nature communications

Article

https://doi.org/10.1038/s41467-025-61195-1

Learning dynamical systems with hit-and-run random feature maps

Received: 13 January 2025

Accepted: 16 June 2025

Published online: 01 July 2025

Check for updates

Pinak Mandal & Georg A. Gottwald

We show how random feature maps can be used to forecast dynamical systems with excellent forecasting skill. We consider the tanh activation function and judiciously choose the internal weights in a data-driven manner such that the resulting features explore the nonlinear, non-saturated regions of the activation function. We introduce skip connections and construct a deep variant of random feature maps by combining several units. To mitigate the curse of dimensionality, we introduce localization where we learn local maps, employing conditional independence. Our modified random feature maps provide excellent forecasting skill for both single trajectory forecasts as well as long-time estimates of statistical properties, for a range of chaotic dynamical systems with dimensions up to 512. In contrast to other methods such as reservoir computers which require extensive hyperparameter tuning, we effectively need to tune only a single hyperparameter, and are able to achieve state-of-the-art forecasting skill with much smaller networks.

Data-driven modelling of complex dynamical systems has sparked much interest in recent years, with remarkable success in, for example, weather forecasting, producing comparable or even better results than traditional operational equation-based forecasting systems[1–3]. Predicting chaotic dynamical systems with their inherent sensitivity to initial conditions is a formidable challenge. Direct numerical simulation of the underlying dynamical systems often requires small time steps and high spatial resolution due to the presence of multi-scale phenomena; moreover, the underlying equations may not even be known for some complex systems and scientists have to face a certain degree of model error. Substituting costly direct simulation of the underlying dynamical system by a surrogate model which is learned from data is an attractive alternative. Scientists have adopted recurrent networks as their go-to architecture for mimicking dynamical systems. Remarkably, more complex architectures such as Long Short-Term Memory (LSTM) architectures[4] have been replaced by much simpler architectures such as reservoir computers (RC) or Echo-State Networks (ESN)[5–7], exhibiting better forecasting capabilities with forecasting times exceeding several Lyapunov units[8,9]. Indeed, reservoir computing has emerged as the prominent architecture for modeling and predicting the behavior of chaotic dynamical systems[10–14]. Its appeal lies in the ability to process complex, high-dimensional data with relatively simple training procedures. Recently, it was shown that RCs can be further simplified in a variant resembling nonlinear vector autoregression machines, requiring fewer hyperparameters[15,16].

We consider here an even simpler version of RCs, which eliminates the internal dynamics of the reservoir and hence requires fewer parameters. These well-known random feature maps (RFMs)[17] can be viewed as a single-layer feedforward network in which the internal weights and biases are fixed, and the outer weights are determined by least-square regression. This approach simplifies the training process and reduces computational costs compared to fully trainable recurrent networks. RFMs have recently been shown to perform very well for learning dynamical systems[13,18–20]. RFMs enjoy the universal approximation property, and can, in principle, approximate any continuous function arbitrarily well[21–24]. This, however, does not tell a practitioner how to construct a random feature map model so that it well approximates smooth functions, and in particular, how to optimally choose the internal weights. Indeed, the performance of RFMs is sensitive to the random but fixed internal weights. Recently, there has been interest in finding approximate methods to choose the internal parameters to increase the forecasting capabilities of random feature maps[13,20,25]. We follow here our previously developed strategy[20] designed for tanh-activation functions, and employ a hit-and-run algorithm to initialize the non-trainable internal parameters ensuring that for the given training data the weights do not project the data into

University of Sydney, Sydney, Australia. e-mail: pinak.mandal@sydney.edu.au; georg.gottwald@sydney.edu.au

either the saturated region of the tanh-function or the approximately linear region. In the former case, the RFM would not be able to distinguish different data points, whereas in the latter case, the RFMs would reduce to a linear model, which would not be able to capture a nonlinear dynamical system.

In addition, we introduce several modifications to the classical RFMs. Rather than learning the propagator map, we formulate the learning problem to estimate tendencies instead. This is similar to skip connections in residual networks[26] and has recently been used in RCs[27]. We then formulate a deep variant of RFMs by constructing a succession of different RFMs that are individually trained. Together with the skip connection, this construction resembles an Euler discretization of a neural ODE[28]. A similar construction of multi-step learning has been applied to ESNs for forecasting[29] and classification problems[30,31]. RFMs suffer, like all kernel methods, from a curse of dimensionality, requiring an exponentially increasing amount of data for increasing dimension to achieve a specified degree of accuracy. To mitigate the curse of dimensionality, we employ a localization scheme, assuming that in typical dynamical systems, interactions are local and the learning problem can be restricted to a smaller dimensional local region rather than globally for the whole state space. Localization has the additional computational advantage of being parallelizable. Localization schemes have previously been applied to RCs, LSTMs, and generative models[8,10,14,32].

We evaluate our RFMs and the various modifications on three benchmark systems of increasing complexity: the 3-dimensional Lorenz-63 model, the 40-dimensional Lorenz-96 model and the Kuramoto-Sivashinsky equation as an example of a partial differential equation. These systems highlight the versatility of random feature models, which achieve state-of-the-art forecasting performance with one or more orders of magnitude fewer parameters and lower computational cost compared to RCs, making them powerful tools for prediction and analysis. We shall see that the width of the RFM needs to be sufficiently large in order to produce reliable features. Once RFMs are of a sufficiently large width, the forecasting performance of RFMs is increased more by increasing depth rather than increasing the width (when the total number of parameters is kept fixed).

Results

We consider a D-dimensional dynamical system which is observed at discrete times $t_n = n\Delta t$ with constant sampling time Δt. Given $N+1$ observations $\mathbf{u}_0, \mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_N$ with $\mathbf{u}_n = \mathbf{u}(t_n)$, our goal is to construct a surrogate model $\mathbf{\Psi}_{\Delta t}$ that approximates the map $\mathbf{\Psi}$: $\mathbf{u}_n \mapsto \mathbf{u}_{n+1}$ of the underlying dynamical system as closely as possible. We assume that our observations are complete and noise-free. We employ here random feature maps to construct the surrogate models. We begin with a description of classical random feature maps before introducing our modifications, namely skip connections and deep and localized variants.

We remark that the framework of RFMs can be extended to deal with the more realistic scenario of partial and noisy observations. Observational noise, which, when untreated, has a detrimental effect on learning with RFMs, can be controlled by combining the RFM learning task with an ensemble Kalman filter[33]. To overcome the implied non-Markovianity of a partially observed dynamical system, time-delay embedding techniques can be employed[18]. We do not consider these extensions here and restrict to noise-free and complete observations, which allow for better benchmarking.

Classical random feature maps

Random feature maps are feedforward neural networks consisting of an internal layer of width D_r and an external layer. We use tanh as the activation function for the internal layer. The weights $\mathbf{W}_{\text{in}}$ and biases $\mathbf{b}_{\text{in}}$ of the internal layer are drawn from some user-defined distribution and are kept fixed. The external layer weights $\mathbf{W}$ are learned. An RFM is compactly written as

$$\mathbf{u} \mapsto \mathbf{W}\tanh(\mathbf{W}_{\text{in}}\mathbf{u} + \mathbf{b}_{\text{in}}), \tag{1}$$

where $\mathbf{u} \in \mathbb{R}^D$, $\mathbf{W}_{\text{in}} \in \mathbb{R}^{D_r \times D}$, $\mathbf{b}_{\text{in}} \in \mathbb{R}^{D_r}$, and $\mathbf{W} \in \mathbb{R}^{D \times D_r}$. The surrogate map

$$\mathbf{\Psi}_{\Delta t}(\mathbf{u}_n) = \mathbf{W}\tanh(\mathbf{W}_{\text{in}}\mathbf{u}_n + \mathbf{b}_{\text{in}}) \tag{2}$$

provides an estimate for the observed $\mathbf{u}_{n+1}$. Training RFMs amounts to training the external layer by minimizing the following regularized cost,

$$\arg\min_{\mathbf{W}} \parallel \mathbf{W}\mathbf{\Phi}(\mathbf{U}) - \mathbf{U}'\parallel_F^2 + \beta \parallel \mathbf{W}\parallel_F^2, \tag{3}$$

where $\mathbf{U} \in \mathbb{R}^{D \times N}$ contains the observations $\{\mathbf{u}_n\}_{n=0}^{N-1}$ across its columns and $\mathbf{U}' \in \mathbb{R}^{D \times N}$ contains the time-shifted observations $\{\mathbf{u}_{n+1}\}_{n=0}^{N-1}$ across its columns. The feature matrix $\mathbf{\Phi}(\mathbf{U})$ denotes the output of the internal layer computed as $\mathbf{u} \mapsto \tanh(\mathbf{W}_{\text{in}}\mathbf{u} + \mathbf{b}_{\text{in}})$. The regularization parameter β is a hyperparameter which requires tuning. Here $\|\cdot\|_F$ denotes the Frobenius norm. The solution to the ridge regression problem (3) is explicitly given by

$$\mathbf{W} = \mathbf{U}'\mathbf{\Phi}^\top\left(\mathbf{\Phi}\mathbf{\Phi}^\top + \beta\mathbf{I}\right)^{-1}, \tag{4}$$

where we have omitted the dependency of the feature matrix $\mathbf{\Phi}$ on the data $\mathbf{U}$; in particular, no costly backpropagation is required. The quality of the learned surrogate model sensitively depends on the random initialization of the internal layer and the hyperparameter β.

We propose several modifications to the classical random feature maps that we will show significantly improve their forecasting capabilities. These modifications will allow random feature maps to outperform current state-of-the-art architectures at a fraction of previous computational costs. In particular, we will:

(1) employ a judicious choice of the random internal weights and biases ($\mathbf{W}_{\text{in}}$, $\mathbf{b}_{\text{in}}$);
(2) introduce skip connections, where we learn the tendencies $\mathbf{u}_{n+1} - \mathbf{u}_n$;
(3) introduce a deep architecture of sequentially arranged, individually trained, random feature map units;
(4) employ localization, where we learn several small local RFMs rather than a single large RFM, invoking conditional independence in the data, which significantly reduces the effective dimension.

We describe these modifications in detail in the Methods Section. RFMs with skip connections are labelled SkipRFM, and deep variants are referred to as DeepRFM or DeepSkip. Localized variants are labelled, for example, LocalDeepRFM or LocalDeepSkip. We recommend that the reader read the Methods section before continuing.

We evaluate our random feature surrogate models on three widely-used benchmark dynamical systems: the 3-dimensional Lorenz-63 system, the 40-dimensional Lorenz-96 system and the Kuramoto-Sivashinsky equation as an example of a partial differential equation, which we discretize with 512 gridpoints. For all three systems, we ensure that the training data and the test data evolve on the attractor by running simulations of the original dynamical system for a sufficiently long time.

To quantify the forecasting skill of our surrogate models, we compute the valid prediction time (VPT), described in the Methods Section. To obtain meaningful statistics of the forecast performance metric VPT, we generate 500 random realizations differing in the training data, the testing data and the non-trainable internal weights

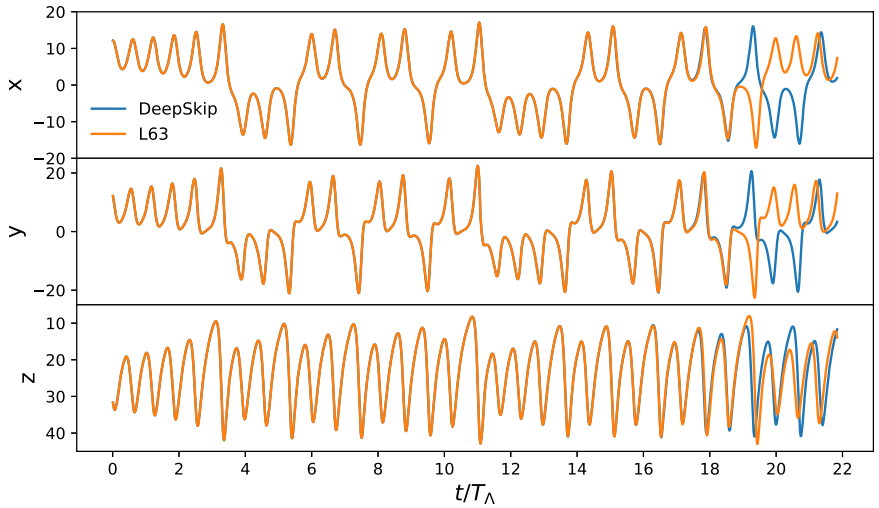

Fig. 1 | Forecasting an individual trajectory. An example of a forecast by a DeepSkip model with width $D_r = 1024$ and depth $B = 16$ for the L63 system (5). The surrogate model is able to forecast accurately up to VPT ≈ 19 Lyapunov time units.

and biases of the surrogate model. For each model, the regularization hyperparameter β is optimized via grid search.

To compare empirical histograms obtained from long-time simulations, Wasserstein distances W_2 are estimated from 3×10^4 random samples for each model using the Sinkhorn algorithm with an entropy regularization parameter of 10^{-2} [34].

Lorenz-63

In this section we demonstrate the forecasting skill and long-term behavior of surrogate models for the Lorenz-63 (L63) system with standard parameters[35],

$$\begin{aligned}\frac{dx}{dt} &= 10(y - x),\\ \frac{dy}{dt} &= x(28 - z) - y,\\ \frac{dz}{dt} &= xy - \frac{8}{3}z.\end{aligned} \tag{5}$$

The maximal Lyapunov exponent is estimated to be $\Lambda = 0.91$[14]. Localization is not required for this low-dimensional system, and we consider here the non-localized versions RFM, SkipRFM and DeepSkip.

Figure 1 shows a sample forecast of a DeepSkip model which is accurate up to approximately VPT ≈ 19 Lyapunov time units. However, there is a significant variability in the VPT due to the sensitivity to initial conditions of the chaotic L63 system. Figure 2A shows the distributions of VPT for training data of length $N = 5 \times 10^4$ sampled with $\Delta t = 0.01$, and a VPT error threshold value of $\varepsilon = 0.3$. It is seen that increasing the width D_r past $D_r = 512$ does not lead to an improvement of the mean forecast VPT for the shallow versions RFM and SkipRFM, which saturate around $\mathbb{E}[\text{VPT}] \approx 9.6$ for RFM, and slightly higher with $\mathbb{E}[\text{VPT}] \approx 10.6$ for SkipRFM. On the other hand, increasing the depth B consistently improves the performance of DeepSkip for each fixed width D_r. The best performing deep models are able to forecast approximately 1.4 Lyapunov time units longer compared to the best performing shallow models. The best mean forecast VPT is achieved for $D_r = 1024$ and depth $B = 32$ with $\mathbb{E}[\text{VPT}] = 12$. Deep models improve with depth even when the model size $S = (3D + 1)D_rB$ is kept fixed, as seen in Fig. 3A for two different model sizes. Since depth allows us to train larger models we are able to train deep models that are 3 times larger than the largest shallow model increasing the expressivity of the model. In the Supplementary Information (Section 5) we show that deep architectures allow for an order of magnitude faster training. Supplementary Tables 1 and 2 provide a comparison of our variants of the random feature map, including DeepRFM, for different model sizes, reporting on the mean, median, standard deviation of the VPT, as well as the maximum and minimum values. DeepSkip performs better than DeepRFM with a 2 Lyapunov units larger average VPT for $\Delta t = 0.01$.

Table 1A shows a comparison of our best performing DeepSkip models with recent benchmark results, highlighting that DeepSkip is able to achieve state-of-the-art forecast times with an order of magnitude smaller model size.

In general, finer temporal resolution is beneficial for learning the dynamics. In Fig. 4A we see the effect of increasing the sampling time to a fairly large value of $\Delta t = 0.1$, which is about a tenth of a Lyapunov time, on the forecasting skill for various models of nearly similar size. The deep model outperforms the shallow models by ~4.8 Lyapunov units on average. The mean VPT drops for RFM from 9.5 to 4.8, for SkipRFM from 10.4 to 4.8, and for DeepSkip from 11.4 to 9.6 when Δt is changed from 0.01 to 0.1. Smaller sampling times Δt allow for a better approximation of temporal derivatives and therefore SkipRFM outperforms RFM for small Δt. This advantage, however, vanishes at higher Δt and both perform equally. For RFM, SkipRFM, and DeepSkip, the mean VPT drops by 49.5%, 53.8% and 15.8%, respectively, indicating that deep architectures are least susceptible to the temporal resolution of the training data.

The effect of the sampling time Δt on the forecasting capability of RFMs has been previously studied by Levine and Stuart[13]. In particular, a standard RFM with an RFM for which the vector field of the underlying dynamical system is learned was compared[13]. The vector field was determined from $\dot{\mathbf{u}}_n$, which was computed from data $\mathbf{u}_n$ using splines. In Fig. 4B we compare these results for $D_r = 200$ with our implementation of a standard RFM using a hit-and-run algorithm and with SkipRFM, which as we discuss in the Methods Section, approximates the tendency via an explicit Euler discretization. To allow for a comparison with the results of Levine and Stuart[13] we use the validity time τ_f instead of the VPT, defined by

$$\tau_f = \min\{n\Delta t : ||\hat{\mathbf{u}}_n - \mathbf{u}_n||_2 \geq \gamma \overline{||\mathbf{u}||}_2\}, \tag{6}$$

where the mean $\overline{||\mathbf{u}||}_2$ is estimated from the training data. We use the same threshold $\gamma = 0.05$ as Levine and Stuart[13]. We show results for several values of the sampling time using a fixed integration time $T = 1000$, which implies that the larger sampling times correspond to smaller amounts of training data. Figure 4B illustrates two separate facets of RFMs. First, for small values of the sampling time Δt, the

https://doi.org/10.1038/s41467-025-61195-1

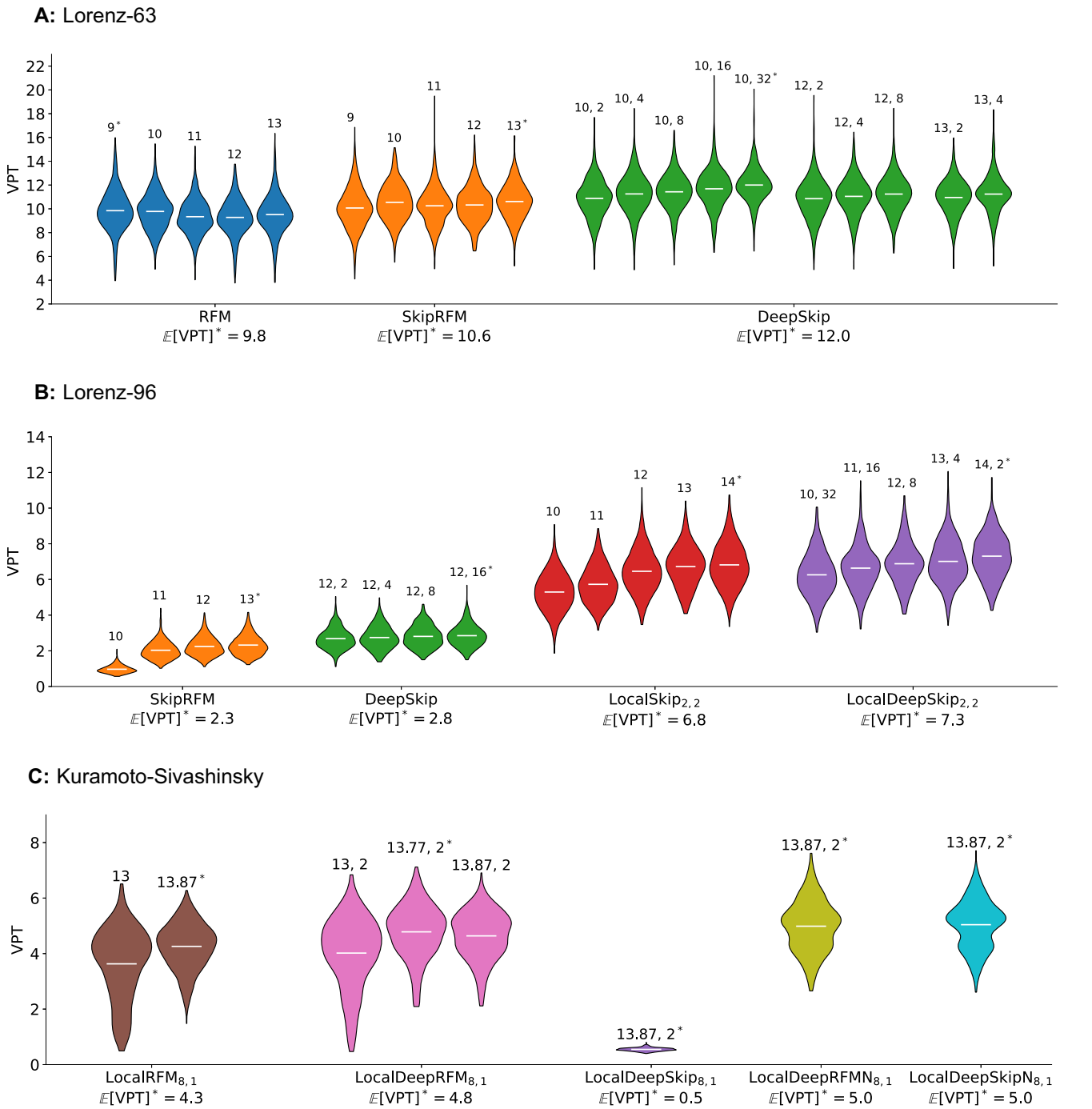

Fig. 2 | Statistics of short-term forecasting skill. Kernel density plots of VPT for **A** L63 system (5) for $(N, \Delta t, \varepsilon) = (5 \times 10^4, 0.01, 0.3)$, **B** L96 system (7) for $(N, \Delta t, \varepsilon) = (10^5, 0.01, 0.5)$ and **C** KS system (8) for $(N, \Delta t, \varepsilon) = (10^5, 0.25, 0.5)$. For shallow variants $\log_2(D_r)$ is indicated on the top of each density plot. For deep variants $(\log_2(D_r), B)$ is indicated on the top of each plot. For localized variants, the subscript denotes the values of G, I of the localization scheme (G, I). The *-symbol indicates the model with the best mean VPT within each architecture. The maximal value of D_r achievable by our GPU for the KS system is 15,000 (i.e. $\log_2(D_r) \approx 13.87$).

RFM which learns the vector field (labelled *rhs* (L+S)) outperforms the standard RFM (labelled *RFM (L+S)*), but this ordering changes for large values of the sampling time[13]. The deterioration of their *rhs (L+S)* method with a vanishing mean validity time for $\Delta t = 0.1$ can be related to the deterioration of the estimate of the time-derivative $\dot{\mathbf{u}}$ for large sampling times. In contrast, our SkipRFM, which does not learn the vector field but the tendency $\mathbf{u}_{n+1} - \mathbf{u}_n$ does not exhibit such deterioration at $\Delta t = 0.1$, and never performs worse than the standard RFM. Secondly, a Bayesian optimization algorithm to determine all hyperparameters can be used[13], including the internal weights, whereas we use the hit-and-run Algorithm 1 described in the Supplementary Information (Section 1). This leads to a superior performance at large values of Δt compared to our standard RFM. However, the hit-and-run algorithm performs significantly better for small sampling times.

The kernel density plots in Figs. 2A and 4A show a large degree of variance of VPT. This is to be expected for an underlying chaotic dynamics, and the distribution of VPT does not significantly change upon increasing the width D_r beyond a certain value. Ideally, one would like to shift the tails of the distribution of VPT towards larger

https://doi.org/10.1038/s41467-025-61195-1

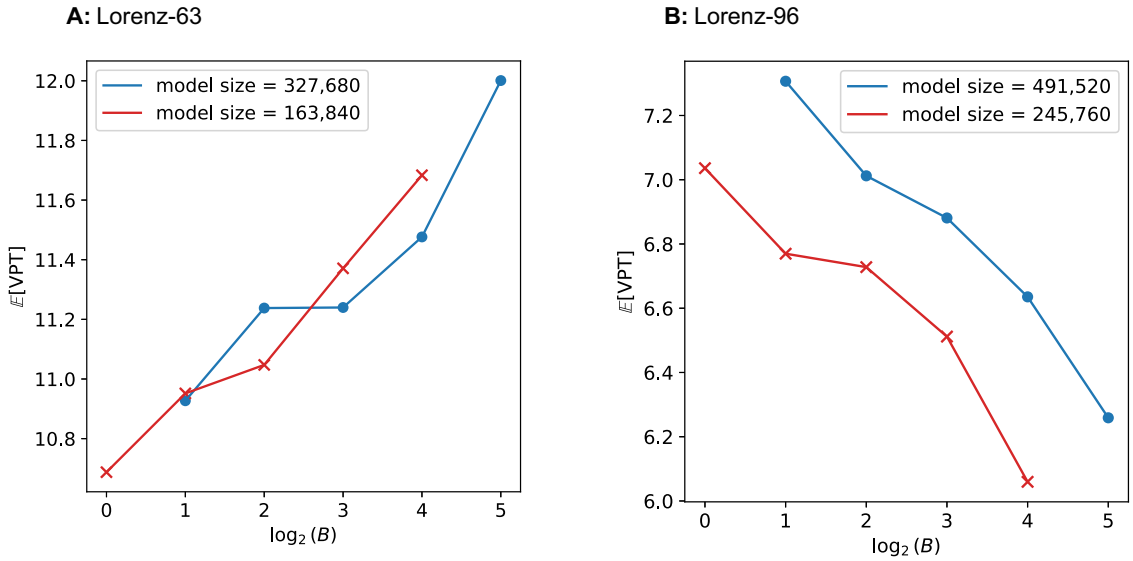

Fig. 3 | Effect of depth on forecasting skill. Mean VPT as a function of depth B for constant model size S. **A** L63 system (5) with DeepSkip. Along each curve the model size $S = (3D+1)D_rB$ remains constant and the width D_r decreases with depth. **B** L96 system (7) with LocalDeepSkip with localization scheme $(G, l) = (2, 2)$. Along each curve the model size $S = (\hat{D}+G+1)D_rB$ with $\hat{D} = 2G(l+1)$ remains constant and the width D_r decreases with depth.

Table 1 | Comparison of mean VPT and corresponding model sizes from recent benchmark results for forecasting

A: Lorenz-63						
Source	**Model**	**$\log_{10}$ (model size)**	**$\mathbb{E}$ [VPT]**	***N***	**Δ*t***	**ε**
Akiyama et al. (2022)[29]	Multi-step ESN	5.05	9.3	2×10^4	0.02	0.4
Platt et al. (2022)[14]	RC	6.60	11.8–12.0*	5×10^4	0.01	0.3
Koster et al. (2023)[56]	DI-RC (SINDy)	6.00	4.0	10^4	0.01	$\sqrt{0.4}$
Our work	DeepSkip RFM	5.52*	12.0*	5×10^4	0.01	0.3
Our work	DeepSkip RFM	5.52	11.8	2×10^4	0.02	$\sqrt{0.05}$
B: Lorenz-96						
Source	**Model**	**$\log_{10}$ (model size)**	**$\mathbb{E}$ [VPT]**	***N***	**Δ*t***	**ε**
Penny et al. (2022)[14,57 2]	RC	7.56	2.5–2.8	2×10^5	0.01	0.5
Vlachas et al. (2022)[8]	Localized LSTM	5.95	3.9	10^5	0.01	0.5
Platt et al. (2022)[14]	Localized RC	7.03	6.5–6.8	4×10^4	0.01	0.5
Our work	LocalDeepSkip RFM	5.69*	7.3*	10^5	0.01	0.5
C: Kuramoto-Sivashinsky						
Source	**Model**	**$\log_{10}$ (model size)**	**$\mathbb{E}$ [VPT]**	***N***	**Δ*t***	**ε**
Vlachas et al. (2022)[8]	Localized RC	8.77	4.8	10^5	0.25	0.5
Our work	LocalDeepSkipN RFM	6.09*	5.0*	10^5	0.25	0.5

The results corresponding to the best mean VPT are reported for each source. The *-symbol indicates the largest $\mathbb{E}$[VPT] and the corresponding smallest model sizes. **A**: The 3-dimensional L63 system (5). **B**: The 40-dimensional L96 system (7). **C**: The KS system (8) with 512 spatial grid points on a domain of length $L = 200$.
[2]We remark that the results listed as Penny et al. were reported in Figure 14 and Table 13 of Platt et al.[14] rather than in the original work[57].

forecast times and avoid occasional short forecast times. In fact, the hit-and-run algorithm achieves this: the presence of features which correspond to the linear and/or saturated region of the activation function, contribute to a higher variance of VPT (see Figure 9 in our previous work[20]).

Besides being able to track individual trajectories, surrogate models need to produce reliable long-term predictions of the statistical features of the underlying dynamical system. Figure 5A compares the marginal densities estimated from the invariant measures of the original L63 system (5) and the learned surrogate models. The data shown were generated with long simulations spanning 910 Lyapunov time units. All three surrogate models are able to reproduce the long-term statistics of the L63 system equally well with comparable Wasserstein distances.

Lorenz-96

In this section we demonstrate the forecasting skill for the Lorenz-96 (L96) system

$$\frac{dx_i}{dt} = (x_{i+1} - x_{i-2})x_{i-1} - x_i + F,\ i = 1, 2, \cdots, D, \tag{7}$$

with dimension $D = 40$, forcing $F = 10$ and periodic boundary conditions $x_{i+D} = x_i$[36]. The maximal Lyapunov exponent is estimated to be $\Lambda = 2.27$[8]. We consider here SkipRFM and DeepSkip, and their localized counterparts LocalSkip and LocalDeepSkip. We do not show results for RFM and LocalRFM as their performance is comparable to SkipRFM and LocalSkip. We will see that for this 40-dimensional dynamical system, localization is the dominant factor in ensuring good performance.

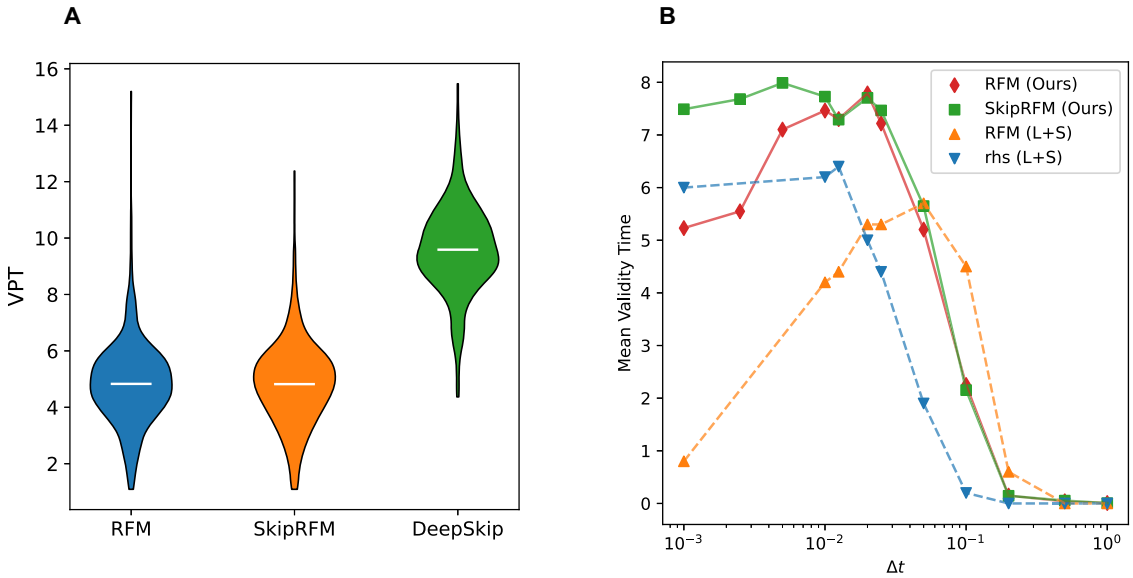

Fig. 4 | Effect of sampling time on forecasting skill. A Kernel density plots of VPT for the L63 system (5) for $(N, \Delta t, \varepsilon) = (5 \times 10^4, 0.1, 0.3)$. Sizes of the models are $S = 114{,}688$, $S = 114{,}688$ and $S = 114{,}560$, from left to right, with, $D_r = 16{,}384$, $D_r = 16{,}384$ and $D_r = 716$, respectively. The DeepSkip model has depth $B = 16$. **B** Mean validity time τ_f as a function of the sampling time Δt for the L63 system (5) with $D_r = 200$. Each model was trained on a time-series spanning $T = 1000$ time units. and hence the length of the training data N decreases with increasing $\Delta t = T/N$. Averages for our models were computed using 500 realizations differing in the training data, the testing data and the non-trainable internal weights. The mean validity times labelled as (L+S) are taken from Fig. 5 in Levine and Stuart[13]. Note that we show results for two extra values of $\Delta t = 2.5 \times 10^{-3}, 5 \times 10^{-3}$, which are not present in Levine and Stuart[13].

Figure 2B shows the distribution of VPT for these models for $N = 10^5$, $\Delta t = 0.01$ and $\varepsilon = 0.5$. We choose a localization scheme with $(G, I) = (2, 2)$; see the Methods Section for the definition of G and I and the Supplementary Information (Section 3) for different localization schemes and general guidelines for selecting an optimal localization scheme. The positive effect of localization is clearly seen with roughly 3 times longer forecasting times when compared to the respective non-localized versions. We achieve an optimal mean VPT with $\mathbb{E}[\text{VPT}] = 7.3$ for LocalDeepSkip with $D_r = 16{,}384$ and $B = 2$. The largest localized models that we could accommodate on the GPU were twice as wide as the largest non-localized models, allowing for a much greater expressivity. The performance of non-localized models plateaus quickly with increasing model width or depth, whereas we run out of GPU memory before observing saturation in the forecasting skill of the localized models. The best deep models are able to forecast approximately 0.5 Lyapunov time units longer than their shallow counterparts for both localized and non-localized models.

Unlike for the L63 system, the performance of deep models decreases with increasing depth when the model size $S = (\hat{D} + G + 1)D_r B$ with $\hat{D} = 2G(I + 1)$ is kept fixed, as seen in Fig. 3B for two different model sizes. We see that shallow but wide models perform better than deeper models of the same size by approximately 1 Lyapunov time unit. We believe that this is due to the more complex nature of the L96 system. The learning task requires (for given data length N) a sufficiently large internal layer width D_r to ensure reliable forecasting at each of the B layers of a deep architecture. Since increasing the depth B for fixed model size S implies a decrease in the width D_r, the deeper networks are not able to resolve the dynamics to sufficient accuracy at each layer. Hence, deeper architectures with $B > 1$ are only beneficial once the width D_r is sufficiently large such that the forecasting skill has saturated. The Supplementary Information (Section 4) explores the interplay between the width D_r and the depth B for the L63 system and the L96 system supporting this claim.

Table 1B shows a comparison of our best performing LocalDeepSkip model with recent benchmark results, highlighting that LocalDeepSkip achieves state-of-the-art forecast times with $\mathbb{E}[\text{VPT}] = 7.3$ at 1.3 orders of magnitude smaller model size. In Supplementary Tables 3 and 4 a comparison of our variants of random feature maps is shown for different model sizes, reporting on the mean, median, standard deviation of the VPT as well as the maximal and minimal values.

Figure 5B compares the empirical marginal densities estimated from the invariant measures of the original L96 system (7) and the learned surrogate models. Both the non-localized and the localized variants are able to reproduce the long-term statistics of the L96 system equally well with comparable Wasserstein distances.

Kuramoto-Sivashinsky

We further consider the Kuramoto-Sivashinsky (KS) equation

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + \frac{\partial^2 u}{\partial x^2} + \frac{\partial^4 u}{\partial x^4} = 0 \tag{8}$$

for $x \in [0, L]$ with periodic boundary conditions $u(0, t) = u(L, t)$, as an example of a partial differential equation exhibiting spatio-temporal chaos[37,38]. The data are subsampled in time to produce a time series of 512-dimensional states of length $N = 10^5$ with sample time $\Delta t = 0.25$ for the learning task. The maximal Lyapunov exponent is estimated to be $\Lambda = 0.094$[8]. We solve equation (8) on a domain of length $L = 200$ with a uniform grid of 512 nodes using the ETDRK4 method[39] with a time step of $h = 0.001$. We employ a VPT error threshold of $\varepsilon = 0.5$. For this high-dimensional system, localization is essential to obtain any reliable forecasting skill. We choose a localization scheme of $(G, I) = (8, 1)$; see the Supplementary Information (Section 3) for different localization schemes and general guidelines for selecting an optimal localization scheme. To allow for a large expressive model capable of capturing the complexity of the chaotic dynamics, we focus mainly on deep architectures.

The trajectory data for KS generated by the ETDRK4 algorithm has a large condition number ~10^{15}. Ill-conditioned data matrices imply ill-conditioned learned outer weight matrices $\mathbf{W}$ which has a catastrophic effect on long-term forecasts and might also affect short-term forecasts, see the discussion on dealing with ill-conditioned data in the

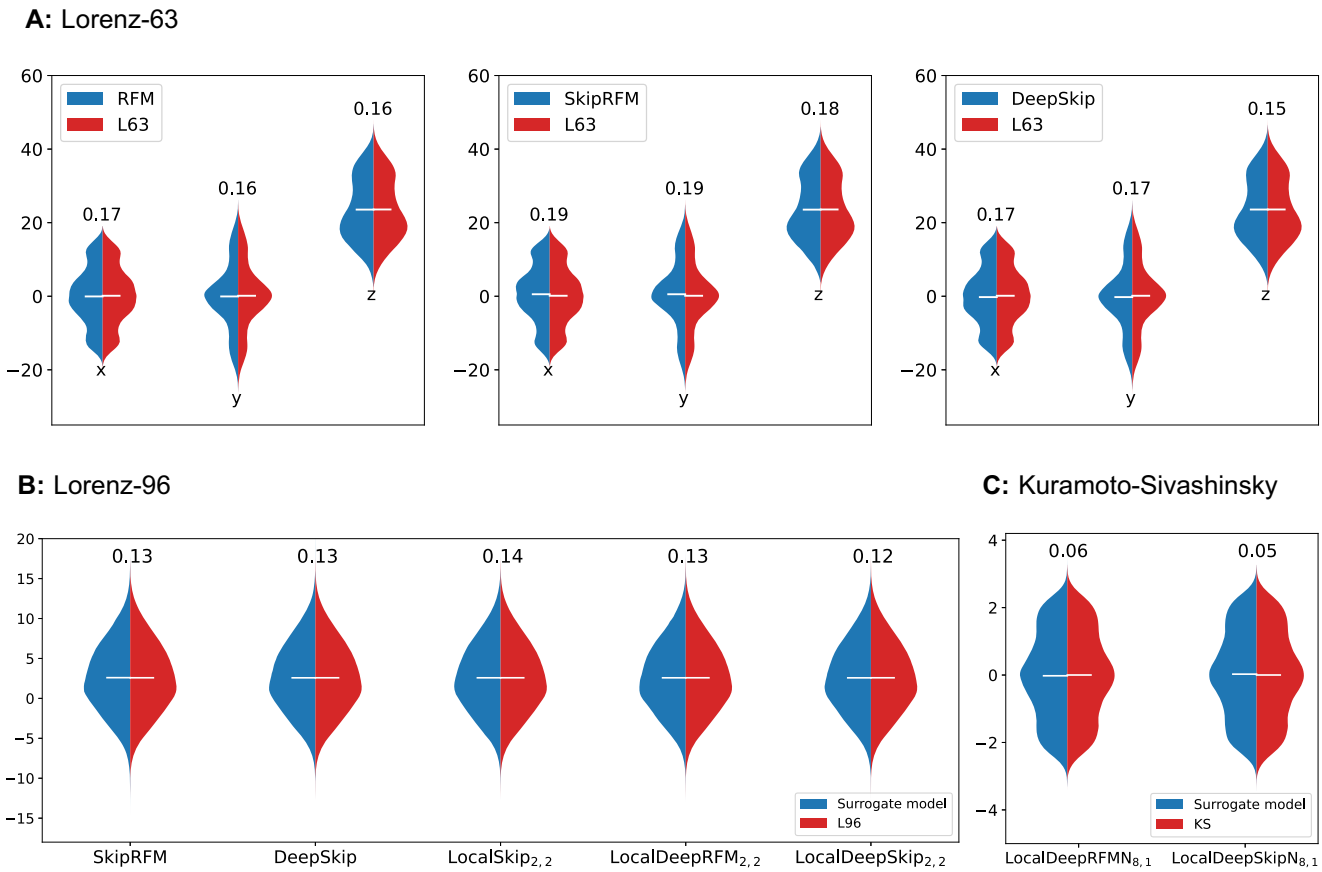

Fig. 5 | Recovering long-time statistics. Comparison of the marginal densities obtained by the original systems and the surrogate models. The respective Wasserstein W_2 distances are indicated at the top of the kernel density plots. **A** L63 system (5) with marginal densities for each component x, y and z. We used the best models marked with the *-symbol in Fig. 2A and in Supplementary Table 1 to generate the data. The mean VPT for these models are 9.8, 10.6 and 12.0 from left to right. **B** L96 system (7). We employ translational symmetry and use all 40 components to estimate the densities. We used the best models marked with the *-symbol in Fig. 2B and in Supplementary Tables 3 and 4 to generate the data. The mean VPT for these models are 2.3, 2.8, 6.8, 7.2, and 7.3 from left to right. **C** KS system (8). We employ translational symmetry and use all 512 components to estimate the densities. We used the best models marked with the *-symbol in Fig. 2C and in Supplementary Table 5 to generate the data. The mean VPT for both of these models is 5.0.

Methods Section. The LocalDeepRFM and LocalDeepSkip models tested on this problem have outer weight matrices with condition numbers ~950 and ~1350, respectively. Due to the larger condition number, LocalDeepSkip performs much worse than LocalDeepRFM, with $\mathbb{E}[\mathrm{VPT}] = 0.5$ for LocalDeepSkip and $\mathbb{E}[\mathrm{VPT}] = 4.8$ for LocalDeepRFM (see Fig. 2C). This is contrary to our observations that skip connections improve performance for the L63 and the L96 system. We remark that LocalSkip models perform equally badly when trained on ill-conditioned data. A possible reason for the larger condition numbers of the $\mathbf{W}$ matrix for skip connections may be the following. For skip connections, $\mathbf{W}$ depends on the matrix of differences $\mathbf{u}_{n+1} - \mathbf{u}_n$ rather than just on $\mathbf{u}_{n+1}$. Hence, its condition number depends on the condition number of this difference matrix. For the KS equation, significant values of the differences $\mathbf{u}_{n+1} - \mathbf{u}_n \in \mathbb{R}^{512}$ appear only in small spatially localized regions with small entries in most components, implying a large condition number.

To mitigate the detrimental effect of large condition numbers, we add zero-mean Gaussian noise with standard deviation 10^{-3} to the training data, as described in the Methods Section. LocalDeepSkip models trained on artificially noised data are able to forecast up to 5 Lyapunov units on average, as shown in Fig. 2C. Models with and without skip connections are seen to perform equally well when the training data are artificially contaminated by small but non-negligible noise.

A comparison of our results with the benchmark results, where the same experimental setup was used[8] (including the addition of noise to the training data), is reported in Table 1C and shows that LocalDeepSkip trained on artificially noised data achieves marginally better results with $\mathbb{E}[\mathrm{VPT}] = 5.0$, but with an approximately 2.7 orders of magnitude smaller model. In Supplementary Table 5 a comparison of our variants of the random feature map is shown for different model sizes, reporting on the mean, median, standard deviation of the VPT as well as the maximal and minimal values.

For reproducing long-term statistics, models trained on noise-free ill-conditioned data are not suitable since they accumulate large errors during long simulations. However, models trained on noisy well-conditioned data are able to reproduce the invariant measure of the KS equation well, as seen in Fig. 5C. The LocalDeepRFM and the LocalDeepSkip architectures with artificially added noise are able to reproduce the long-term statistics of the KS system equally well with comparable Wasserstein distances. We remark that LocalDeepRFM trained on pure data or on noisy data performs approximately equally well on short time scales. However, without the addition of artificial noise to the training data, all surrogate models exhibit numerical instability for long-time forecasting.

Discussion

In this work we extend random feature maps with a tanh-activation function by introducing skip connections, a deep architecture and localization with the aim to produce reliable surrogate models for dynamical systems. We considered a 3-dimensional Lorenz-63 system, a 40-dimensional Lorenz-96 system and a 512-dimensional finite difference discretization of the Kuramoto-Sivashinsky equation, and

studied the ability of the learned surrogate models to forecast individual trajectories as well as the long-time statistical behavior. In all three systems, our modifications lead to either better or equal performance when compared to recent benchmark results using RCs or LSTMs, with orders of magnitude smaller models. For all architectures we judiciously chose the internal weights using a computationally efficient hit-and-run algorithm[20]. This algorithm ensured that for the training data, the features were neither linear nor saturated, and took advantage of the nonlinear nature of the tanh-activation function.

We showed in which situations each of our modifications can be beneficial and that they can significantly improve the forecast capabilities of random feature models. We showed that introducing skip connections typically leads to better performance. However, when the data matrix has too high a condition number, ridge regression leads to an ill-conditioned trained outer weight matrix. This renders the learned surrogate model unstable and unreliable as a forecast model. To combat this, we proposed to add small artificial noise to the data. This allowed for state-of-the-art forecast times for the Kuramoto-Sivashinsky equation with more than an order of magnitude smaller model size when compared against recent benchmark results. It is well known that adding sufficiently strong noise to the training data can severely affect the training of RFMs[18]. We only apply very small noise such that by eye the training data appear unchanged. Although adding noise to the training data is frequently used[8,40], it may be beneficial to add the noise on the features $\mathbf{\Phi}$ instead.

For higher dimensional systems localization was found to be essential. The optimal choice of the localization scheme requires balancing the required accuracy for a given data set of length N, the decay of the spatial correlations of the underlying dynamical system and the available GPU memory.

Our simulations suggest that the performance of random feature models can be significantly improved by considering a deep architecture, chaining RFM units together where each unit is individually trained to match the data. However, the improvement can only be observed once the width of each individual layer is large enough to allow for a sufficiently accurate representation of the dynamics. For instance, for the Lorenz-96 system, we observed that the localized models had not yet plateaued with increasing D_r, and the available GPU memory was fully utilized before reaching saturation.

Our random feature map variants can achieve comparable or even superior performance to RCs while requiring only a fraction of the model size and hence, computational effort. Moreover, although RFMs and RCs share similar learning mechanisms, RFMs offer several advantages over their RC counterparts. One key advantage is that RCs require tuning multiple hyperparameters, such as the spectral radius and density of the reservoir adjacency matrix, degrees of freedom, leak rate, strength of the input signal, strength of the input bias, regularization etc[14], which is computationally expensive. In contrast, RFMs only require optimization of the regularization hyperparameter. Furthermore, RCs are comprised of layers similar to RFMs and a reservoir. These reservoirs are represented by weight matrices of size D_r^2 whereas the weight matrices in RFMs have size DD_r. Since typically, $D_r \gg D$, for the same width, RFMs are significantly lighter models compared to RCs. We remark that implementing sparse matrix and dense vector multiplication on a GPU is not efficient unless the matrix is very sparse. However, the reservoirs employed in the benchmark results reported here are not sparse e.g. Platt et al.[14] report the density of the RC adjacency matrix as being 0.98. To deal with the high memory demands for large RC models a batched approach can be used[8].

Recently, Bayesian methods were proposed to estimate the RFM hyperparameters, including the internal weights[13,25]. It appears that Bayesian hyperparameter tuning has advantages for large sampling times, whereas the hit-and-run sampling algorithm seems to be beneficial for smaller sampling times (cf. Fig. 4B). Combining these two approaches could further improve the forecasting capabilities of RFMs. The Bayesian optimization strategy can also be employed to more efficiently tune hyperparameters such as the localization scheme and the regularization hyperparameter which was determined here using grid-search.

Our skip connection is of the form of a forward-Euler numerical integrator for an underlying continuous time dynamical system. One could aim to learn higher-order multistep integrators such as a Runge-Kutta integrator to improve the accuracy of the prediction[41,42]. However, one needs to be wary of potential "inverse crimes" where the learned map is used for forecasting with a time step different to the sampling time Δt used for training, which may result in numerical instabilities[43]. Embedding the model into a computational graph that is defined by a higher-order numerical integrator, such as a Runge-Kutta method, may be used in future work to better represent the continuous time character of the underlying dynamics, allowing for the application of variable time steps[44].

We considered here noise-free and complete observations for a set of dynamical systems with known equations, allowing for benchmarking. Data from real-world systems typically are noise-contaminated and the system is accessible only via partial observations. It will be interesting to see if the superior forecasting skill in the case of noise-free and complete observations extends to this relevant case. There has been recent progress on learning real-world dynamical systems using Bayesian learning methods with remarkable accuracy such as *eSPA*[45–47] and *BayesNF*[48] which provide benchmarks to test against. The forecasting capability of RFMs quickly deteriorates when observations are contaminated by noise. However, when combining RFMs with data assimilation procedures such as the ensemble Kalman filter, the noise can be successfully controlled for training RFMs[33]. The lack of complete observations renders the dynamical system for the observed states non-Markovian. A Markovian dynamical system can be achieved by formulating the learning task in an enlarged space of time-delay coordinates[49]. This requires determining an appropriate delay embedding[50]. These techniques have been shown to be applicable for learning RFMs from partial observations[18]. Further, to improve the forecasting skill of RFMs in the relevant case of partial noisy observations, it may be beneficial to combine our modifications of RFMs with the hybrid approach promoted by Levine and Stuart[13]. This is planned for further research.

Methods

Initialization of the internal layer

We briefly describe the effective sampling scheme for the internal layer introduced in our previous work[20], which is used throughout this work. Our algorithm is based on the sigmoidal structure of the tanh-activation function. Consider a row of the internal weight matrix $\mathbf{W}_{\text{in}}$ which we denote by $\mathbf{w}_{\text{in}} \in \mathbb{R}^D$, and the corresponding entry of the bias vector which we denote by b_{in}. The domain of the tanh-activation function has three distinct regions: a saturated region, a linear region, and the complement of these two, as illustrated in Fig. 6. Internal weights for which the features $\phi(\mathbf{u}) = \tanh(\mathbf{w}_{\text{in}}\mathbf{u} + b_{\text{in}})$ are saturated i.e. $\phi(\mathbf{u}) \approx \pm 1$ or equivalently $|\mathbf{w}_{\text{in}}\mathbf{u} + b_{\text{in}}| \geq L_1$ (we use $L_1 = 3.5$ throughout) are clearly bad choices as the RFMs would not be able to distinguish between different input signals. Internal weights for which the features lie in the linear region with $|\mathbf{w}_{\text{in}}\mathbf{u} + b_{\text{in}}| \leq L_0$ (we use $L_0 = 0.4$ throughout), lead to a linear model, which is undesirable for learning nonlinear systems. We hence aim to draw internal weights such that the associated features are neither saturated nor linear for any of the training data, these features are labelled as *good* in Fig. 6. The method introduced in our previous work[20] achieves this by a hit-and-run algorithm: starting from a feasible solution $\mathbf{w}_{\text{in}} = 0$ and b_{in} uniformly sampled from the interval $\pm [L_0, L_1]$ we pick random directions in a convex set determined by the training data and the inequalities $L_0 < \mathbf{w}_{\text{in}}\mathbf{u} + b_{\text{in}} < L_1$ or $-L_1 < \mathbf{w}_{\text{in}}\mathbf{u} + b_{\text{in}} < -L_0$. Determining where the line segment defined by

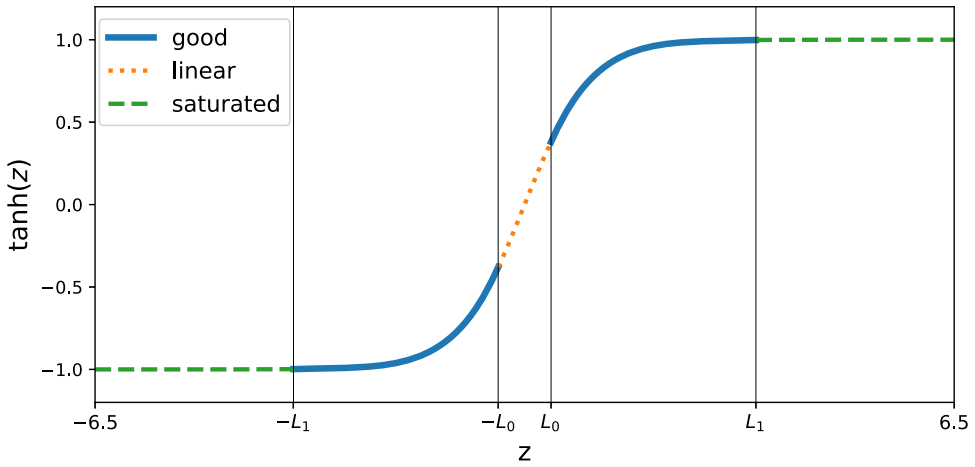

Fig. 6 | Types of features produced by a tanh-activation function. Illustration of the types of features produced by a tanh-activation function, motivating the choice of the internal weights and biases ($\mathbf{W}_{\text{in}}$, $\mathbf{b}_{\text{in}}$). Here and elsewhere $L_0 = 0.4$ and $L_1 = 3.5$.

this direction intersects the convex set allows us to sample weights that map the training data to the aforementioned good features. This process is repeated until D_r independent rows $\mathbf{w}_{\text{in}}$ and biases b_{in} are drawn. We stress that the hit-and-run algorithm does not perform any training by optimization but simply samples the internal weights from a data-informed convex set.

A pseudo-algorithm is provided in the Supplementary Information in Algorithm 1; for a detailed discussion regarding the geometry of the algorithm we refer the reader to our previous work[20]. We emphasize that L_0 and L_1 are treated as constants in our approach. While the selection of their values to delineate good features from bad features could, in principle, be considered hyperparameters requiring tuning, we observe no significant changes in the forecasting capabilities of the learned surrogate maps for values close to $L_0 = 0.4$ and $L_1 = 3.5$. Consequently, we have consistently used $L_0 = 0.4$ and $L_1 = 3.5$ for all the experiments presented in this work.

Skip connections

A simple but effective modification of the random feature map is the introduction of a skip connection from the input to the output[26,27]. In particular, we learn the tendency map $\mathbf{F}_{\Delta t}$: $\mathbf{u}_n \mapsto \mathbf{u}_{n+1} - \mathbf{u}_n$ with an RFM, rather than the propagator map $\mathbf{\Psi}_{\Delta t}$: $\mathbf{u}_n \mapsto \mathbf{u}_{n+1}$. We hence solve the least-square problem (3) where now $\mathbf{U}' \in \mathbb{R}^{D\times N}$ contains the observed tendencies $\{\mathbf{u}_{n+1} - \mathbf{u}_n\}_{n=0}^{N-1}$ across its columns, with solutions given by (4). We will refer to this variant of RFM as SkipRFM.

Bar the constant factor of Δt, SkipRFM can be viewed as learning a single Euler step in a forward-Euler discretization

$$\mathbf{u}_{n+1} = \mathbf{u}_n + \mathbf{F}_{\Delta t}(\mathbf{u}_n) = \mathbf{u}_n + \Delta t\, \tilde{\mathbf{F}}_{\Delta t}(\mathbf{u}_n), \tag{9}$$

for a dynamical system with the vector field

$$\tilde{\mathbf{F}}_{\Delta t}(\mathbf{u}_n) = \frac{1}{\Delta t}\mathbf{W}\tanh(\mathbf{W}_{\text{in}}\mathbf{u}_n + \mathbf{b}_{\text{in}}). \tag{10}$$

Note that the map $\mathbf{F}_{\Delta t}$ is learned from data with a fixed and specified value of the sampling time Δt. When applying the learned map to forecast unseen data, we show results for the same value of Δt we used for learning. We stress that we do not aim to learn the vector field but only the tendency $\mathbf{u}_{n+1} - \mathbf{u}_n$ for fixed sampling time Δt. Choosing different values, which would be possible for actual numerical integrators, may lead to instabilities[13,43], significantly deteriorating the forecast capabilities of the learned model. However, we empirically found our surrogate models to work well when trained on data of finer temporal resolution compared to the testing data for moderate ratios of sampling times of training and testing data.

RFMs with skip connections tend to be marginally better at forecasting than those without skip connections. In fact, in all our test cases, models with skip connections achieved the highest forecast times, as shown in the Results Section.

Deep random feature maps

We now increase the complexity of random feature models by chaining multiple units together to construct deep models and explore some of their benefits. Figure 7A provides an outline of a deep model. We initialize the input with two copies of the state $\mathbf{u}_n$ at time t_n, which are concatenated, to form

$$\mathbf{y}_n^{(0)} = \begin{bmatrix} \mathbf{u}_n \\ \mathbf{u}_n \end{bmatrix}. \tag{11}$$

This augmented state is passed through the first single random feature model unit. The output of the first single unit (and of all following units) replaces one half of the augmented state to form

$$\mathbf{y}_n^{(\ell)} = \begin{bmatrix} \mathbf{W}^{(\ell)}\tanh(\mathbf{W}_{\text{in}}^{(\ell)}\mathbf{y}_n^{(\ell-1)} + \mathbf{b}_{\text{in}}^{(\ell)}) \\ \mathbf{u}_n \end{bmatrix}, \tag{12}$$

and the updated augmented state is again passed through the next unit and so on. Here $\mathbf{W}_{\text{in}}^{(\ell)} \in \mathbb{R}^{D_r\times 2D}$ and $\mathbf{b}_{\text{in}}^{(\ell)} \in \mathbb{R}^{D_r}$ with $\ell = 1, \ldots, B$ denote the inner weights and biases of the ℓ^{th} unit. Similarly, $\mathbf{W}^{(\ell)} \in \mathbb{R}^{D\times D_r}$ denotes the outer weight of the ℓ^{th} unit which are learned sequentially by solving the least-square problem

$$\underset{\mathbf{W}^{(\ell)}}{\arg\min} \parallel \mathbf{W}^{(\ell)}\mathbf{\Phi}(\mathbf{Y}^{(\ell)}) - \mathbf{U}'\parallel_F^2 + \beta \parallel \mathbf{W}^{(\ell)}\parallel_F^2, \tag{13}$$

where $\mathbf{Y}^{(\ell)} \in \mathbb{R}^{2D\times N}$ contains $\{\mathbf{y}_n^{(\ell)}\}_{n=0}^{N-1}$ across its columns. We consider the case when each unit is a standard RFM with $\mathbf{U}' \in \mathbb{R}^{D\times N}$ containing the time-shifted observations $\{\mathbf{u}_{n+1}\}_{n=0}^{N-1}$ across its columns, as well as the case when each unit is a SkipRFM unit with $\mathbf{U}' \in \mathbb{R}^{D\times N}$ containing the tendencies $\{\mathbf{u}_{n+1} - \mathbf{u}_n\}_{n=0}^{N-1}$ across its columns. This process is repeated until we go through the final unit with $\ell = B$ and the final updated upper half of the augmented state is our approximation of the state $\mathbf{u}_{n+1}$ (or $\mathbf{u}_{n+1} - \mathbf{u}_n$ when SkipRFMs are considered) at time t_{n+1}. When the unit is an RFM, the resulting deep model is referred to as DeepRFM. Similarly, when the unit is a SkipRFM, the corresponding deep model is referred to as DeepSkip. We found empirically that using

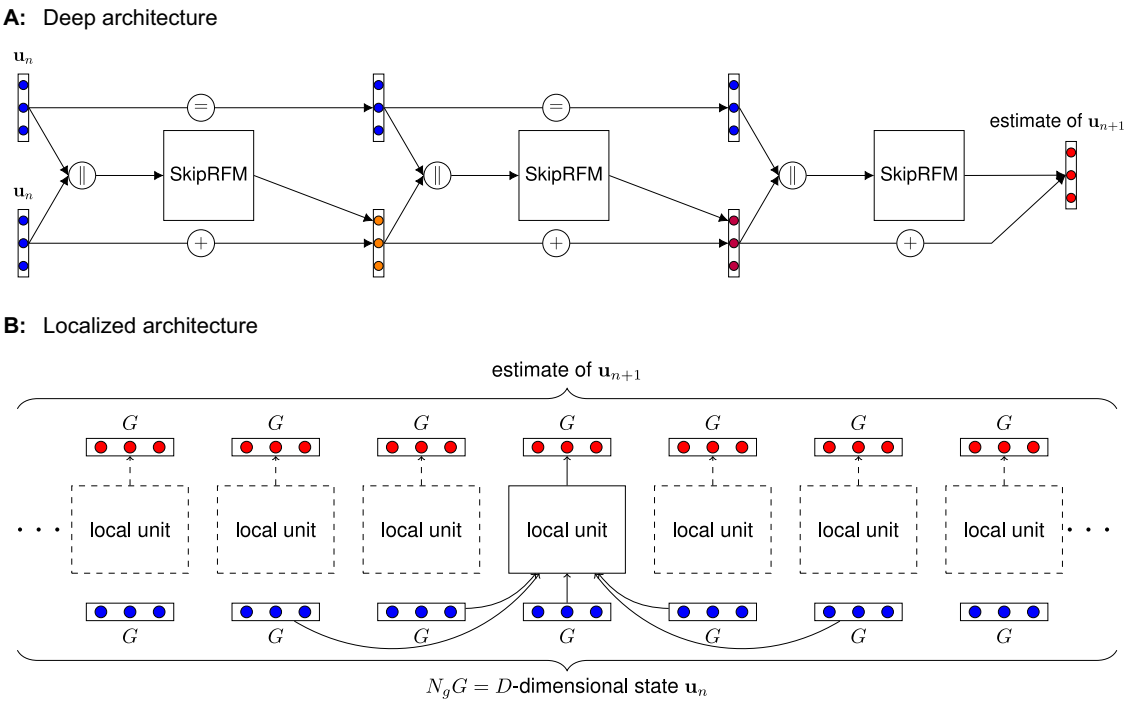

Fig. 7 | Schematics of deep and localized architectures. A Deep architecture DeepSkip with depth $B = 3$. The symbols $\|$, $=$ and $+$ denote concatenation, identity operation and addition (skip connection), respectively. **B** Localized architecture with local state dimension $G = 3$ and interaction length $I = 2$.

augmented states (12), such that each unit has the state $\mathbf{u}_n$ as part of its input, rather than using $\mathbf{y}_n^{(\ell)} = \mathbf{W}^{(\ell)} \tanh(\mathbf{W}_{\text{in}}^{(\ell)} \mathbf{y}_n^{(\ell-1)} + \mathbf{b}_{\text{in}}^{(\ell)})$ with $\mathbf{y}_n^{(0)} = \mathbf{u}_n$, leads to better performing surrogate models. We also found that solving a regression problem at each unit rather than a single regression problem at the last unit i.e. $\mathbf{y}_n^{(\ell)} = \tanh(\mathbf{W}_{\text{in}}^{(\ell)} \mathbf{y}_n^{(\ell-1)} + \mathbf{b}_{\text{in}}^{(\ell)})$, $(l = 1, 2, \cdots, B-1)$ with $\mathbf{y}_n^{(0)} = \mathbf{u}_n$ and $\mathbf{y}_n^{(B)} = \mathbf{W}\mathbf{y}_n^{(B-1)}$, produces better models. The non-trainable internal weights $\mathbf{W}_{\text{in}}^{(\ell)}$ and $\mathbf{b}_{\text{in}}^{(\ell)}$ are determined for all units with the hit-and-run Algorithm 1 described in the Supplementary Information using the same input data $\mathbf{Y}^{(0)}$. It suffices to tune the regularization hyperparameter β in deep RFM architectures for a single unit and reuse it for all the units.

Similar constructions have recently been used in the context of echo state networks[29–31], and universal approximation theorems for a different version of a deep RFM were recently proved[51]. Our deep random feature architecture updates the outer weights sequentially based on the errors incurred at the prior units and is hence reminiscent of stacked boosting[52,53] in machine learning.

Deep versions of random feature models exhibit improved forecasting capabilities when compared to their shallow counterparts, as shown in the Results Section. Moreover, depth has significant computational advantages. A major benefit of introducing depth is that it allows us to train larger models on a GPU with fixed memory. The total number of weights and biases in a model, henceforth referred to as the model size S, significantly influences the model's forecasting skill. But the total memory occupied on the GPU during training of deep RFM models primarily depends on the model width D_r and the size of training data N, and not on the model size. This is because we train the constituent units sequentially, and hence the GPU needs to handle only one linear regression problem at a time. Therefore, a shallow and a deep model with the same width roughly occupy the same amount of GPU memory during training despite the deeper model having a larger size. Furthermore, introducing depth allows for a significant speed-up of training. For a shallow and a deep model of the same size, the deep model necessarily has a smaller width. Therefore, when trained on the same amount of data, the deeper model requires solving regression problems of smaller size. Consequently, among models of the same size, deeper models can train up to an order of magnitude faster, as shown in the Supplementary Information (Section 5).

We remark that the frequency of observations, or the temporal resolution of the data Δt, plays a crucial role in determining the forecasting skill of a trained surrogate model. Generally, smaller values of Δt enable better learning of the underlying dynamical system. In the Results section, we present an example for the Lorenz-63 system where shallow models struggle with large Δt, while deep models demonstrate superior performance.

Localization

To mitigate the curse of dimensionality associated with high-dimensional systems with large D, we design localized variants of random feature models. Typically in high-dimensional systems, for sufficiently small sampling times Δt, the state of a variable at future time t_{n+1} does not depend on all other variables at the current time t_n. An example comes from weather forecasting where the weather at one location typically does not depend on the weather at locations which are several thousand kilometres away. Localization techniques have been successfully employed recently for RCs and LSTMs[8,10,14]. Here we set out to learn N_g localized models by subdividing the state vector into $N_g = D/G$ local states of dimension G each. For each local vector of dimension G we train a local random feature unit. Each local unit takes its own local state along with the states of its neighbours as input, aiming to predict its own local state at the next time step. Concretely, we assume that the state of a local region at time t_{n+1} depends on the state of the same local region as well as on its $2I$ neighbouring local regions at time t_n, where I is called the interaction length. The pair (G, I) defines a localization scheme. Figure 7B illustrates the structure of a localized random feature model. For shallow localized models the input dimension for each unit is $(2I+1)G$. For deep localized models, instead of doubling the input dimension, we augment the input of a local unit with only its own local state giving us an input dimension of $2(I+1)G$. Localized variants of RFM, SkipRFM, DeepRFM and DeepSkip are coined LocalRFM,

LocalSkipRFM, LocalDeepRFM and LocalDeepSkip, respectively. We indicate the localization scheme in the subscript e.g. a LocalDeepSkip model utilizing local state dimension $G = 4$ and interaction length $l = 2$ is referred to as LocalDeepSkip$_{4,2}$. A good localization scheme is crucial for the success of a localized model. Section 3 in the Supplementary Information explores various localization schemes for our test problems and provides some general guidelines for selecting an optimal localization scheme.

Besides controlling the curse of dimensionality, localization also allows for a considerable computational advantage via a tensorized implementation. For dynamical systems with translational symmetry such as the Lorenz-96 system and the Kuramoto-Sivashinsky equation which we consider here, all local units within the complete architecture can be chosen to be identical, allowing us to train a single unit and replicate the trained parameters across the entire model. We exploit this a step further by working with only a single unit that processes information from the entire state using matrix-tensor operations, producing the complete state vector for the next time step. This approach eliminates the need to store multiple local units, reducing the model size by a factor of N_g. The reduction in input dimension allows us to accommodate localized models with much larger width D_r compared to their non-localized counterparts. This, in turn, allows localized models to be significantly more expressive. We will show that the localized models far outperform the non-localized models in the high-dimensional test cases.

Dealing with possibly ill-conditioned data

The data $\mathbf{U}$ may be ill-conditioned, for example, subsequent snapshots of a partial-differential equation may only vary significantly in a small region for a sufficiently small sampling time Δt. For simplicity, let us assume that we are employing an RFM to learn a dynamical system. The outer weight matrix $\mathbf{W}$ depends on the training data $\mathbf{U}$, and as a result, is also ill-conditioned. If the condition number of $\mathbf{W}$ is too large, then the learned surrogate model becomes unstable if run in autonomous mode for the test data. Indeed, during multiple recursive applications of the surrogate model small errors accumulate leading to the predicted state departing from the attractor. The internal parameters ($\mathbf{W}_{\text{in}}$, $\mathbf{b}_{\text{in}}$), sampled by Algorithm 1 (see the Supplementary Information), are then unable to produce good features, and further recursions typically lead to numerical blow-up.

We mitigate such instabilities by artificially adding small noise to the training data. Indeed, adding small noise to an ill-conditioned matrix has been shown rigorously to produce a well-conditioned matrix with high probability[54]. The added artificial noise on the data matrix $\mathbf{U}$ reduces the condition number of the training data and, consequently, that of the outer weight matrix $\mathbf{W}$. The noise should be sufficiently small as not to contaminate the signal and ensure no degradation of the accuracy of the one-step surrogate map. We found that noise, for which the noisy and the original noise-free data are indistinguishable by eye, is sufficient to control instability while still providing accuracy of the learned surrogate model. This strategy is relevant to all the architectures. An example of ill-conditioned training data and its catastrophic effect on the forecasting skill of a LocalDeepSkip model is encountered for the Kuramoto-Sivashinski equation (see Fig. 2C). In the Results Section, we distinguish the models trained on data with added artificial noise by appending 'N' to their name, e.g., a LocalDeepSkip model trained on noisy data is referred to as LocalDeepSkipN. Adding noise to training data has been used routinely and unconditionally for learning deterministic dynamical systems with LSTMs and RCs[8,40]. We only apply noise when dealing with ill-conditioned training data $\mathbf{U}$.

Performance metrics

To evaluate the forecasting skill of our surrogate models, we test them on unseen test data. We initialize the surrogate model with the initial condition of a noise-free test trajectory, and then let the model run in autonomous mode according to

$$\hat{\mathbf{u}}_{n+1} = \boldsymbol{\Psi}_{\Delta t}(\hat{\mathbf{u}}_n) \tag{14}$$

with $\hat{\mathbf{u}}_0 = \mathbf{u}_0$. Note that here $\mathbf{u}$ denotes test data. For simplicity, we label test data the same way as training data when there is no danger for confusion. We compare the surrogate forecasts $\hat{\mathbf{u}}_n$ with the test data $\mathbf{u}_n$. To quantify the forecasting skill we compute the valid prediction time (VPT), measured in Lyapunov time units,

$$\text{VPT} = \frac{1}{T_\Lambda} \sup_n \left\{ n\Delta t : \sqrt{\frac{1}{D}\sum_{i=1}^{D} \left(\frac{\hat{\mathbf{u}}_{n,i} - \mathbf{u}_{n,i}}{\sigma_i} \right)^2} < \varepsilon \right\}, \tag{15}$$

where $T_\Lambda = 1/\Lambda$ is the Lyapunov time with Λ being the maximal Lyapunov exponent. The data mismatch is normalized componentwise by the standard deviation $\sigma \in \mathbb{R}^D$. The standard deviation is numerically estimated from the training data. The parameter $\varepsilon > 0$ is a chosen error threshold. VPT is a diagnostic which has been used for RCs and LSTMs and allows us to compare with several benchmark results from the literature. To obtain meaningful results with a statistical significance we run many realizations where we randomly draw the training data, test data and the internal weights.

We further test the long-term behavior of the surrogate models by running long simulations and comparing their empirical invariant measures with those of the original dynamical system. To quantify the quality of the long-time statistical behavior we estimate the Wasserstein distance W_2 between the 1-dimensional empirical marginal distributions under comparison. We have further estimated the power spectral density of the mean state evolution, another popular probe for long-term statistics. However, we found that the power spectral density is too well recovered by all our RFM variants and hence is not suitable to study their relative performance. We therefore only report on the empirical histograms.

Computational set-up

We ran our experiments on an A100 GPU provided by Google Colab using Python 3.11.12. We used NumPy 2.2.3, SciPy 1.15.2 and PyTorch 2.4.1.

Data availability

The training/testing data and the models shown here can be recreated using the publicly available code referenced in the Code availability statement. The forecast data is publicly available at https://doi.org/10.5281/zenodo.15478037 as well as at the GitHub repository https://github.com/pinakm9/DeepRFM.

Code availability

The code for reproducing the results shown here is openly available on GitHub at https://github.com/pinakm9/DeepRFM[55]. The code is written in Python and utilizes PyTorch for implementation of the random feature models as well as a parallelized version of Algorithm 1 (see the Supplementary Information, Section 1), optimized for GPUs.

References

1. Bi, K. et al. Accurate medium-range global weather forecasting with 3D neural networks. *Nature* **619**, 533–538 (2023).
2. Lam, R. et al. Learning skillful medium-range global weather forecasting. *Science* **382**, 1416–1421 (2023).
3. Price, I. et al. Probabilistic weather forecasting with machine learning.*Nature* **637**, 84–90 (2024).
4. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural Comput.* **9**, 1735–1780 (1997).

5. Maass, W., Natschläger, T. & Markram, H. Real-time computing without stable states: a new framework for neural computation based on perturbations. *Neural Comput.* **14**, 2531–2560 (2002).
6. Jaeger, H. A tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the "echo state network" approach. *GMD-Report 159, German National Research Institute for Computer Science*. (2002).
7. Jaeger, H. & Haas, H. Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. *Science* **304**, 78–80 (2004).
8. 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).
9. Bompas, S., Georgeot, B. & Guéry-Odelin, D. Accuracy of neural networks for the simulation of chaotic dynamics: precision of training data vs precision of the algorithm. *Chaos: Interdiscip. J. Nonlinear Sci.* **30**, 113118 (2020).
10. 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).
11. Rafayelyan, M., Dong, J., Tan, Y., Krzakala, F. & Gigan, S. Large-scale optical reservoir computing for spatiotemporal chaotic systems prediction. *Phys. Rev. X* **10**, 041037 (2020).
12. Nakajima, K. & Fischer, I. *Reservoir Computing* (Springer, 2021).
13. Levine, M. E. & Stuart, A. M. A framework for machine learning of model error in dynamical systems. *Comm. Am. Math. Soc.* **2**, 283–344 (2022).
14. Platt, J. A., Penny, S. G., Smith, T. A., Chen, T.-C. & Abarbanel, H. D. A systematic exploration of reservoir computing for forecasting complex spatiotemporal dynamics. *Neural Netw.* **153**, 530–552 (2022).
15. Gauthier, D. J., Bollt, E., Griffith, A. & Barbosa, W. A. S. Next generation reservoir computing. *Nat. Commun.* **12**, 5564 (2021).
16. 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: Interdiscip. J. Nonlinear Sci.* **31**, 013108 (2021).
17. Rahimi, A. & Recht, B. Random features for large-scale kernel machines. In Platt, J. C., Koller, D., Singer, Y. & Roweis, S. T. (eds.) *Advances in Neural Information Processing Systems 20*, 1177–1184 (Curran Associates, Inc., 2008).
18. Gottwald, G. A. & Reich, S. Combining machine learning and data assimilation to forecast dynamical systems from noisy partial observations. *Chaos: Interdiscip. J. Nonlinear Sci.* **31**, 101103 (2021).
19. Nelsen, N. H. & Stuart, A. M. The random feature model for input-output maps between Banach spaces. *SIAM J. Sci. Comput.* **43**, A3212–A3243 (2021).
20. Mandal, P., Gottwald, G. A. & Cranch, N. On the choice of the non-trainable internal weights in random feature maps for forecasting chaotic dynamical systems. *Foundations of Data Science* https://www.aimsciences.org/article/id/68343b1f4b81a70bbe48bcb3 (2025).
21. Cybenko, G. Approximation by superposition of a sigmoidal function. *Math. Contr. Sign. Syst.* **2**, 303–314 (1989).
22. Park, J. & Sandberg, I. Universal approximation using radial-basis-function networks. *Neural Comput.* **3**, 246–257 (1991).
23. Barron, A. Universal approximation bounds for superposition of a sigmoidal function. *IEEE Trans. Inform. Theory* **39**, 930–945 (1993).
24. Rahimi, A. & Recht, B. Uniform approximation of functions with random bases. In *2008 46th Annual Allerton Conference on Communication, Control, and Computing*. 555–561 (2008).
25. Dunbar, O. R. A., Nelsen, N. H. & Mutic, M. Hyperparameter optimization for randomized algorithms: a case study on random features. *Stat. Comput.* **35**, 56 (2025).
26. He, K., Zhang, X., Ren, S. & Sun, J. Identity mappings in deep residual networks. In Leibe, B., Matas, J., Sebe, N. & Welling, M. (eds.) *Computer Vision – ECCV 2016*. 630–645 (Springer International Publishing, Cham, 2016).
27. Ceni, A. & Gallicchio, C. Residual echo state networks: residual recurrent neural networks with stable dynamics and fast learning. *Neurocomputing* **597**, 127966 (2024).
28. E, W. A proposal on machine learning via dynamical systems. *Commun. Math. Stat.* **5**, 1–11 (2017).
29. Akiyama, T. & Tanaka, G. Computational efficiency of multi-step learning echo state networks for nonlinear time series prediction. *IEEE Access* **10**, 28535–28544 (2022).
30. Ding, S., Zhang, N., Xu, X., Guo, L. & Zhang, J. Deep extreme learning machine and its application in EEG classification. *Math. Probl. Eng.* **2015**, 129021 (2015).
31. Uzair, M., Shafait, F., Ghanem, B. & Mian, A. Representation learning with deep extreme learning machines for efficient image set classification. *Neural Comput. Appl.* **30**, 1211–1223 (2018).
32. Gottwald, G. & Reich, S. Localized Schrödinger bridge sampler. preprint at *arXiv* https://doi.org/10.48550/arXiv.2409.07968 (2024).
33. Gottwald, G. A. & Reich, S. Supervised learning from noisy observations: Combining machine-learning techniques with data assimilation. *Phys. D: Nonlinear Phenom.* **423**, 132911 (2021).
34. Feydy, J. et al. Interpolating between optimal transport and mmd using Sinkhorn divergences. In *The 22nd International Conference on Artificial Intelligence and Statistics*, 2681–2690 (PMLR, 2019).
35. Lorenz, E. N. Deterministic nonperiodic flow. *J. Atmos. Sci.* **20**, 130–141 (1963).
36. Lorenz, E. N. Predictability: a problem partly solved. In *Proc. Seminar on predictability*, 1, (Reading, 1996).
37. Kuramoto, Y. Diffusion-induced chaos in reaction systems. *Prog. Theor. Phys. Suppl.* **64**, 346–367 (1978).
38. Sivashinsky, G. Nonlinear analysis of hydrodynamic instability in laminar flames–i. Derivation of basic equations. In *Dynamics of Curved Fronts*, 459–488 (Elsevier, 1988).
39. Kassam, A.-K. & Trefethen, L. N. Fourth-order time-stepping for stiff PDEs. *SIAM J. Sci. Comput.* **26**, 1214–1233 (2005).
40. Vlachas, P. R., Byeon, W., Wan, Z. Y., Sapsis, T. P. & Koumoutsakos, P. Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks. *Proc. R. Soc. A: Math. Phys. Eng. Sci.* **474**, 20170844 (2018).
41. Keller, R. T. & Du, Q. Discovery of dynamics using linear multistep methods. *SIAM J. Numer. Anal.* **59**, 429–455 (2021).
42. Du, Q., Gu, Y., Yang, H. & Zhou, C. The discovery of dynamics via linear multistep methods and deep learning: error estimation. *SIAM J. Numer. Anal.* **60**, 2014–2045 (2022).
43. Krishnapriyan, A. S., Queiruga, A. F., Erichson, N. B. & Mahoney, M. W. Learning continuous models for continuous physics. *Commun. Phys.* **6**, 319 (2023).
44. Queiruga, A. F., Erichson, N. B., Taylor, D. & Mahoney, M. W. Continuous-in-depth neural networks. preprint at *arXiv* https://doi.org/10.48550/arXiv.2008.02389 (2020).
45. Horenko, I. Cheap robust learning of data anomalies with analytically solvable entropic outlier sparsification. *Proc. Natl Acad. Sci. USA* **119**, e2119659119 (2022).
46. Horenko, I. et al. On cheap entropy-sparsified regression learning. *Proc. Natl Acad. Sci. USA* **120**, e2214972120 (2023).
47. Groom, M., Bassetti, D., Horenko, I. & O'Kane, T. J. On the comparative utility of entropic learning versus deep learning for long-range ENSO prediction. *Artif. Intell. Earth Syst.* **3**, 240009 (2024).

48. Saad, F. et al. Scalable spatiotemporal prediction with Bayesian neural fields. Nature Communications https://doi.org/10.1038/s41467-024-51477-5 (2024).
49. Takens, F. Detecting strange attractors in turbulence. In *Dynamical systems and turbulence, Warwick 1980 (Coventry 1979/1980)*, vol. 898 of *Lecture Notes in Math.*, 366–381 (Springer, Berlin, 1981).
50. Kantz, H. & Schreiber, T. *Nonlinear Time Series Analysis*. (Cambridge University Press, Cambridge, 1997).
51. Bosch, D., Panahi, A. & Hassibi, B. Precise asymptotic analysis of deep random feature models. In Neu, G. & Rosasco, L. (eds.) *Proceedings of Thirty Sixth Conference on Learning Theory*, vol. 195 of *Proceedings of Machine Learning Research*, 4132–4179 (PMLR, 2023).
52. Schapire, R. E. & Freund, Y. *Boosting: Foundations and Algorithms*. https://doi.org/10.7551/mitpress/8291.001.0001 (The MIT Press, 2012).
53. Kim, I.-C. & Myoung, S.-H. Text categorization using hybrid multiple model schemes. In R. Berthold, M., Lenz, H.-J., Bradley, E., Kruse, R. & Borgelt, C. (eds.) *Advances in Intelligent Data Analysis V*, 88–99 (Springer Berlin Heidelberg, Berlin, Heidelberg, 2003).
54. Spielman, D. A. & Teng, S.-H. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *J. ACM* **51**, 385–463 (2004).
55. Mandal, P. Learning dynamical systems with hit-and-run random feature maps. Dataset on Zenodo https://doi.org/10.5281/zenodo.15478037 (2025).
56. Köster, F., Patel, D,., Wikner, A. & Jaurigue, L. Jaurigue, L. & Lüdge, K. Data-informed reservoir computing for efficient time-series prediction.*Chaos: Interdiscip. J. Nonlinear Sci.* **33**, 073109 (2023).
57. Penny, S. G. et al. Integrating recurrent neural networks with data assimilation for scalable data-driven state estimation. *J. Adv. Modeling Earth Syst.* **14**, e2021MS002843 (2022).

Acknowledgements

GAG and PM acknowledge support from the Australian Research Council under Grant No. DP220100931.

Author contributions

PM implemented the random feature models and ran the simulations. PM and GAG equally contributed in conceptualizing the methodology and writing 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/s41467-025-61195-1.

Correspondence and requests for materials should be addressed to Pinak Mandal or Georg A. Gottwald.

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.