

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

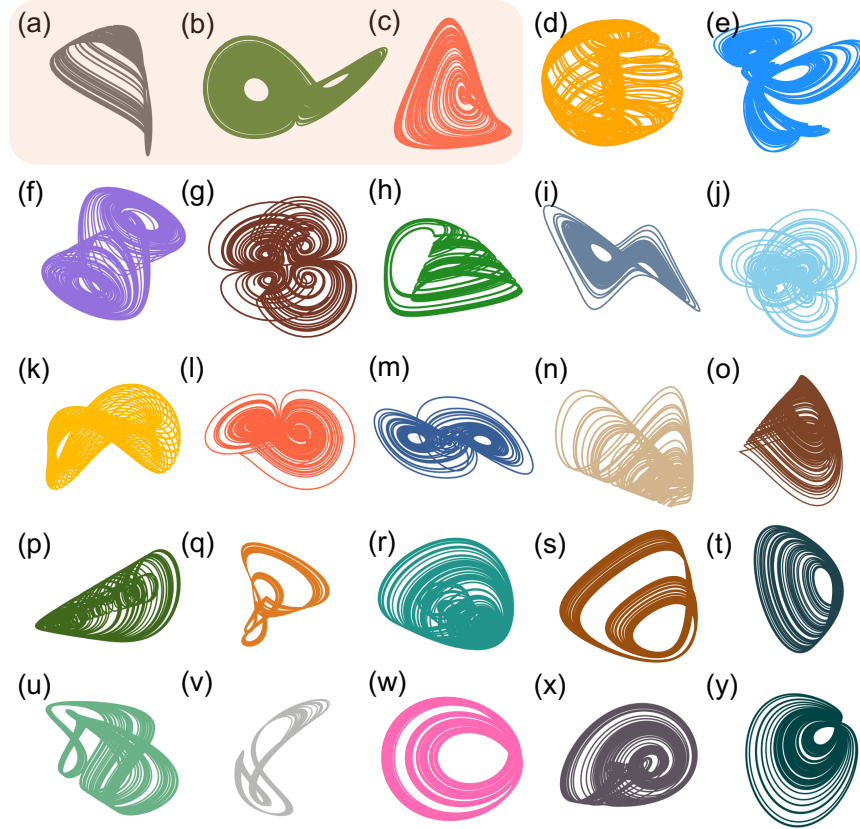


FIG. S1. The chaotic systems for adaptation learning during the training and the target systems for testing. (a-c) Target testing systems: food-chain, Lorenz, and Lotka-Volterra systems, respectively. Systems for learning: (d) Aizawa, (e) Bouali, (f) Chua, (g) Four wing, (h) Hastings-Powell, (i) Rikitake, (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 ($i = 1, \dots, N_c$) competing for finite resources, the equations are given by

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

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

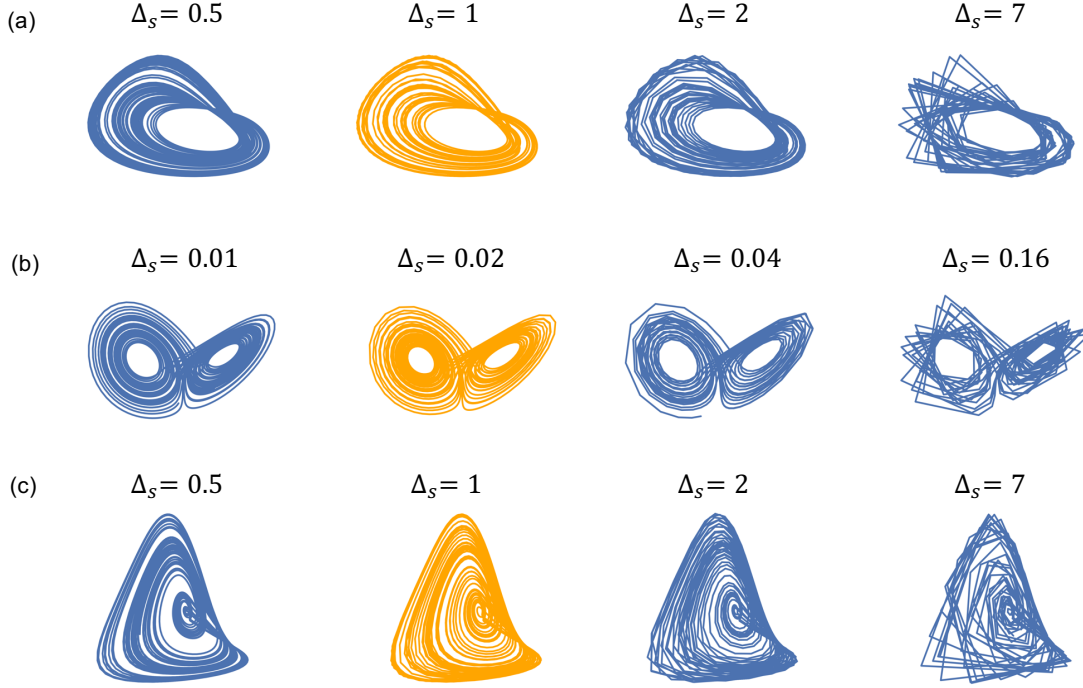


FIG. S2. Simulated attractors of several target systems with varying sampling rate Δs . (a-c) Target testing systems: chaotic food-chain, Lorenz, and Lotka-Volterra systems, respectively. The orange attractors are those reconstructed (ground-truth) using the sampling rate stated in the main text, which 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 Nyquist 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:

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

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.

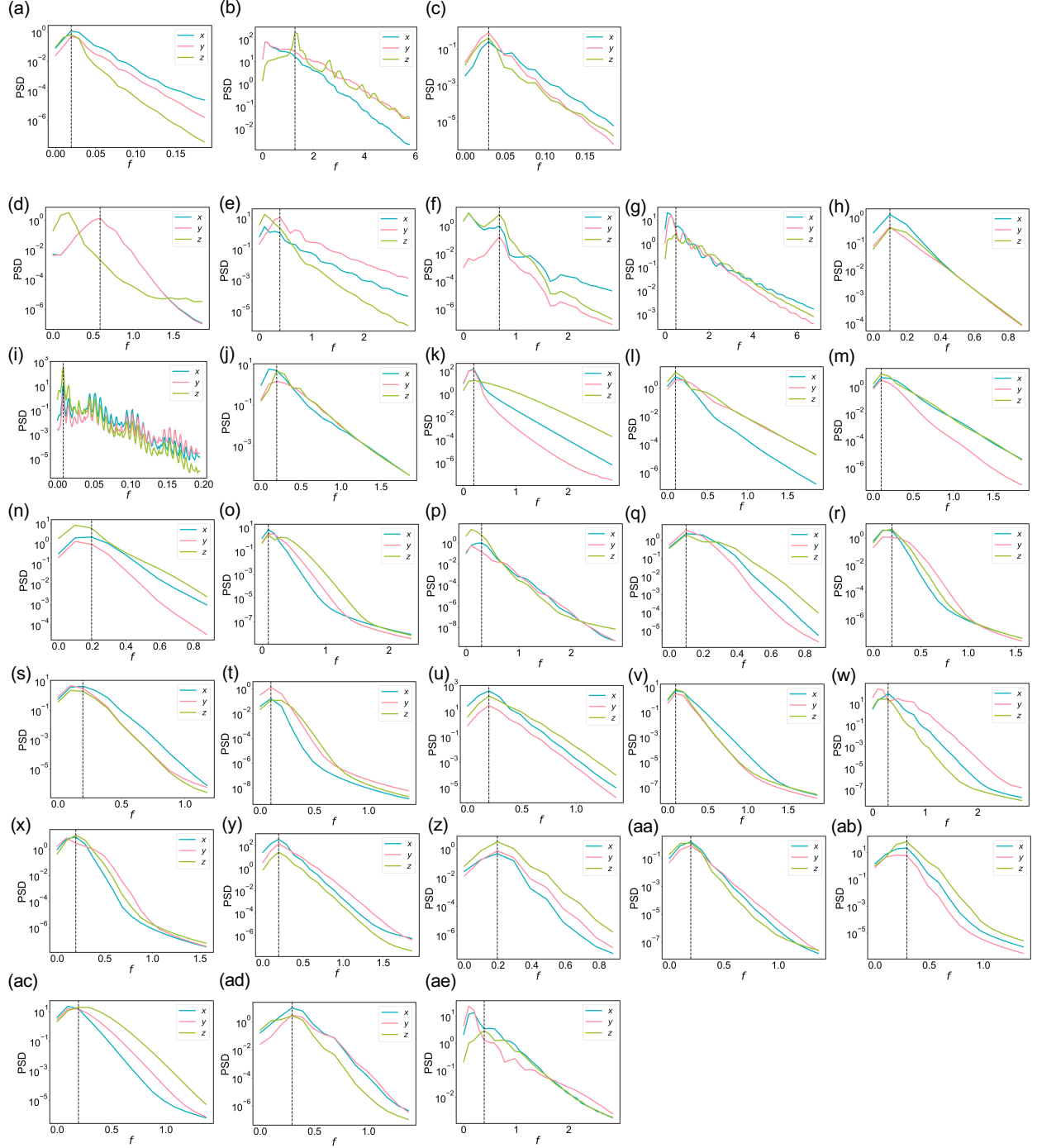


FIG. S3. Power spectral density for the training and the target systems for testing. (a-c) Target testing systems: food-chain, Lorenz, and Lota-Volterra systems, respectively (d) Aizawa, (e) Bouali, (f) Chua, (g) Four wing, (h) Hastings-Powell, (i) Rikitake, (j) Rossler, (k-ad) Sprott, (ae) Wang systems.

SUPPLEMENTARY NOTE 4: RESERVOIR COMPUTING

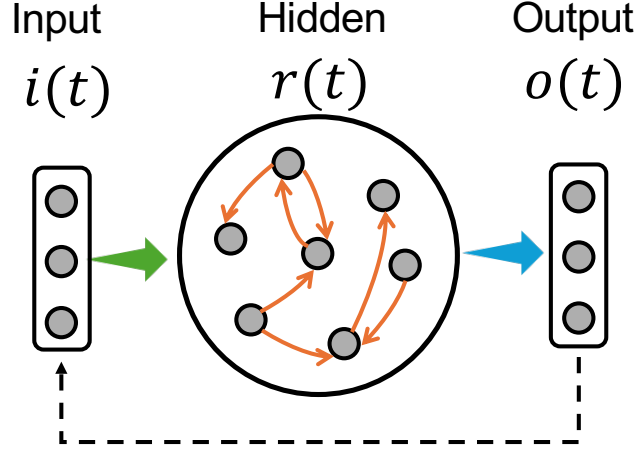


FIG. S4. Basic configuration of reservoir computing.

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 $[\mathbf{i}(:, 1), \mathbf{i}(:, 2), \dots, \mathbf{i}(:, t)]$, the hidden layer state $\mathbf{r}(t)$ is updated step-by-step according to

$$\mathbf{r}(t+1) = (1 - \alpha_r) \cdot \mathbf{r}(t) + \alpha_r \cdot \tanh[\mathcal{A} \cdot \mathbf{r}(t) + \mathcal{W}_{\text{in}} \cdot \mathbf{i}(t)], \quad (\text{S4})$$

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 $N_s \times D_i$, are generated uniformly in the range $[-\gamma_r, \gamma_r]$ prior to training. The elements of the hidden network $\mathcal{A}$ of the dimension $N_s \times N_s$, 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 $D_o \times N_s$. 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 $N_s \times T_l$, 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

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

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:

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

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.

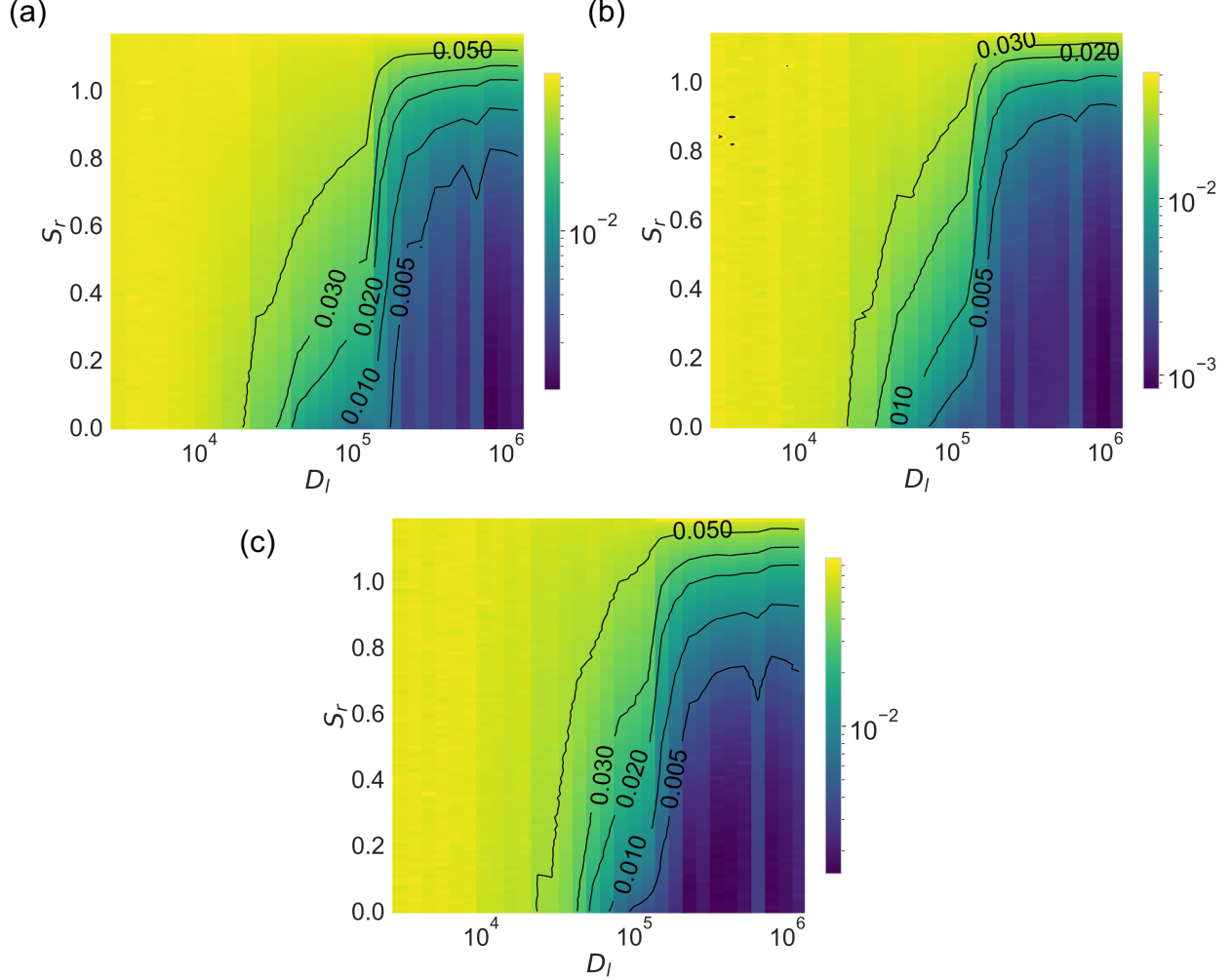


FIG. S5. Effects of the training length D_l on dynamics reconstruction performance, for different sparsity S_r . (a-c) Color-coded ensemble-averaged MSE values in the plane of (D_l, S_r) for the food-chain, Lorenz, and Lotka-Volterra systems, respectively. Each value of the averaged MSE is obtained using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

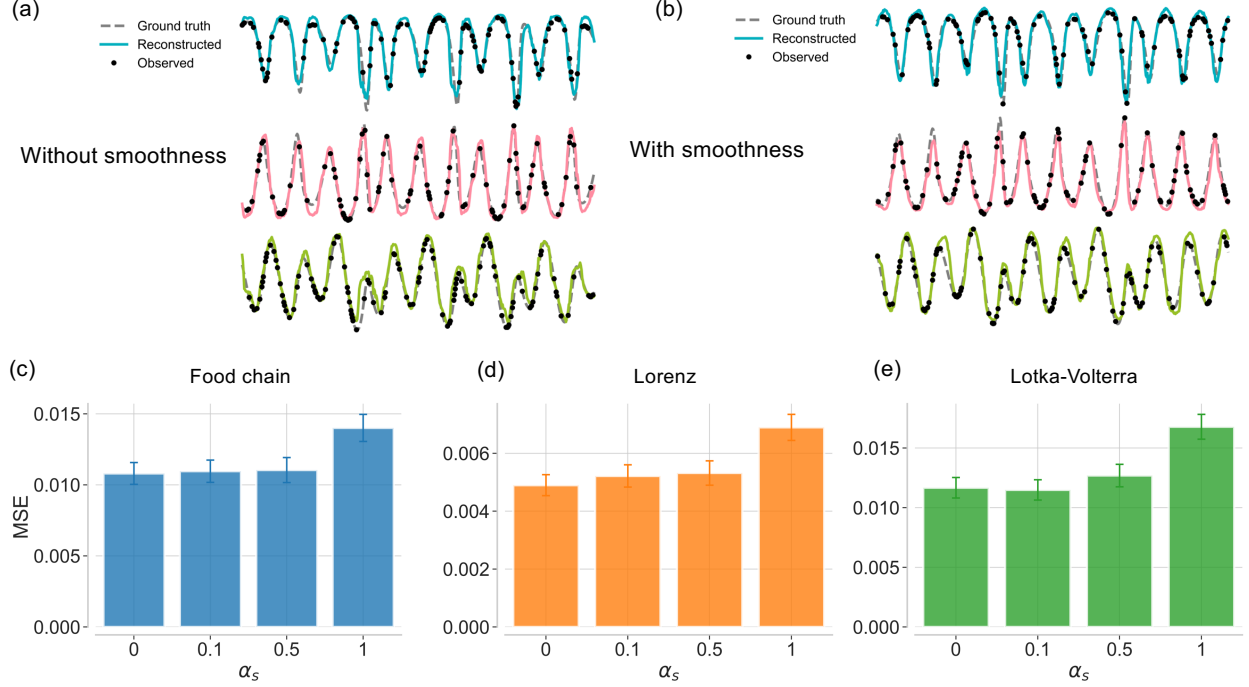


FIG. S7. Effects of the smoothness penalty α_s on dynamics reconstruction performance. (a) An example of training without smoothness and (b) an example of training with smoothness. (c-e) Performance for the food chain, Lorenz, and Lotka-Volterra systems, respectively, with respect to the smoothness penalty 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.

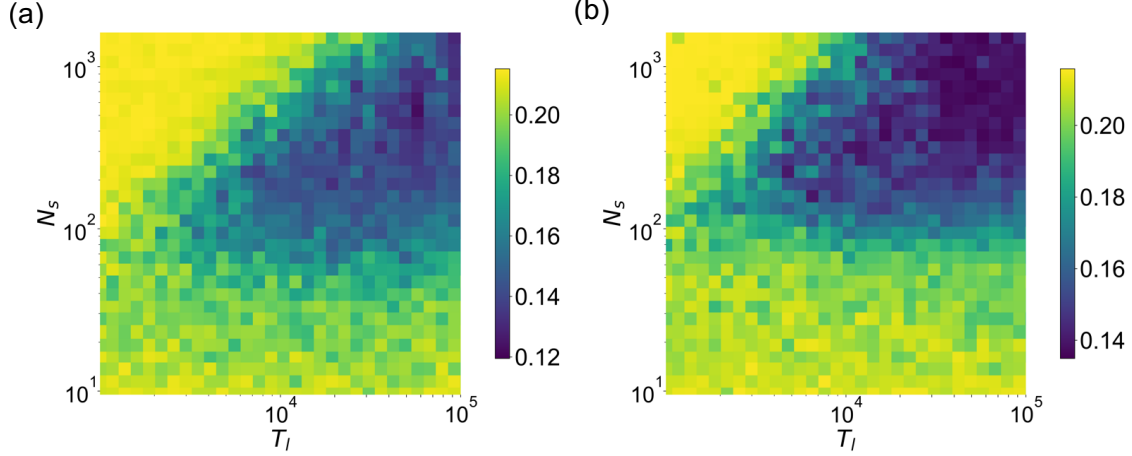


FIG. S8. Effects of reservoir-computing training length and network size on performance. (a,b) Ensemble-averaged MSE obtained from 50 realizations versus changes in the training length T_l and network size N_s for the chaotic Lorenz and Lotka-Volterra systems, respectively. Appropriate parameter ranges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

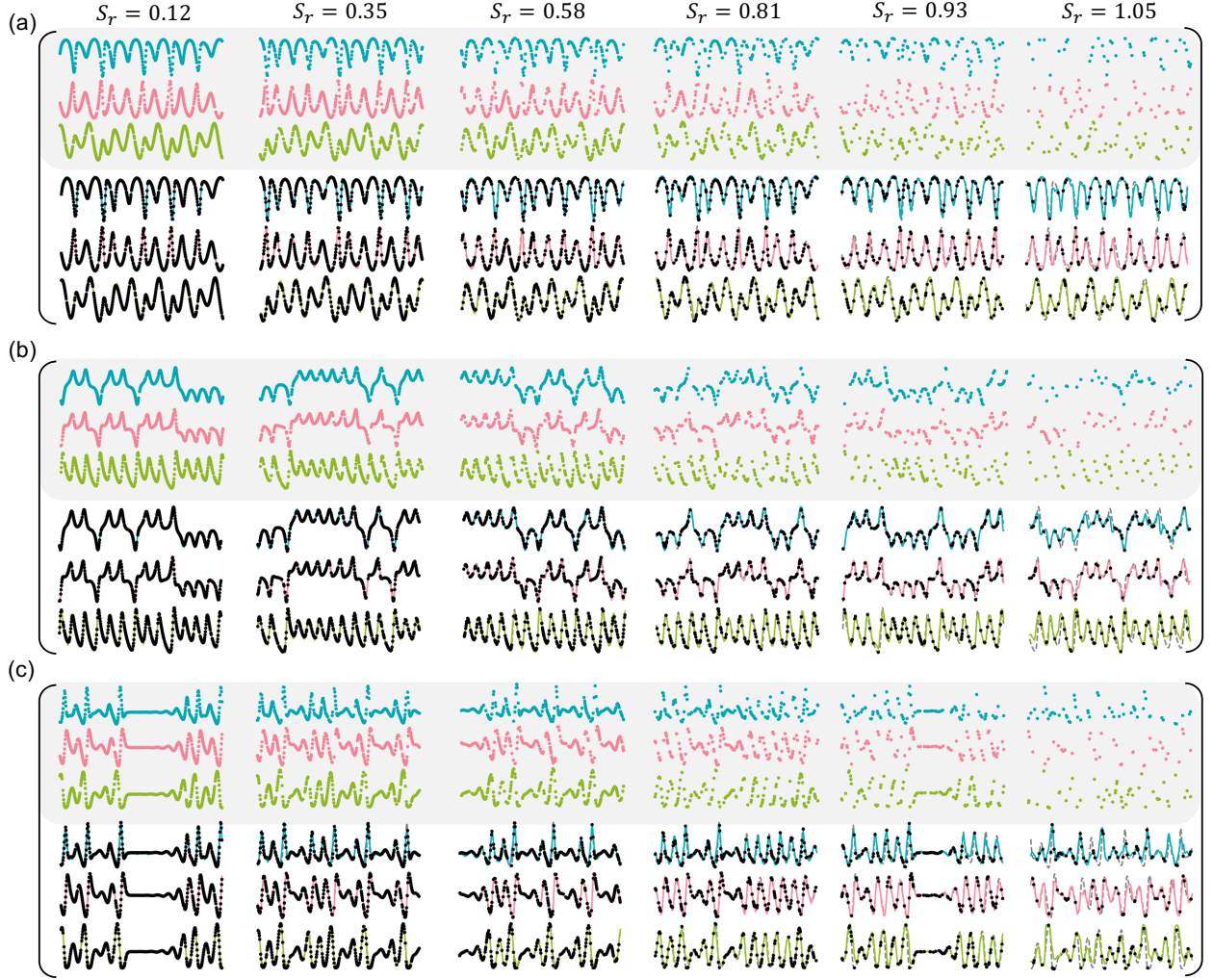


FIG. S9. Additional examples of dynamics reconstruction by transformer for different levels of data sparsity. (a-c) Reconstructed time series for the food chain, Lorenz, and Lotka-Volterra systems, respectively, where the shaded areas represent observable points and the unshaded areas show the corresponding 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

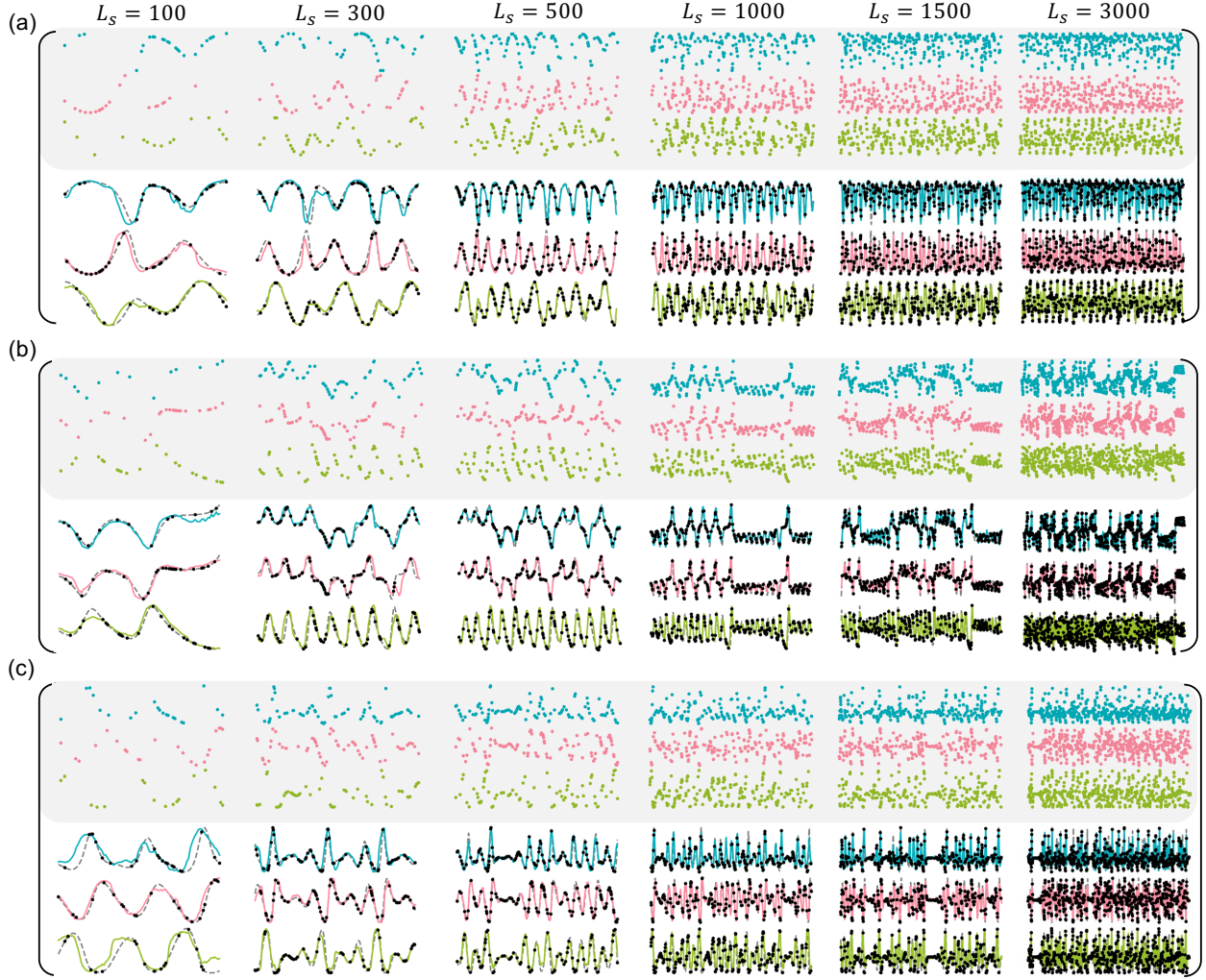


FIG. S10. Additional examples of dynamics reconstruction with respect to varying input sequence length L_s . (a-c) Reconstructed time series for the food chain, Lorenz, and Lotka-Volterra systems, respectively, where the shaded areas represent the observable points and the unshaded areas show the corresponding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

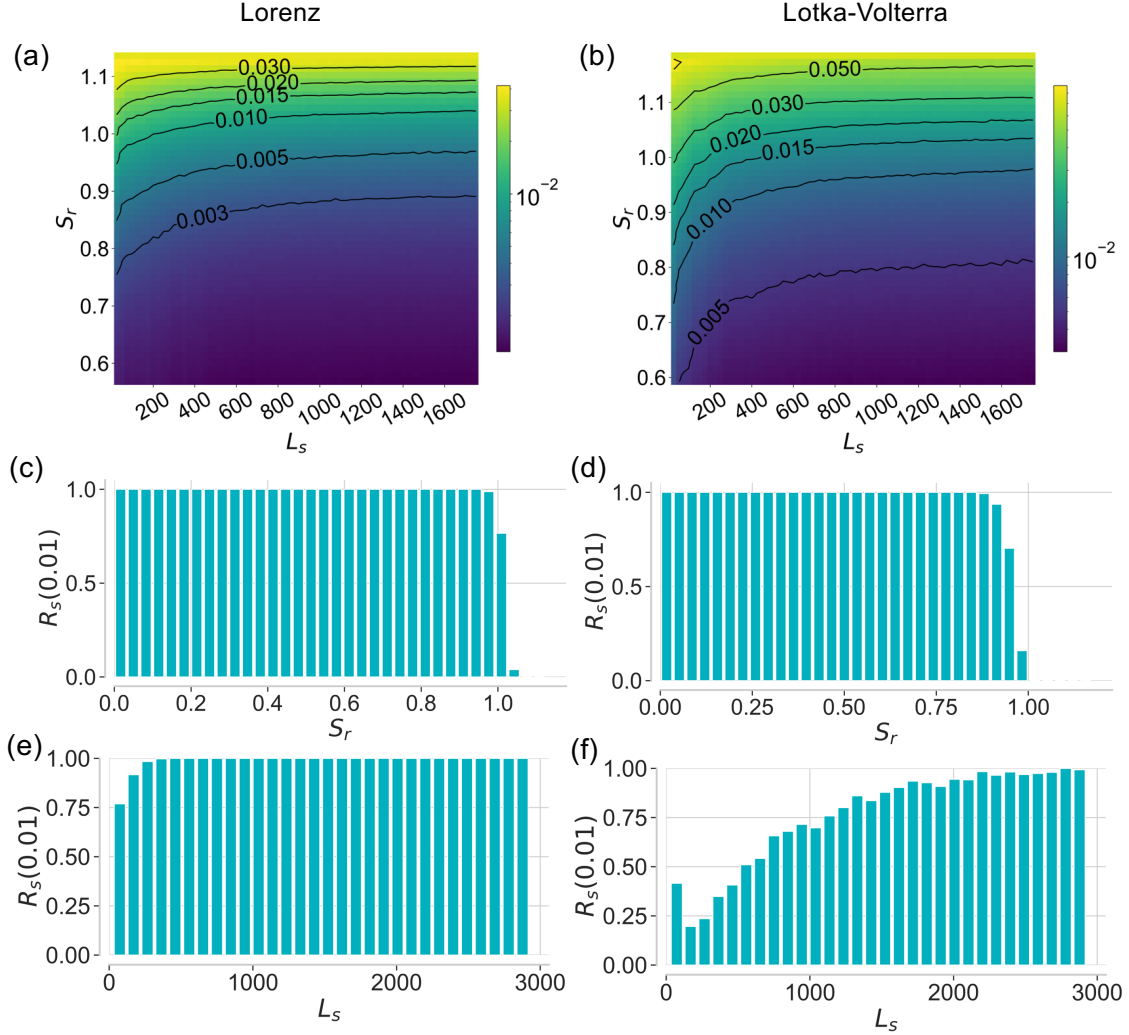


FIG. S11. Dynamics reconstruction performance. The two columns are the results from the Lorenz and Lotka-Volterra systems, respectively. The three rows from top to down present the color-coded averaged MSE in the parameter plane (L_s, S_r), the reconstruction stability R_s versus S_r and L_s for MSE 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)

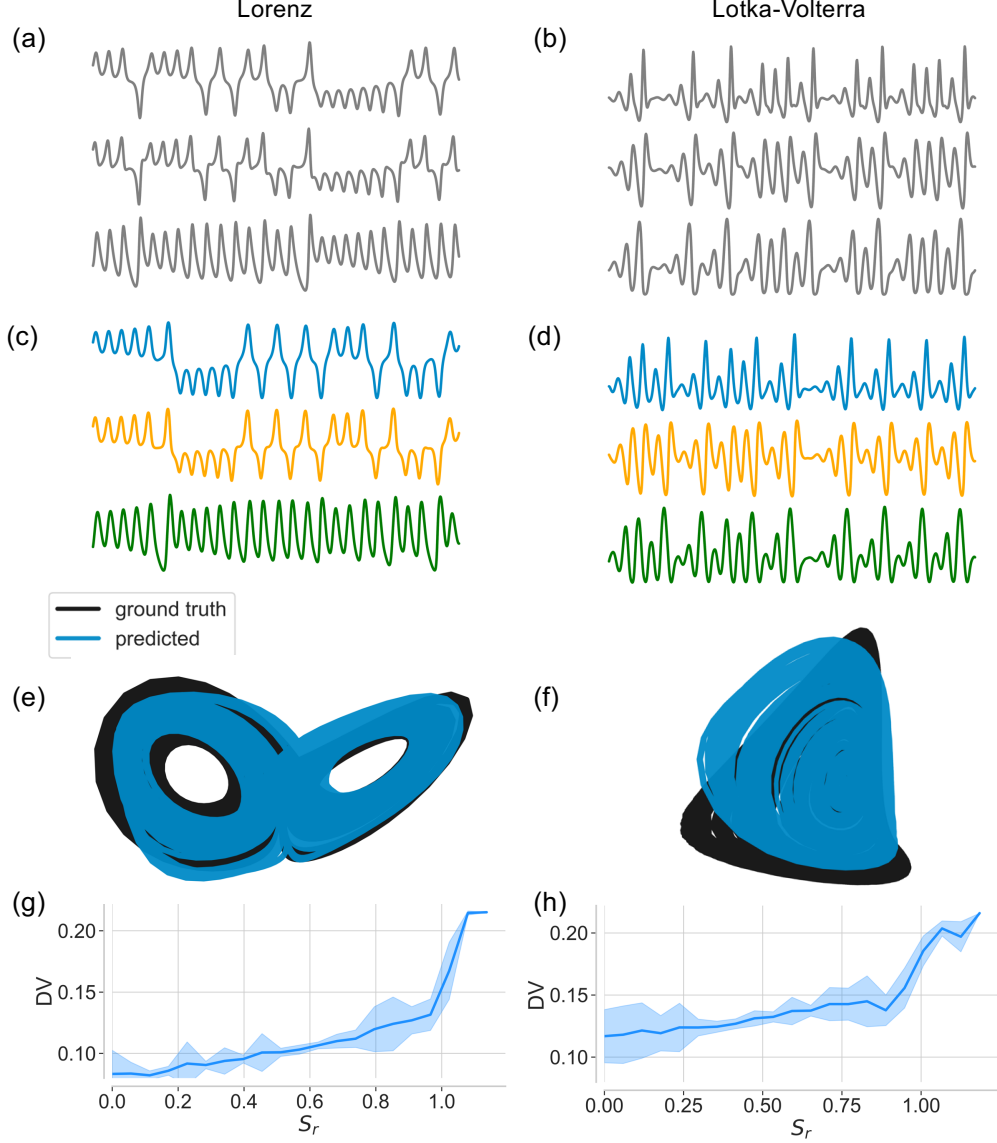


FIG. S12. Attractor reconstruction by reservoir computing. The first and second columns are for the Lorenz and Lotka-Volterra systems, respectively. (a,b) Segments of ground-truth time series, (c,d) short-term predictions by reservoir computing, (e,f) reconstructed long-term attractors as compared with 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 parameter [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-

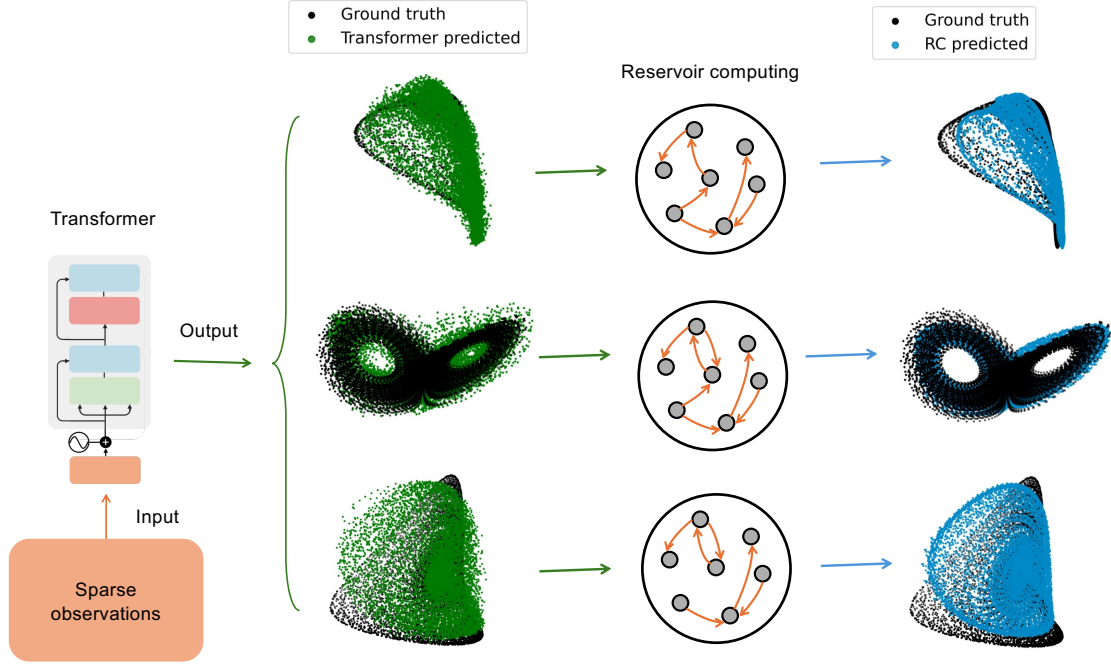


FIG. S13. Attractor reconstruction by the transformer alone and the hybrid transformer/reservoir computing framework. Sparse observations from target systems are provided to the transformer, which reconstruct the time series and present the attractors as green dots. The reconstructed time series by the 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 $K = 0.98$ to $K = 0.981$ 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 $K = 0.98$ (an example in main text), the second also with $K = 0.98$ but under noise of a moderate level σ , and the third with $K = 0.97$. 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

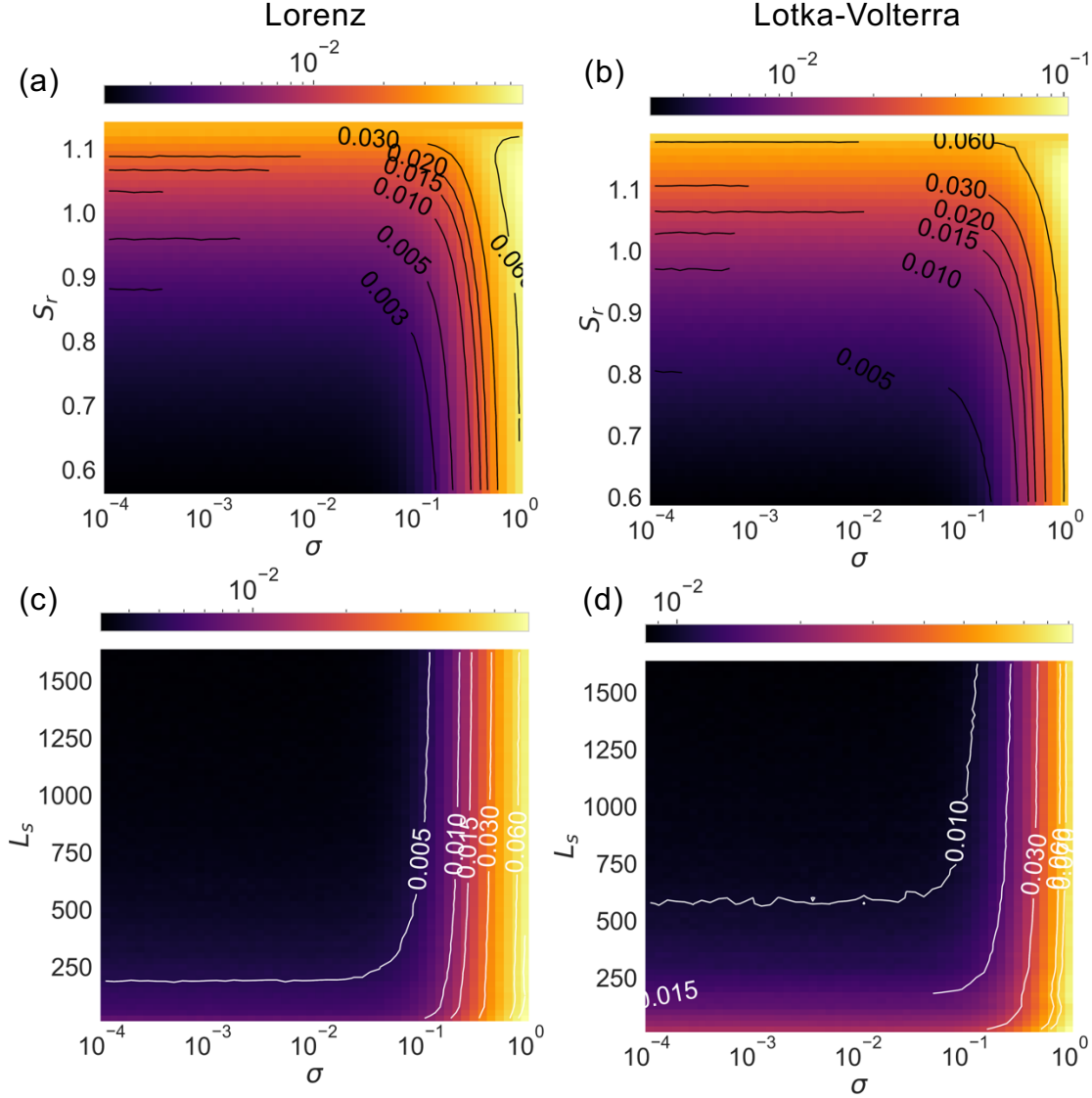


FIG. S14. Robustness against multiplicative noise. Two columns represent results from the Lorenz and Lotka-Volterra systems, respectively. The two rows show ensemble-averaged, color-coded MSE from 50 independent realizations, over the parameter planes (σ, S_r) and (σ, L_s) , respectively. The results 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 $K = 0.97$ and $K = 0.98$ align most closely with their corresponding ground-truth attractors, consistent with our previous findings. However, when noise is added to the observations at $K = 0.98$, the predicted attractor closely matches the ground-truth attractor of the noiseless $K = 0.98$ system, rather than the attractor at $K = 0.97$ or the noisy attractor at $K = 0.98$. 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-

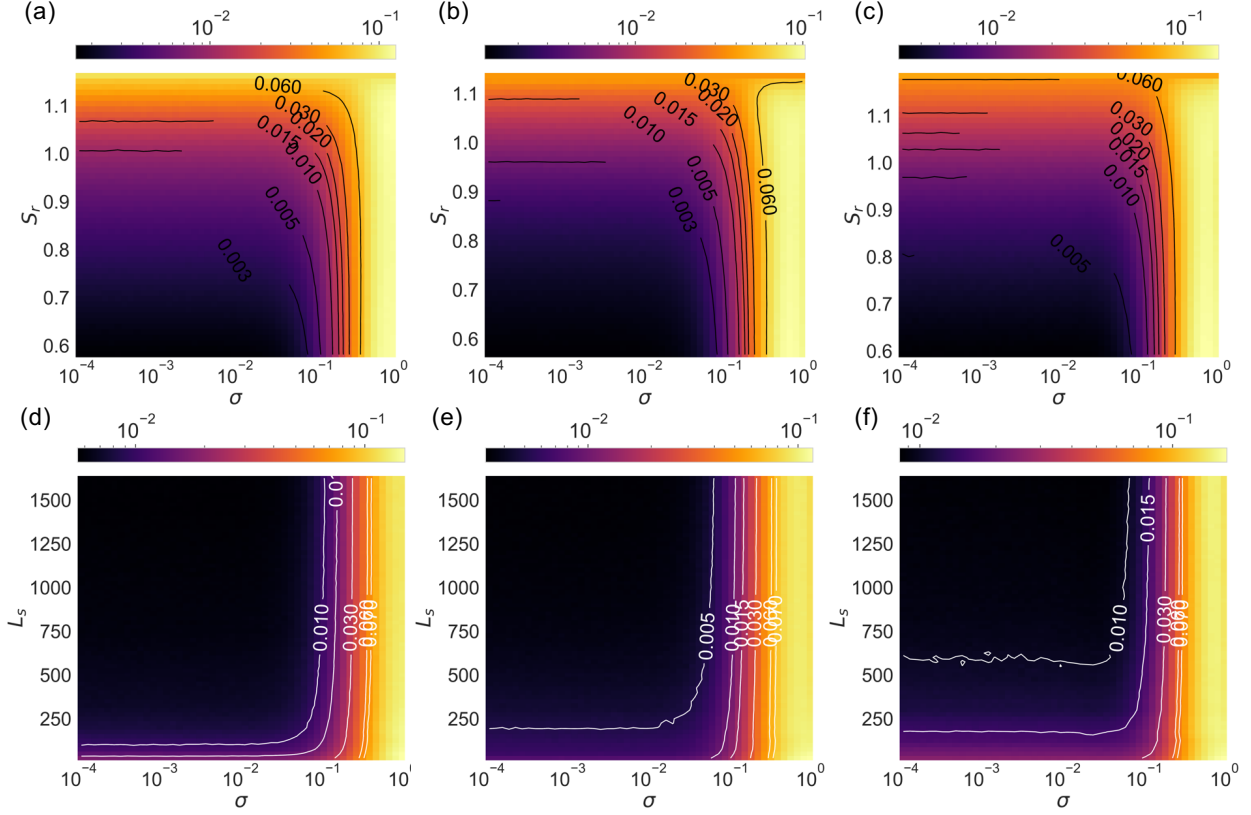


FIG. S15. Robustness against additive noise. The three columns represent results for the food chain, Lorenz, and Lotka-Volterra systems, respectively. The two rows show ensemble-averaged, color-coded MSE from 50 independent realizations, over the parameter planes (σ, S_r) and (σ, L_s) , respectively.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

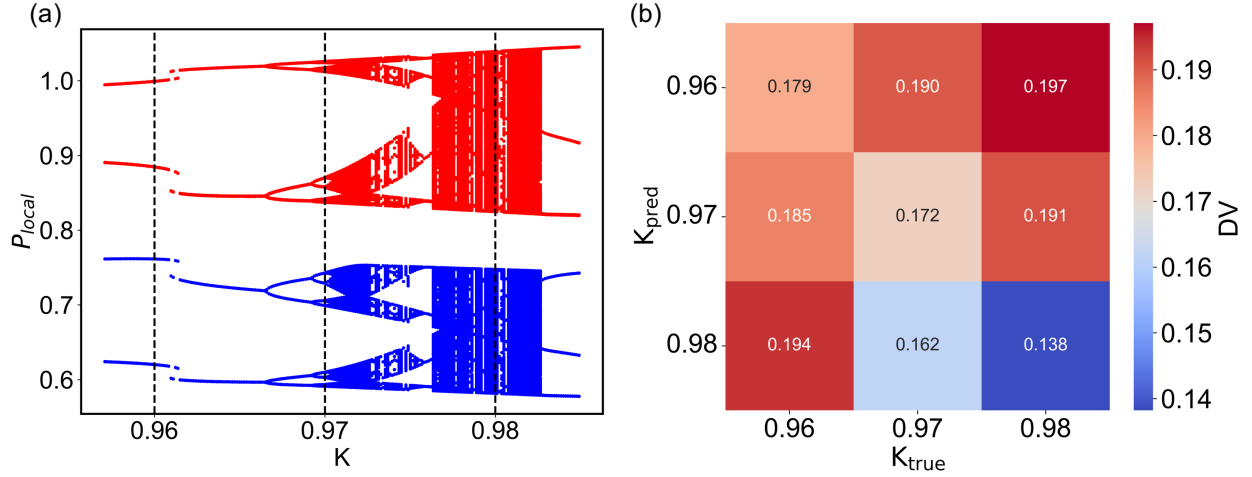


FIG. S16. Robustness against parameter variations. (a) Bifurcation diagram of the chaotic food chain system, with K as the bifurcation parameter. Red and blue dots represent the local maxima and minima, respectively. (b) Confusion matrix of the predicted and ground truth attractors, with DV averaged 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.

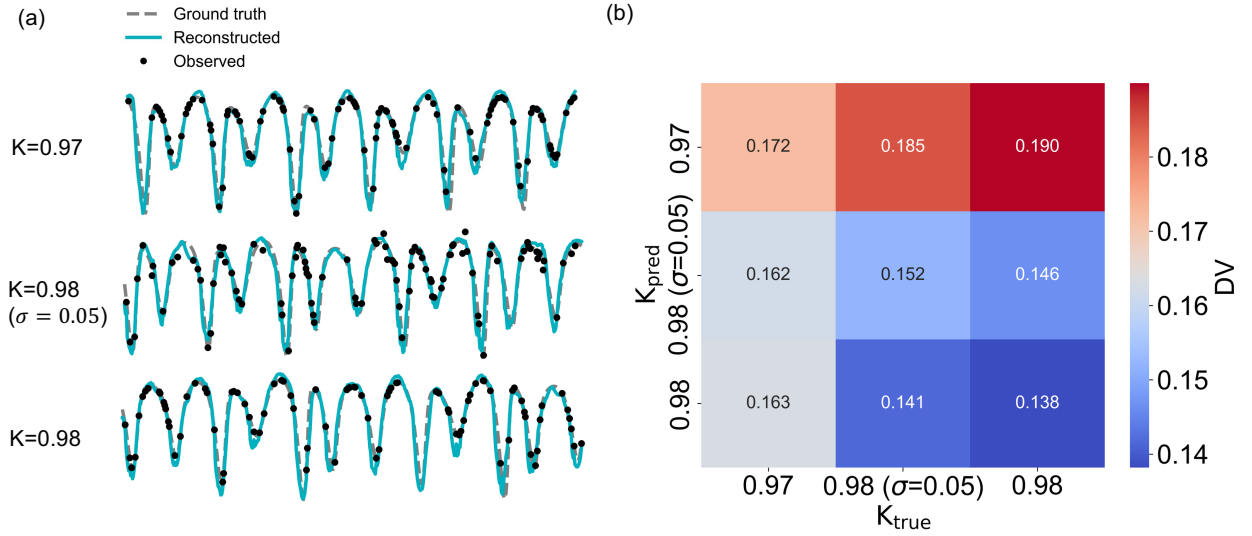


FIG. S17. Distinguishing the dynamics with parameter variations and observational noise. (a) Examples of the dynamics reconstruction of the chaotic food chain system, from top to bottom: $K = 0.97$, $K = 0.98$ with multiplicative noise of noise level $\sigma = 0.05$, $K = 0.98$. (b) Confusion matrix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

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-

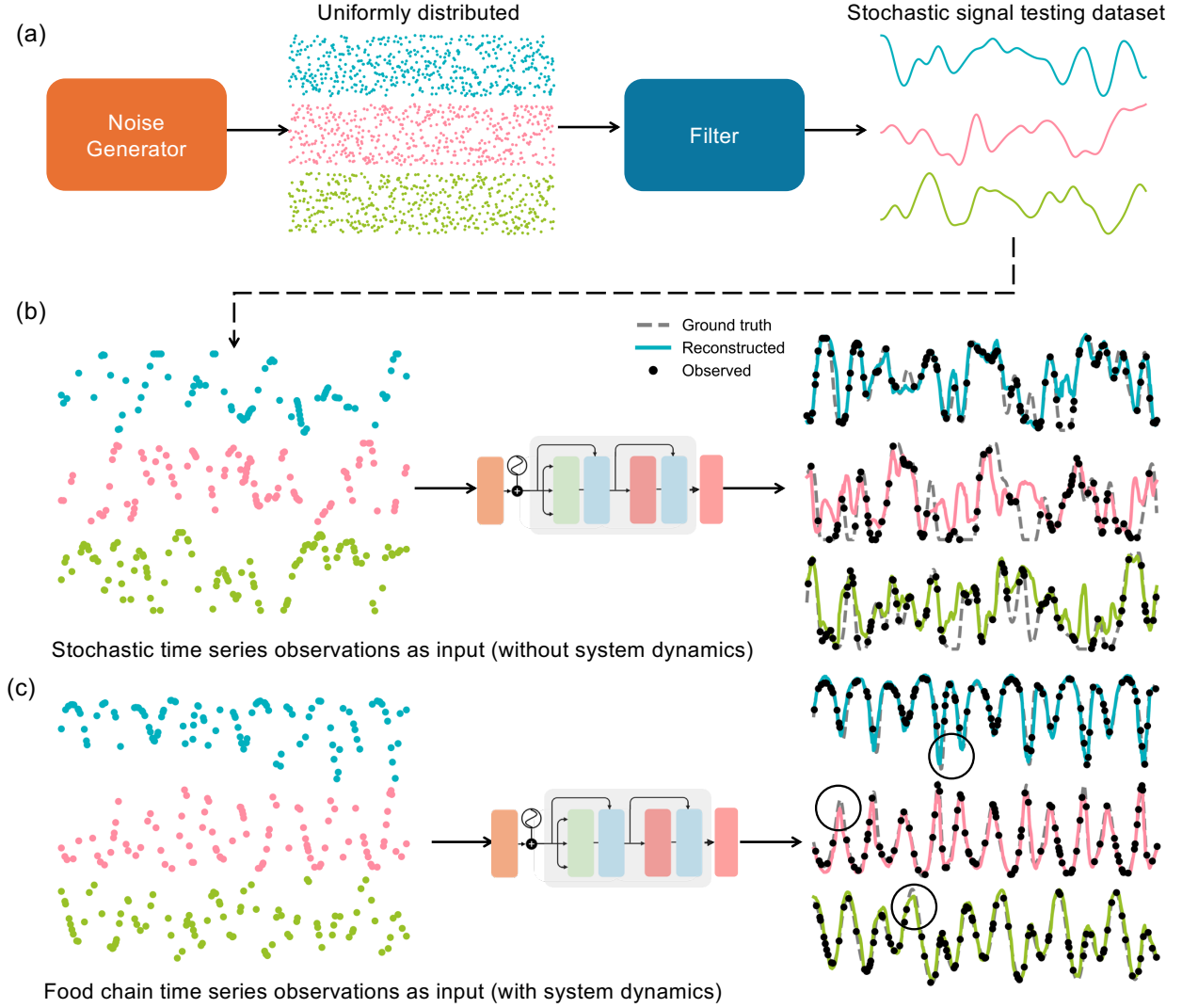


FIG. S18. Illustration of stochastic signal reconstruction. (a) Generation of the stochastic signal dataset. (b,c) Examples of reconstructing time series without and with underlying system dynamics by using a well-trained transformer for $L_s = 1200$ (about 25 cycles of oscillation) and $S_r = 0.93$.

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

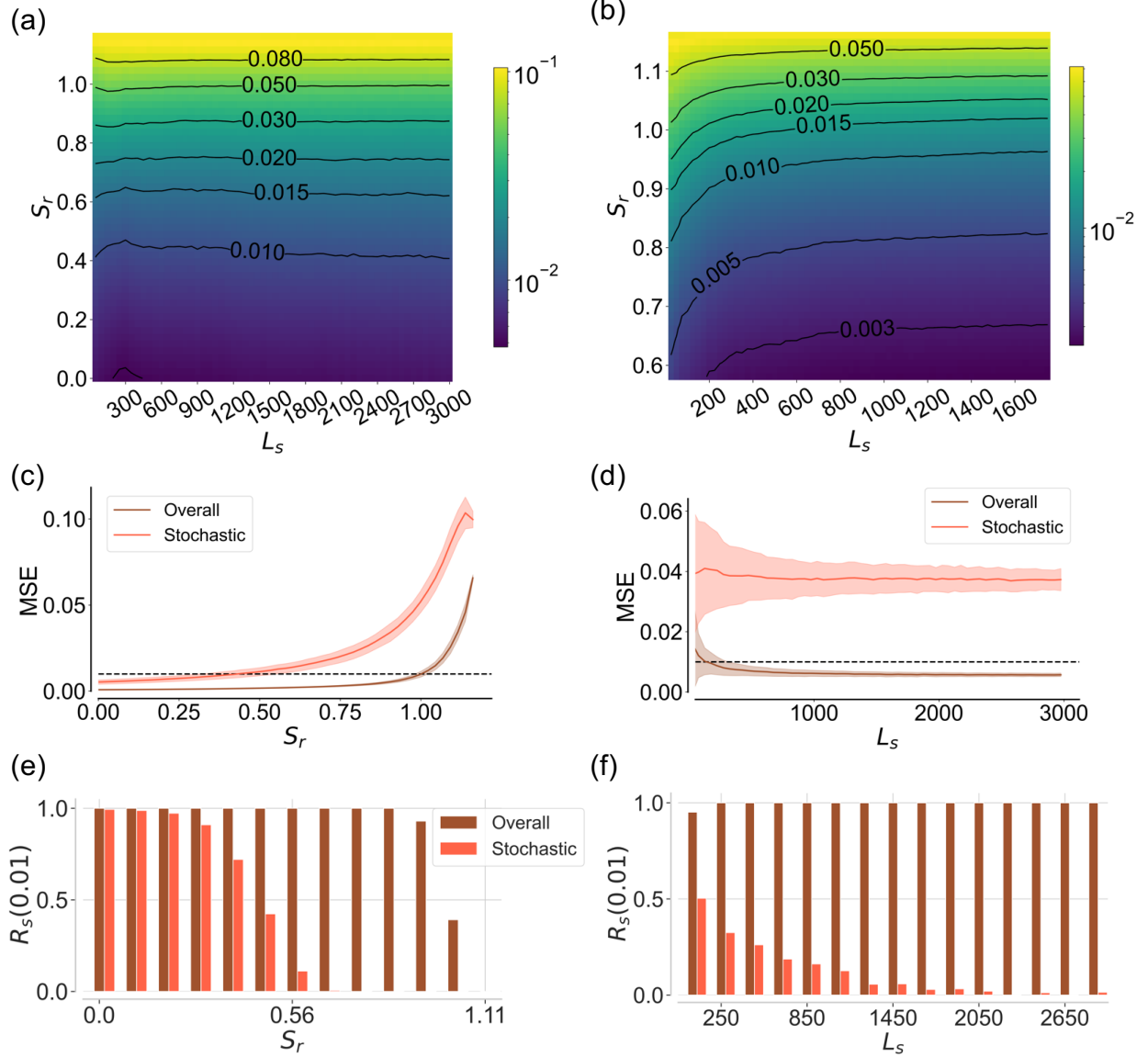


FIG. S19. Performance of transformer on stochastic signals and target systems. (a,b) Ensemble-averaged color-coded MSE in the parameter plane (L_s , S_r) for the stochastic and target dynamical systems, respectively, (c) MSE versus S_r for $L_s = 1200$ (around 25 cycles), (d) MSE versus L_s for $S_r = 0.93$, (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

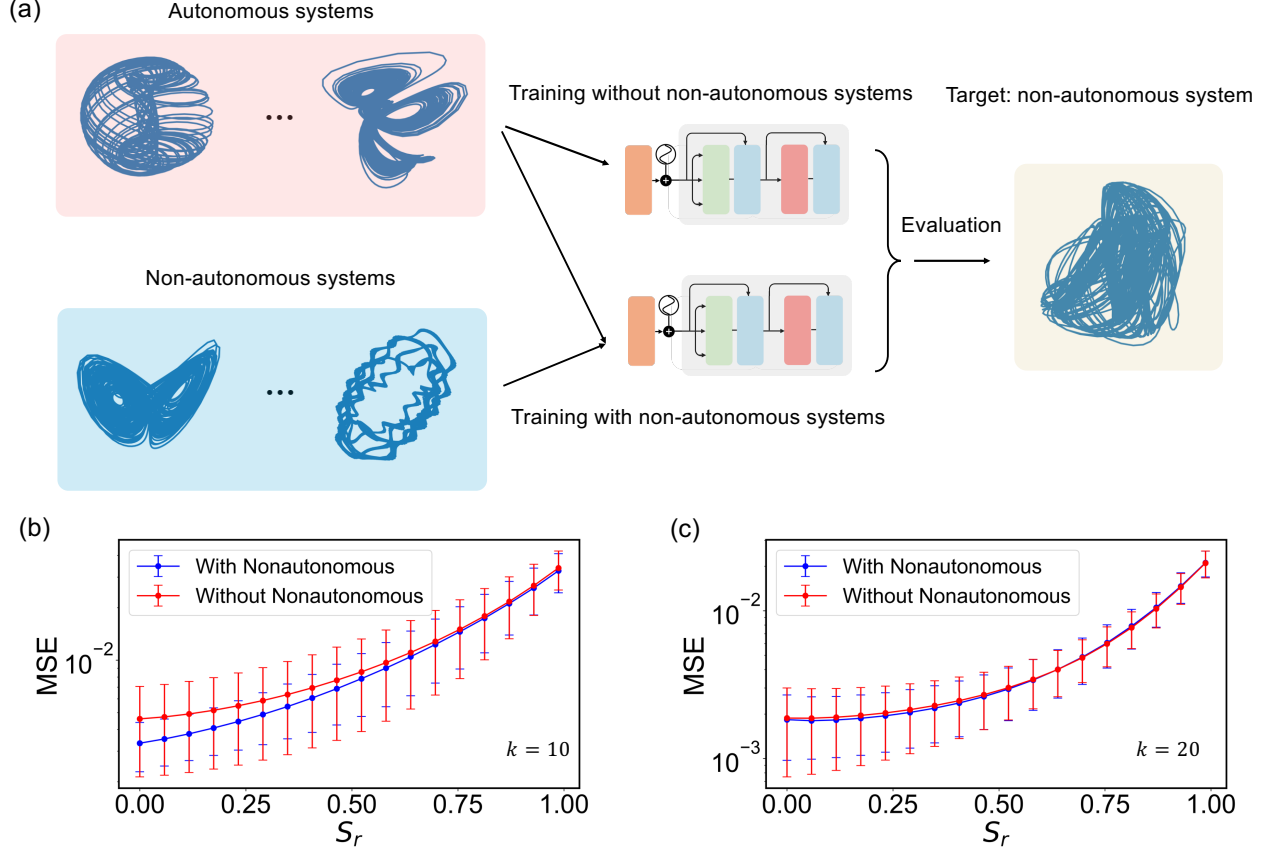


FIG. S20. Comparison of dynamic reconstruction training on autonomous and nonautonomous systems. (a) Illustration of two training strategies: training without and with nonautonomous systems. (b,c) Number of training systems is $k = 10$ and $k = 20$, respectively. In both cases, three nonautonomous systems 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.

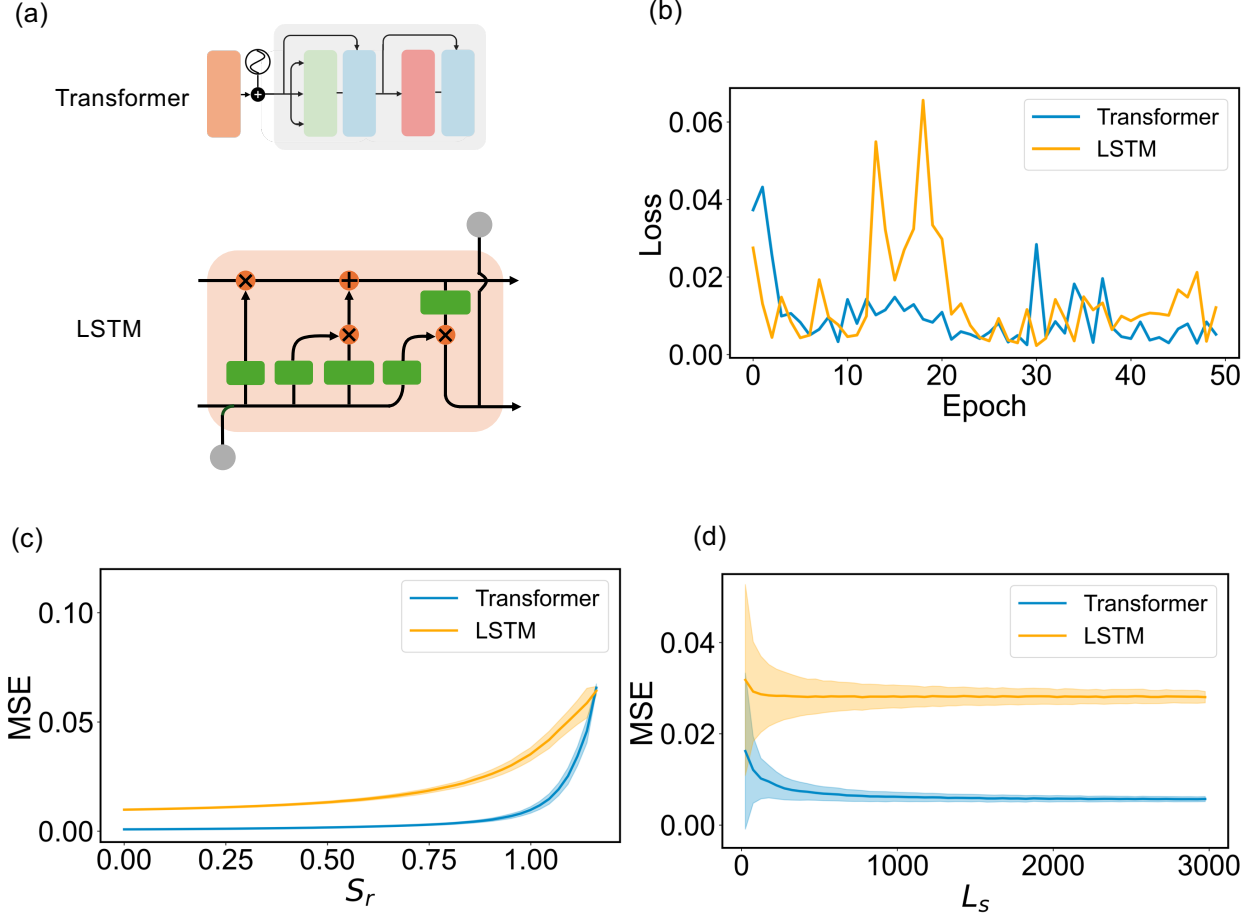


FIG. S21. Comparison of dynamics reconstruction performance between transformer and RNNs. (a) Basic structures of the three machine learning frameworks. (b) Training loss of transformer and LSTM. (c) MSE versus S_r for transformer and LSTM for $L_s = 1200$ (around 25 cycles). (d) MSE versus L_s for $S_r = 0.93$. 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-

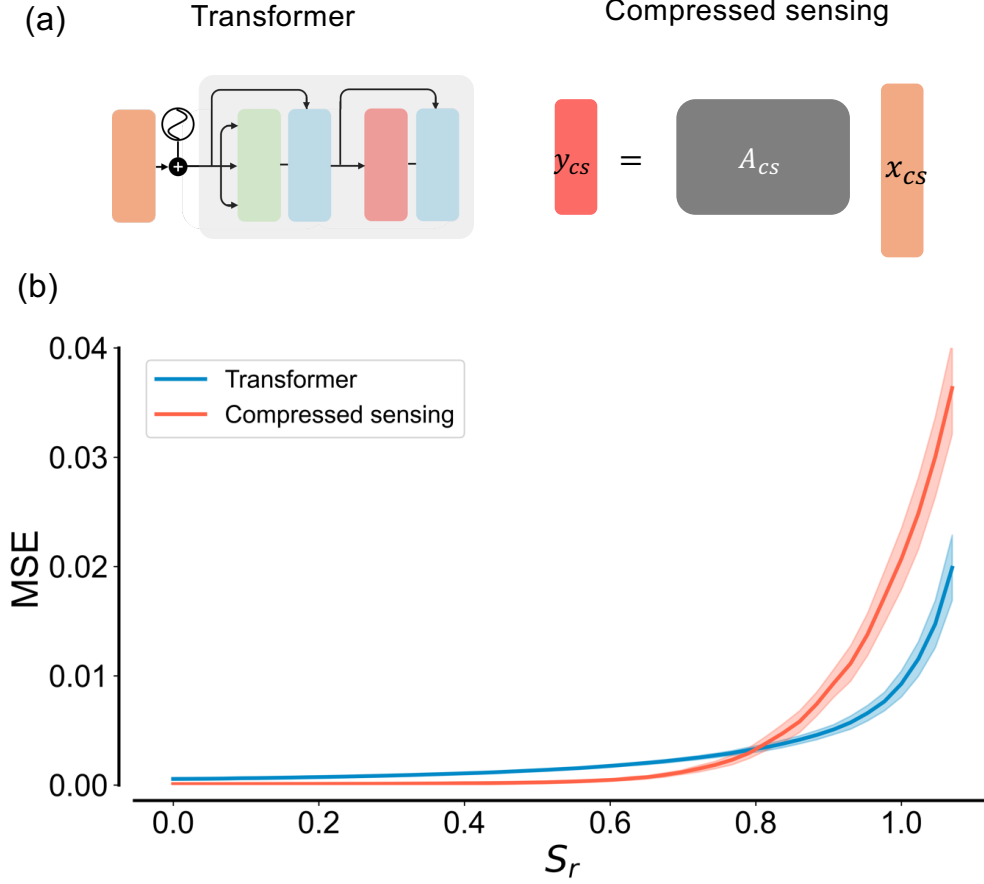


FIG. S22. Comparison of dynamics reconstruction performance by transformer and compressed sensing. (a) Basic structures of the two frameworks. (b) MSE versus S_r for transformer and LSTM for $L_s = 2000$ (around 40 cycles). Averaged MSEs are calculated from 400,000 data points (corresponding to about 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:

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

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	$\dot{x} = \alpha(y - x - ht)$ $\dot{y} = \gamma(x - y + z)$ $\dot{z} = -\beta y$	$\alpha = 15.6, \gamma = 1, \beta = 28$ $\mu_0 = -1.143, \mu_1 = -0.714$ $ht = \mu_1 x + 0.5(\mu_0 - \mu_1)(x + 1 - x - 1)$
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 ft) z $ $\dot{y} = -(1 + c + x^2)y + c - 1$ $\dot{z} = A \sin(2\pi ft)$	$a = 55, b = 167, c = 6.9$ $A = 19, f = 5.7$

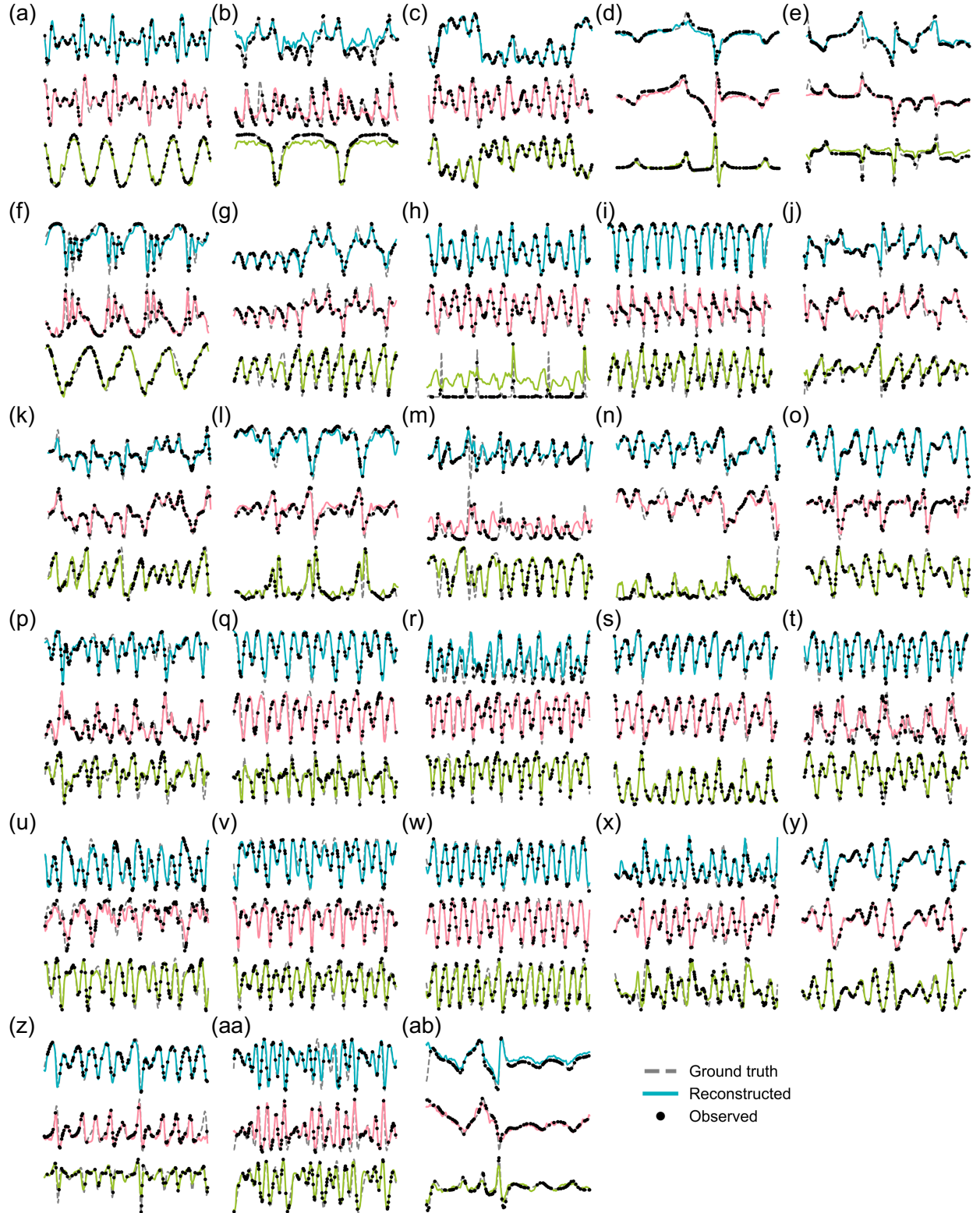


FIG. S23. Dynamics reconstruction of 28 different chaotic systems: (a) Aizawa, (b) Bouali, (c) Chua, (d) Dadras, (e) Four wing, (f) Hastings-Powell, (g) Rikitake, (h) Rossler, (i-aa) Sprott systems. (i-ab) Wang system, for $S_r = 0.93$ and $L_s = 1200$ (about 25 cycles of oscillation).

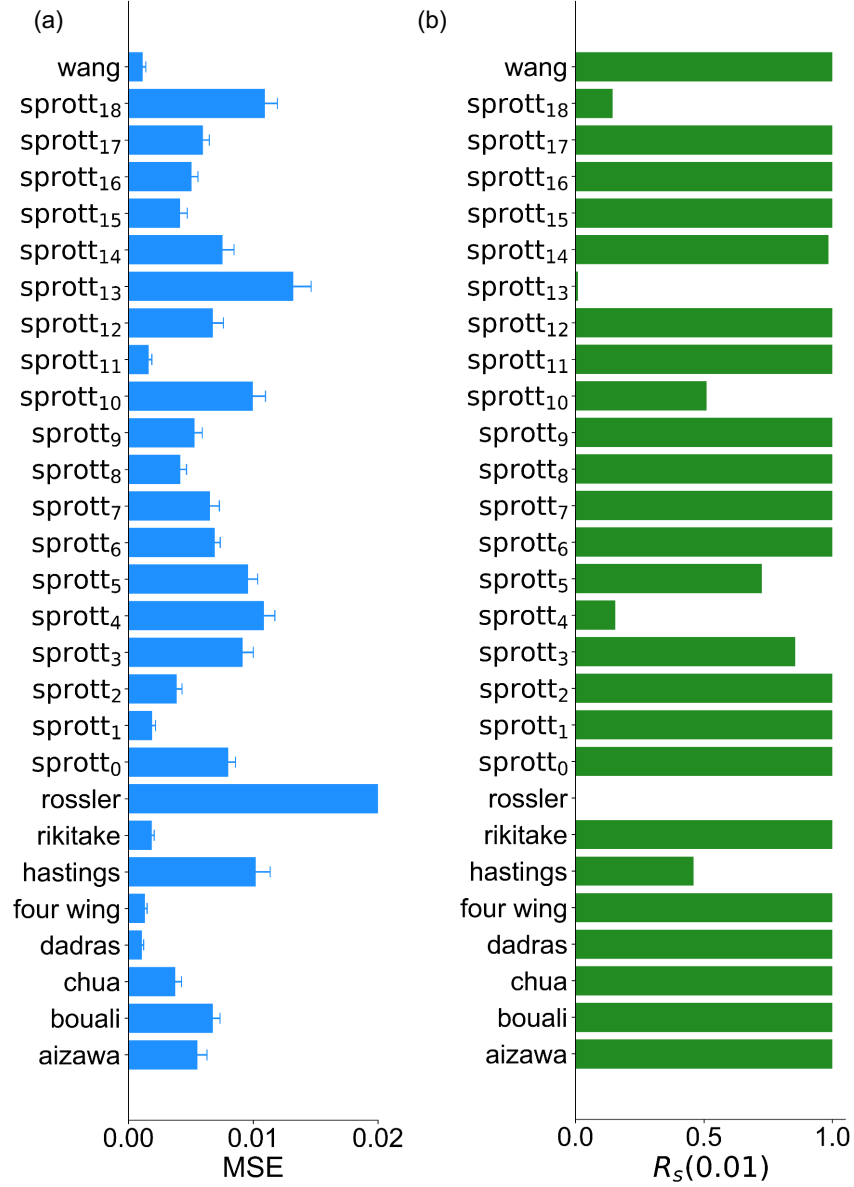


FIG. S24. Performance of dynamics reconstruction on the 28 chaotic systems. (a) Averages MSEs calculated from 400,000 data points (corresponding to about 8,000 cycles of oscillation) in each case, and error bars represent standard deviations. (b) Reconstruction stability $R_s(\cdot)$ for $S_r = 0.93$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

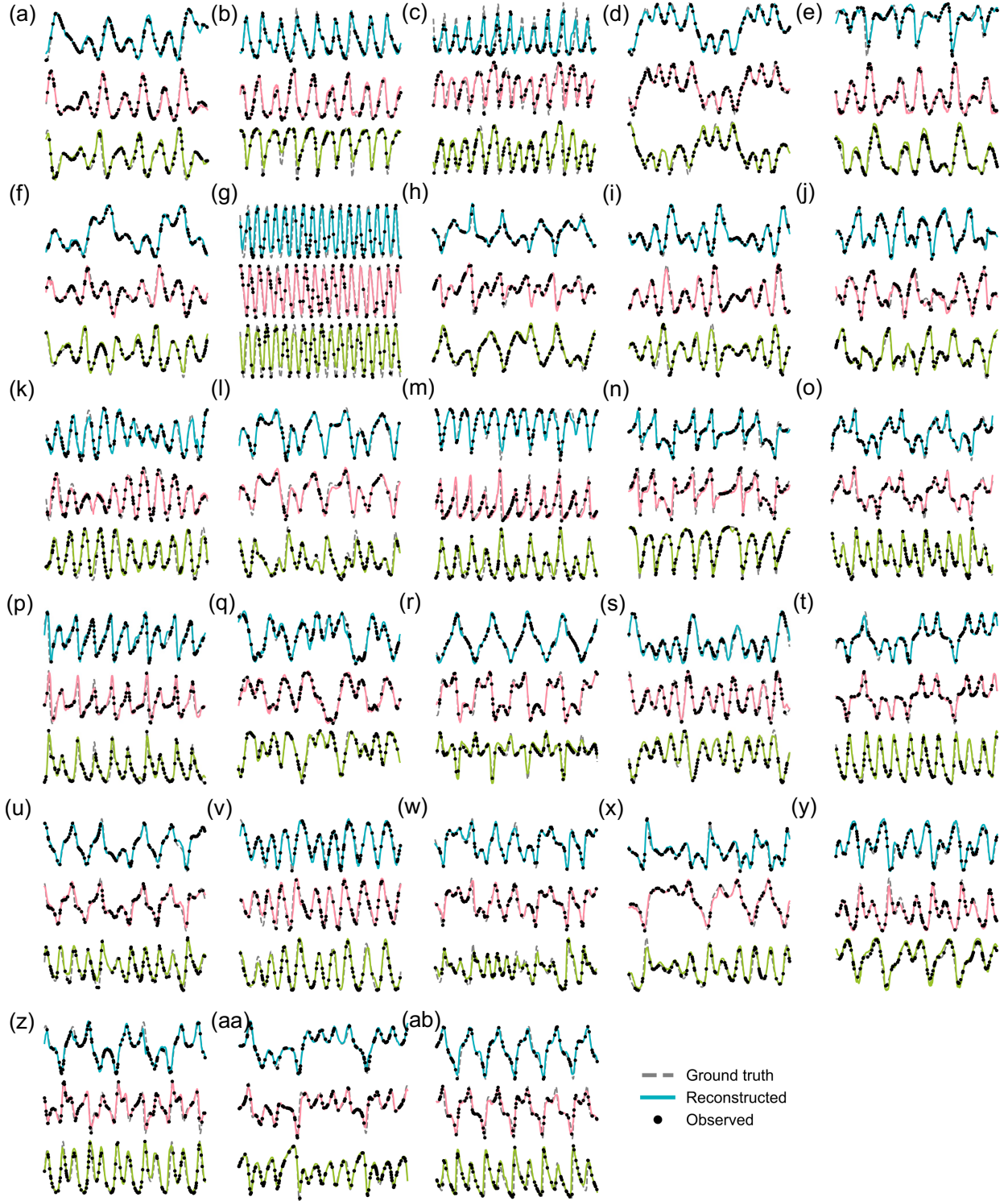


FIG. S25. Dynamics reconstruction of 28 additional unseen chaotic systems: (a) Arneodo, (b) CaTwoPlus, (c) CaTwoPlusQuasiperiodic, (d) CellularNeuralNetwork, (e) Colpitts, (f) Couillet, (g) DoubleGyre, (h) Ffinance, (i) GenesioTesi, (j) Halvorsen, (k) HenonHeiles, (l) HyperBao, (m) IsothermalChemical, (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).

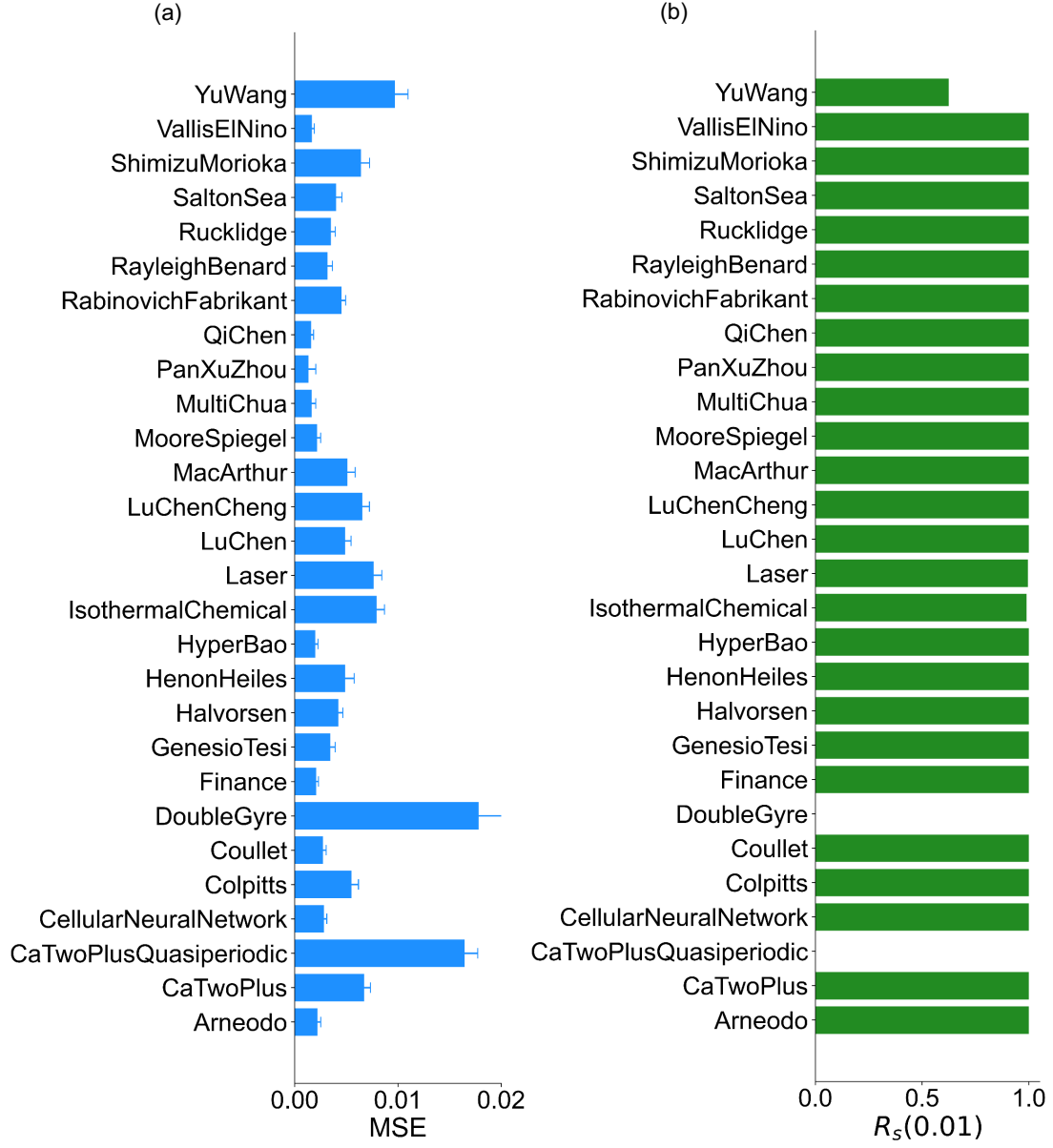


FIG. S26. Performance of dynamics reconstruction on the 28 additional unseen chaotic systems. (a) Averages MSEs calculated from 400,000 data points (corresponding to about 8,000 cycles of oscillation) in each case, and error bars represent standard deviations. (b) Reconstruction stability $R_s(\cdot)$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