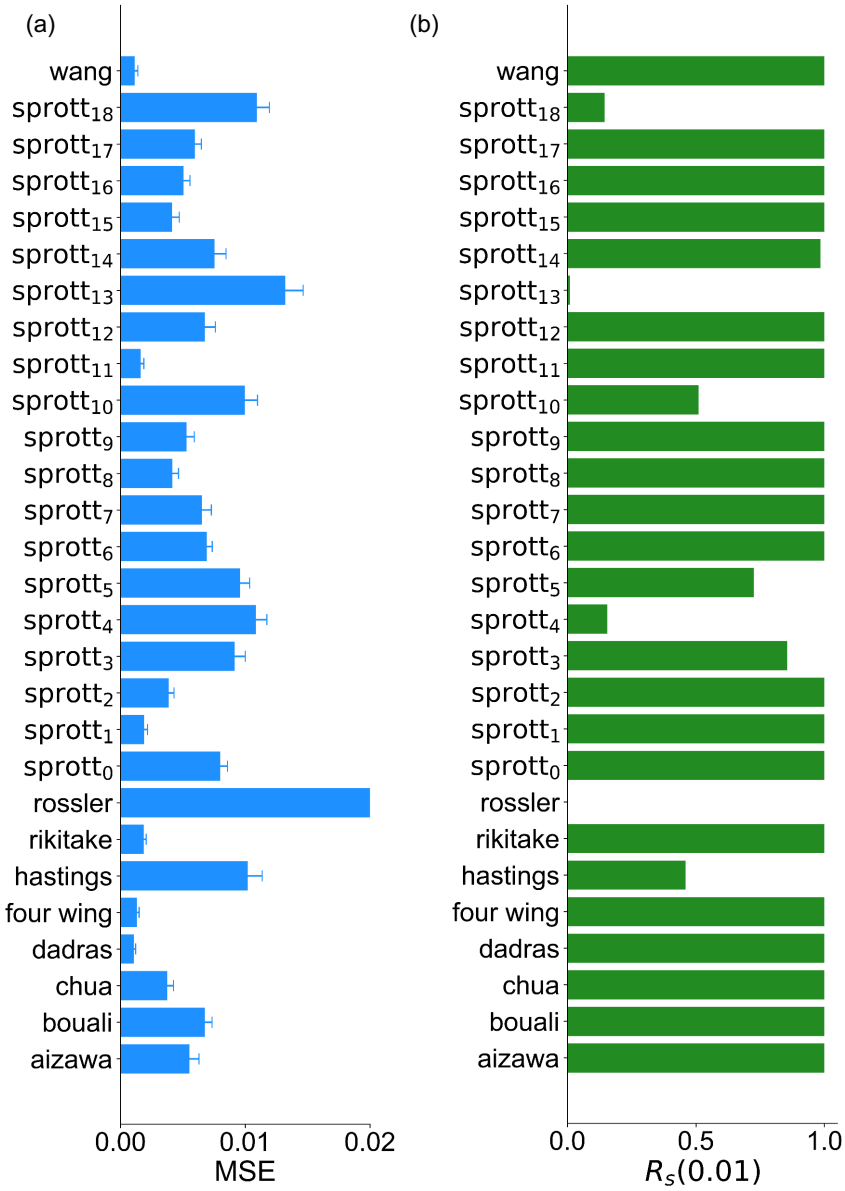

FIG. S24. Performance of dynamics reconstruction on the 28 chaotic systems. (a) Averages MSEs calculated from 400,000 data points (corresponding to about 8,000 cycles of oscillation) in each case, and error bars represent standard deviations. (b) Reconstruction stability $R_s(\cdot)$ for $S_r = 0.93$ and $L_s = 1200$ (about 25 cycles of oscillation).

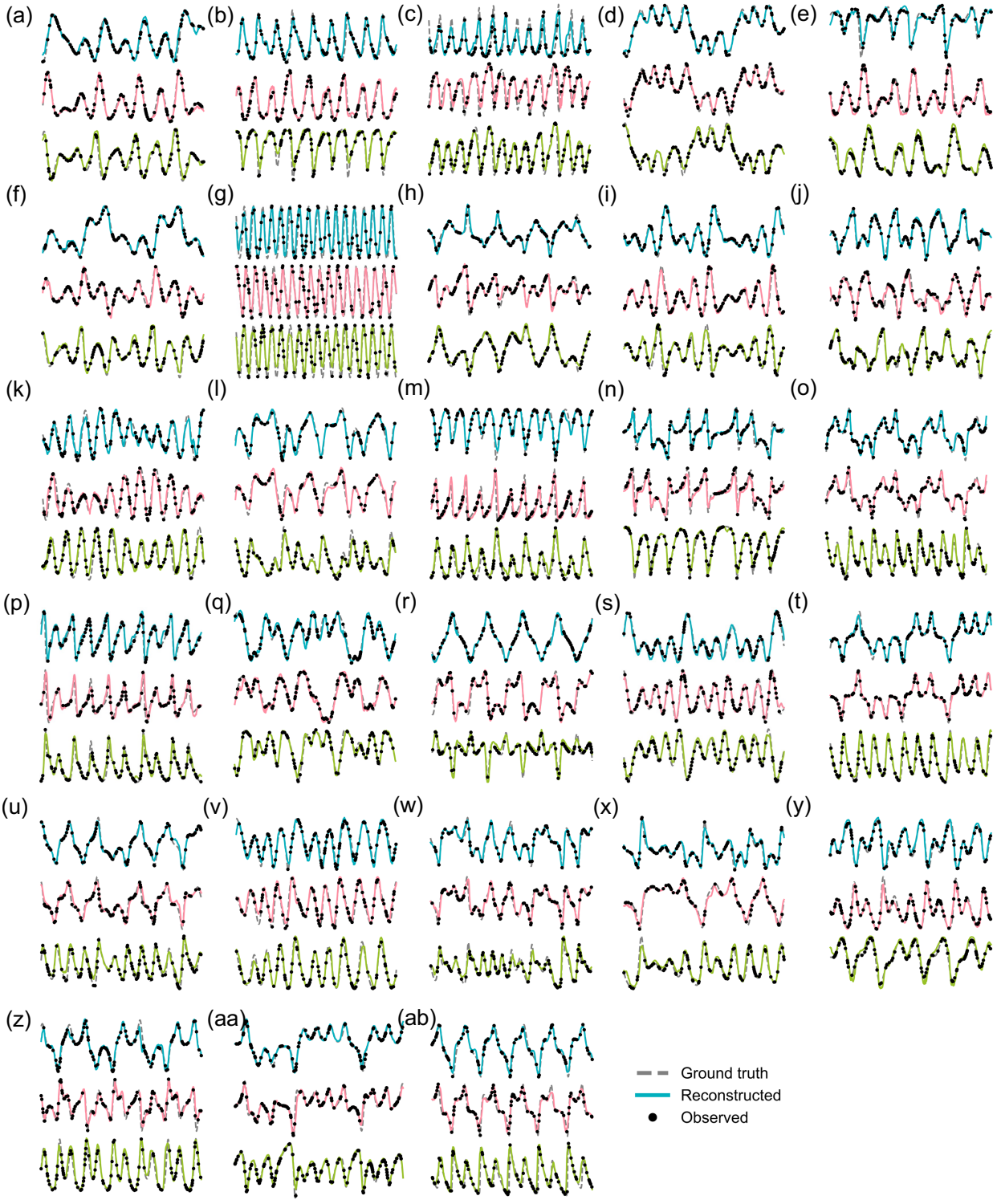

FIG. S25. Dynamics reconstruction of 28 additional unseen chaotic systems: (a) Arneodo, (b) CaTwoPlus, (c) CaTwoPlusQuasiperiodic, (d) CellularNeuralNetwork, (e) Colpitts, (f) Coullet, (g) DoubleGyre, (h) Finance, (i) GenesioTesi, (j) Halvorsen, (k) HenonHeiles, (l) HyperBao, (m) IsothermalChemical, (n) Laser, (o) LuChen, (p) LuChenCheng, (q) MacArthur, (r) MooreSpiegel, (s) MultiChua, (t) PanXuZhou, (u) QiChen, (v) RabinovichFabrikant, (w) RayleighBenard, (x) Rucklidge, (y) SaltonSea, (z) ShimizuMorioka, (aa) VallisElNino, (ab) YuWang, for $S_r = 0.93$ and $L_s = 20000$ (about 40 cycles of oscillation).

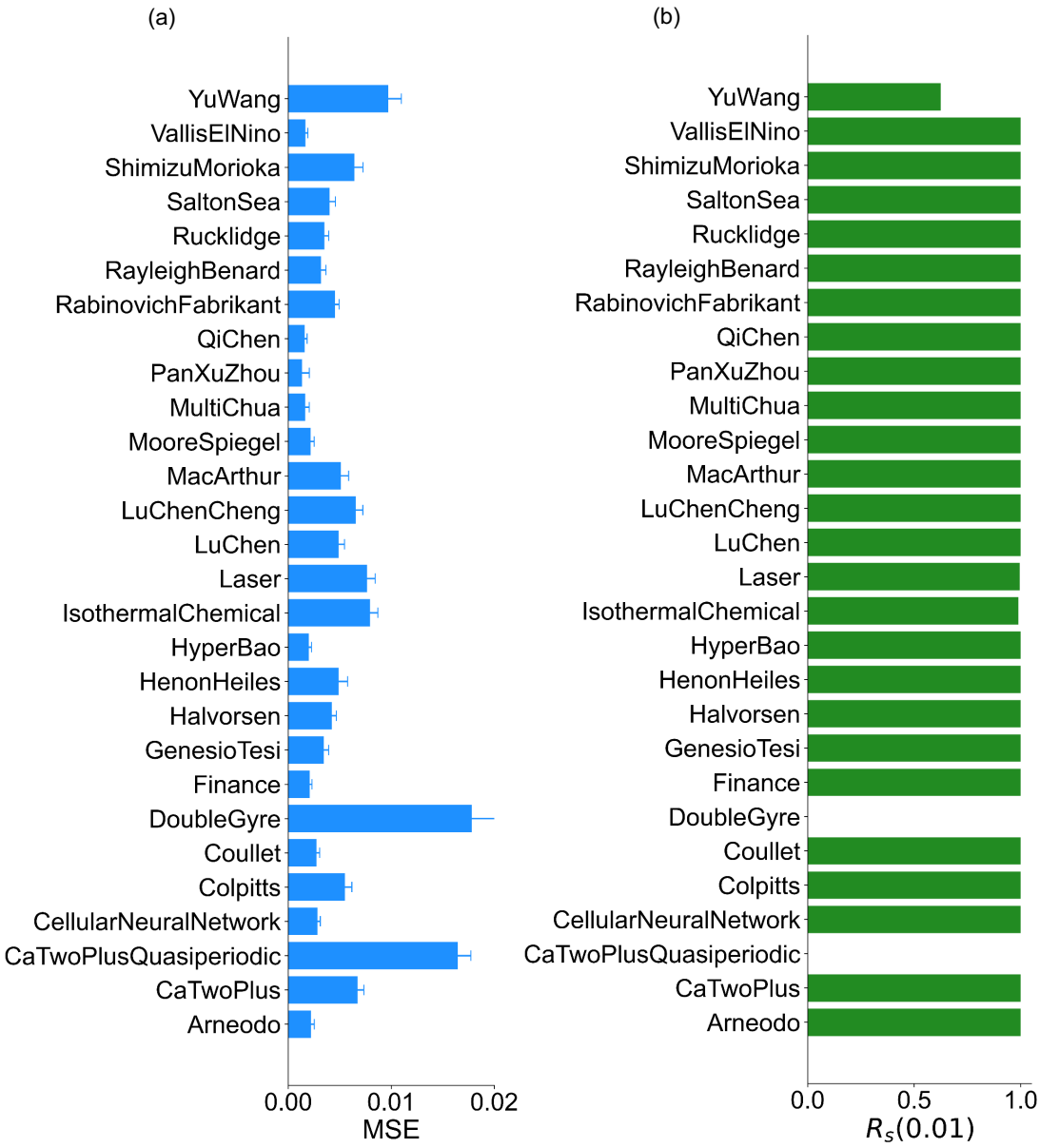

FIG. S26. Performance of dynamics reconstruction on the 28 additional unseen chaotic systems. (a) Averages MSEs calculated from 400,000 data points (corresponding to about 8,000 cycles of oscillation) in each case, and error bars represent standard deviations. (b) Reconstruction stability $R_s(\cdot)$ for $S_r = 0.93$ and $L_s = 2000$ (about 40 cycles of oscillation).

[1] Yu, R. & Wang, R. Learning dynamical systems from data: An introduction to physics-guided deep learning. *Proc. Natl. Acad. Sci. U.S.A.* **121**, e2311808121 (2024).

[2] Batista, G. E., Monard, M. C. *et al.* A study of K-nearest neighbour as an imputation method. *His* **87**, 48 (2002).

[3] Pelckmans, K., De Brabanter, J., Suykens, J. A. & De Moor, B. Handling missing values in support vector machine classifiers. *Neural Netw.* **18**, 684–692 (2005).

[4] Stekhoven, D. J. & Bühlmann, P. MissForest–non-parametric missing value imputation for mixed-type data. *Bioinformatics* **28**, 112–118 (2012).

[5] Du, W., Côté, D. & Liu, Y. Saits: Self-attention-based imputation for time series. *Expert Syst. Appl.* **219**, 119619 (2023).

[6] Che, Z., Purushotham, S., Cho, K., Sontag, D. & Liu, Y. Recurrent neural networks for multivariate time series with missing values. *Sci. Rep.* **8**, 6085 (2018).

[7] Yoon, J., Jordon, J. & Schaar, M. GAIN: Missing data imputation using generative adversarial nets. In *Int. Conf. Mach. Learn.*, 5689–5698 (PMLR, 2018).

[8] Ramchandran, S., Tikhonov, G., Kujanpää, K., Koskinen, M. & Lähdesmäki, H. Longitudinal variational autoencoder. In *Int. Conf. Artif. Intell. Stat.*, 3898–3906 (PMLR, 2021).

[9] Wang, X. *et al.* An observed value consistent diffusion model for imputing missing values in multivariate time series. In *Proc. 29th ACM SIGKDD Conf. Knowl. Discov. Data Min.*, 2409–2418 (2023).

[10] Salimans, T. *et al.* Improved techniques for training gans. *Adv. Neural Inf. Process. Syst.* **29** (2016).

[11] Vaswani, A. *et al.* Attention is all you need. *Adv. Neural Inf. Process. Syst.* **30** (2017).

[12] Liu, J. *et al.* One model to synthesize them all: Multi-contrast multi-scale transformer for missing data imputation. *IEEE Trans. Med. Imaging.* **42**, 2577–2591 (2023).

[13] Dosovitskiy, A. *et al.* An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).

[14] Emmanuel, T. *et al.* A survey on missing data in machine learning. *J. Big Data.* **8**, 1–37 (2021).

[15] Strike, K., El Emam, K. & Madhavji, N. Software cost estimation with incomplete data. *IEEE Trans. Softw. Eng.* **27**, 890–908 (2001).

[16] Acuna, E. & Rodriguez, C. The treatment of missing values and its effect on classifier accuracy. In *Classif. Clust. Data Min. Appl.*, 639–647 (Springer, 2004).

[17] Lin, W.-C. & Tsai, C.-F. Missing value imputation: A review and analysis of the literature (2006–2017). *Artif. Intell. Rev.* **53**, 1487–1509 (2020).

[18] Song, Q. & Shepperd, M. Missing data imputation techniques. *Int. J. Bus. Intell. Data Min.* **2**, 261–291 (2007).

[19] Zhou, Y., Aryal, S. & Bouadjenek, M. R. Review for handling missing data with special missing mechanism. *arXiv preprint arXiv:2404.04905* (2024).

[20] 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).

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

[22] Welch, P. The use of fast fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms. *IEEE Trans. Audio Electroacoust.* **15**, 70–73

(2003).

[23] Vlachas, P.-R. *et al.* Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics. *Neural Netw.* **126**, 191–217 (2020).

[24] Bishop, C. M. Training with noise is equivalent to Tikhonov regularization. *Neural Comput.* **7**, 108–116 (1995).

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

[26] Ahmadi, A. *et al.* A non-autonomous mega-extreme multistable chaotic system. *Chaos, Solitons & Fractals* **174**, 113765 (2023).

[27] Wang, J., Mou, J., Xiong, L., Zhang, Y. & Cao, Y. Fractional-order design of a novel non-autonomous laser chaotic system with compound nonlinearity and its circuit realization. *Chaos, Solitons & Fractals* **152**, 111324 (2021).

[28] Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neu. Comp.* **9**, 1735–1780 (1997).