# LEARNING DYNAMICAL SYSTEMS WITH HIT-AND-RUN RANDOM FEATURE MAPS

Pinak Mandal,[*] Georg A. Gottwald, [†]

University of Sydney, NSW 2006, Australia

ABSTRACT. 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 forecast skill with much smaller networks.

## 1. INTRODUCTION

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, 2, 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 Remarkably, more complex architectures such as Long Short-Term Memory (LSTM) architectures [4] have been replaced by much simpler architectures such as reservoir computers (RC) or Echo-State Networks (ESN), exhibiting better forecasting capabilities with forecasting times exceeding many Lyapunov times [5, 6, 7, 8, 9, 10, 11, 12]. Indeed, reservoir computing has emerged as the prominent architecture for modeling and predicting the behavior of chaotic dynamical systems. 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 [13, 14].

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) [15] 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 [16, 17, 18, 19]. RFMs enjoy the universal approximation property, and can, in principle, approximate any continuous function arbitrarily well [20, 21, 22, 23]. 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 [24, 19]. We follow here our strategy developed in [19] designed for tanh-activation functions, and employ a hit-and-run algorithm to initialize the non-trainable internal parameters ensuring that for the

---

[*]pinak.mandal@sydney.edu.au

[†]georg.gottwald@sydney.edu.au

given training data the weights do not project the data into 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 the vector field instead. This is similar to skip connections in residual networks [25] and has recently been used in RCs [26]. 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 [27]. A similar construction of multi-step learning has been applied to ESNs for forecasting [28] and classification problems [29, 30]. 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. We employ here 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, 9, 12, 31].

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).

The paper is organized as follows. In Section 2, we describe the RFM framework, its deep and local extensions along with the performance metrics used to evaluate our surrogate models. In Section 3 we show that RFMs are capable of producing accurate forecasts for single trajectories as well as accurate estimates of the long-time statistical properties of the underlying dynamical systems. We provide a comparison with benchmark results from recent literature. Finally, in Section 4, we close with a brief summary and discussion.

## 2. Methodology

We consider a $D$-dimensional dynamical system which is observed at discrete times $t_n = n\Delta t$ with constant sampling time $\Delta t$. 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 $\Psi_{\Delta t}$ that approximates the map $\mathbf{u}_n \mapsto \mathbf{u}_{n+1}$ 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.

2.1. **Classical random feature maps.** Random feature maps are feedforward neural networks consisting of an internal layer of width $D_r$ and an external layer. As is customary, we use tanh as the activation 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

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

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

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

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

$$\underset{\mathbf{W}}{\arg\min} \|\mathbf{W}\mathbf{\Phi}(\mathbf{U}) - \mathbf{U}'\|_F^2 + \beta\|\mathbf{W}\|_F^2, \tag{3}$$

2

where $\mathbf{U} \in \mathbb{R}^{D \times N}$ contains the observations $\{\mathbf{u}_i\}_{i=0}^{N-1}$ across its columns and $\mathbf{U}' \in \mathbb{R}^{D \times N}$ contains the time-shifted observations $\{\mathbf{u}_i\}_{i=1}^{N}$ 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 $\beta$ is a hyperparameter which requires tuning. Here $\|\cdot\|_F$ denotes the Frobenius norm. The solution to the ridge regression minimization 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 omitted the dependency of the feature matrix $\mathbf{\Phi}$ on the data $\mathbf{U}$; in particular, no costly back-propagation is required. The quality of the learned surrogate model sensitively depends on the random initialization of the internal layer and the hyperparameter $\beta$.

2.2. **Initialization of the internal layer.** We briefly describe the effective sampling scheme for the internal layer introduced in our previous work [19], which is used throughout this work. Our algorithm is based on the specific functional form of the tanh-activation function. Consider a row of the internal weight matrix $\mathbf{W}_{\text{in}}$ which we denote by $\mathbf{w}_{\text{in}} \in \mathbf{R}^D$, and an entry of the bias vector which we denote by $b_{\text{in}}$. The domain of the tanh-activation function have three distinct regions: a saturated region, a linear region and the complement of these two, as illustrated in Figure 1. 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 are not 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), RFMs become 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. The method proposed in [19] achieves this by a hit-and-run algorithm: starting from a feasible solution $\mathbf{w}_{\text{in}} = 0$ and $b_{\text{in}}$ uniformly distributed in the interval $\pm[L_0, L_1]$ we pick random directions in the weight-bias space $\mathbb{R}^{D+1}$. Determining where the line segment defined by the direction intersects the good region allows us to uniformly sample from that line segment. The new sample can then be used as the starting point for another search in a new random direction in $\mathbb{R}^{D+1}$. This process is repeated until $D_r$ independent rows $\mathbf{w}_{\text{in}}$ and biases $b_{\text{in}}$ are drawn. It turns out that this can be formulated as sampling from a convex set.

The method is summarized in Appendix 6.1 in Algorithm 1; for a detailed discussion regarding the geometry of the algorithm we refer to [19]. 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.
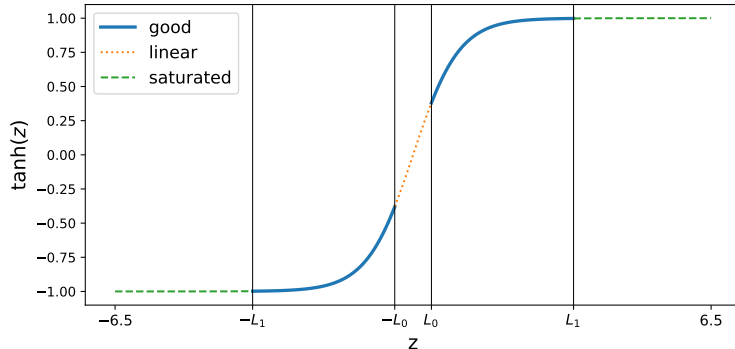


FIGURE 1. Illustration of the types of features produced by a tanh-activation function, motivating the choice of the internal weights and biases $(\mathbf{W}_{\text{in}}, \mathbf{b}_{\text{in}})$. Here and elsewhere $L_0 = 0.4$ and $L_1 = 3.5$.

2.3. **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 [25, 26]. This is akin to learning the map $\mathbf{u}_n \mapsto \mathbf{u}_{n+1} - \mathbf{u}_n$ with an RFM, rather than the propagator map $\mathbf{u}_n \mapsto \mathbf{u}_{n+1}$. We will refer to this variant of the RFM as the SkipRFM. Forecasting with SkipRFM is tantamount to integrating in time, using a forward Euler method

$$\mathbf{u}_{n+1} = \mathbf{u}_n + \Delta t \, \mathbf{F}(\mathbf{u}_n), \tag{5}$$

for an underlying dynamical system with the vetcor field

$$\mathbf{F}(\mathbf{u}_n) = \frac{1}{\Delta t} \mathbf{W} \tanh\left(\mathbf{W}_{\text{in}} \mathbf{u}_n + \mathbf{b}_{\text{in}}\right). \tag{6}$$

RFMs with skip connections tend to be marginally better at forecasting than those without. In fact, in the all our test cases, models with skip connections achieved the highest forecast times, as we will see in Section 3.

2.4. **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 2 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. This augmented state is passed through a single random feature model unit. The output of the single unit is used to replace one half of the augmented state and the updated augmented state is again passed through another unit. This process is repeated until we go through the final unit and the final updated half of the augmented state is our approximation of the state $\mathbf{u}_{n+1}$ 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 called DeepSkip. We train each unit sequentially by solving a linear regression problem, with the goal of predicting the final result at each intermediate step. This type of construction has recently been used in the context of echo state networks [29, 30, 28]. The number of units in a deep model is referred to as its depth, which we denote by $B$.

Deep versions of random feature models exhibit improved forecasting capabilities when compared to their shallow counterparts, as will be shown in Section 3. 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 we will see in Section 6.4.

We remark that the frequency of observations, or the temporal resolution of the data $\Delta t$, plays a crucial role in determining the forecast skill of a trained surrogate model. Generally, smaller values of $\Delta t$ enable better learning of the underlying dynamical system. In Section 3.1, we present an example where shallow models struggle with large $\Delta t$, while deep models demonstrate superior performance.

2.5. **Localization.** To mitigate the curse 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 $\Delta t$, 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 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, 9, 12]. 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
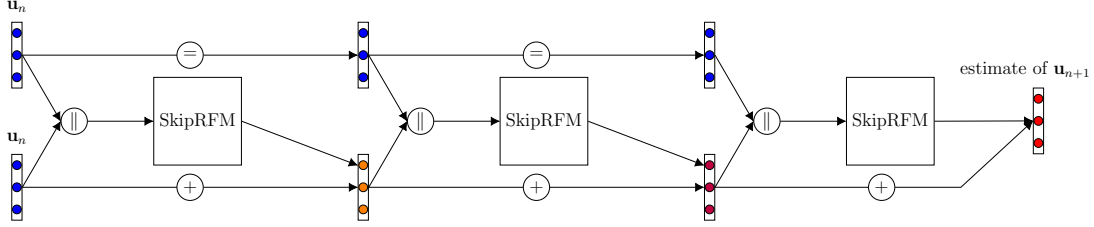
FIGURE 2. Schematic of the deep architecture DeepSkip with depth $B = 3$. The symbols $\parallel$, $=$ and $+$ denote concatenation, identity operation and addition (skip connection), respectively.
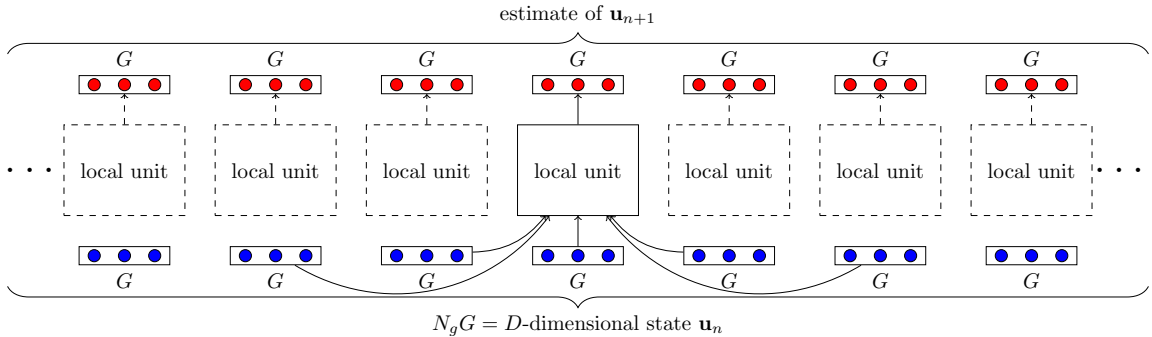


FIGURE 3. Schematic of a localized architecture. In this example, the local state dimension is $G = 3$ and the interaction length is $I = 2$.

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 3 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 $I = 2$ is referred to as LocalDeepSkip$_{4,2}$. A good localization scheme is crucial for the success of a localized model. Appendix 6.3 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 we consider in Section 3, 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 $N_g$. This 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. In Section 3, we see that the localized models far outperform the non-localized models in the high-dimensional test cases.

2.6. **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 $\Delta t$. 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 Appendix 6.1), 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 [32]. 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 covered in this section. In Section 3.3 we show an example of ill-conditioned training data and its catastrophic effect on the forecast skill of a LocalDeepSkip model. In the following, 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.

2.7. **Performance metrics.** To evaluate the forecast 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

$$\hat{\mathbf{u}}_{n+1} = \Psi_{\Delta t}(\hat{\mathbf{u}}_n) \tag{7}$$

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 forecast skill we compute the valid prediction time (VPT), measured in Lyapunov times,

$$\text{VPT} = \frac{1}{T_\Lambda} \sup_n \left\{ n\Delta t : \sqrt{\frac{1}{D} \sum_{i=1}^{D} \left( \frac{\hat{\mathbf{u}}_{n,i} - \mathbf{u}_{n,i}}{\sigma_i} \right)^2} < \varepsilon \right\}, \tag{8}$$

where $T_\Lambda = \frac{1}{\Lambda}$ is the Lyapunov time with $\Lambda$ 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 behaviour 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 behaviour 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 study their relative performance. We therefore only report on the empirical histograms.

2.8. **Data and code.** The code for reproducing the results shown here and the forecast data are openly available on Github at `https://github.com/pinakm9/DeepRFM`. The code is written in Python and heavily utilizes PyTorch for implementation of the random feature models as well as a parallelized version of Algorithm 1 (see Appendix 6.1).

## 3. RESULTS

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 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 and biases of the surrogate model. For each model, the regularization hyperparameter $\beta$ was optimized via grid search.

To compare empirical histograms obtained from long-time simulations, Wasserstein distances $W_2$ are estimated from $3 \times 10^4$ random samples for each model using the Sinkhorn algorithm with an entropy regularization parameter of $10^{-2}$ [33].

All experiments were done on the A100 GPU provided by Google Colab. Additional numerical details regarding the results shown here can be found in Appendix 6.2.

3.1. **Lorenz-63.** In this section we demonstrate the forecast skill and long-term behavior of surrogate models for the Lorenz-63 (L63) system with standard parameters [34],

$$
\begin{aligned}
\frac{dx}{dt} &= 10(y - x), \\
\frac{dy}{dt} &= x(28 - z) - y, \\
\frac{dz}{dt} &= xy - \frac{8}{3}z.
\end{aligned}
\tag{9}
$$

The maximal Lyapunov exponent is estimated to be $\Lambda = 0.91$ [12]. Localization is not required for this low-dimensional system, and we consider here the non-localized versions RFM, SkipRFM and DeepSkip.

Figure 4 shows a sample forecast of a DeepSkip model which is accurate up to approximately VPT $\approx 19$ 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 5 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 $\varepsilon = 0.3$. We show results for RFM, SkipRFM and DeepSkip. 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 $\mathbb{E}[\text{VPT}] \approx 10.5$ 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 $D_r = 1,024$ and depth $B = 32$ with $\mathbb{E}[\text{VPT}] = 12$. Deep models improve with depth even when the model size $S = (3D + 1)D_r B$ is kept fixed, as seen in Figure 6 for two different model sizes. Since depth allows us to train larger models as discussed in Section 2.4, we are able to train deep models that are 3 times larger than the largest shallow model increasing the expressivity of the model. In Appendix 6.4 we show that deep architectures allow for an order of magnitude faster training. In Appendix 6.2 a comparison of our variants of the random feature model is shown in Tables 4 and 5 for different model sizes, reporting on the mean, median, standard deviation of the VPT as well as the maximal and minimal values.

Table 1 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.

| Source | Model | $\log_{10}$(model size) | $\mathbb{E}[\text{VPT}]$ | $N$ | $\Delta t$ | $\varepsilon$ |
|---|---|---|---|---|---|---|
| Akiyama et al. (2022) [28] | Multi-step ESN | 5.05 | 9.3 | $2 \times 10^4$ | 0.02 | 0.4 |
| Platt et al. (2022) [12] | RC | 6.60 | $11.8 - 12.0$ | $5 \times 10^4$ | 0.01 | 0.3 |
| Koster et al. (2023) [35] | DI-RC (SINDy) | 6.00 | 4.0 | $10^4$ | 0.01 | $\sqrt{0.4}$ |
| Our work | DeepSkip RFM | 5.52 | 12.0 | $5 \times 10^4$ | 0.01 | 0.3 |
| Our work | DeepSkip RFM | 5.52 | 11.8 | $2 \times 10^4$ | 0.02 | $\sqrt{0.05}$ |

TABLE 1. Comparison of mean VPT and corresponding model sizes from recent benchmark results for forecasting the L63 system (9). The result corresponding to the best mean VPT $\mathbb{E}[\text{VPT}]$ is reported for each source. The largest $\mathbb{E}[\text{VPT}]$ and the corresponding smallest model size are highlighted by red shading.
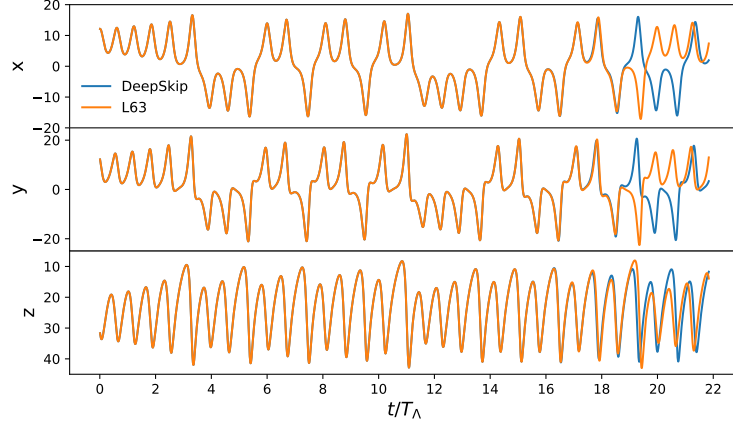
FIGURE 4. An example of a forecast by a DeepSkip model with width $D_r = 1,024$ and depth $B = 16$ for the L63 system (9). The surrogate model is able to forecast accurately up to VPT $\approx 19$ Lyapunov time units.



FIGURE 5. Kernel density plots of VPT for the L63 system (9) for $(N, \Delta t, \varepsilon) = (5 \times 10^4, 0.01, 0.3)$. For RFM and SkipRFM, $\log_2(D_r)$ is indicated on the top of the plots. For DeepSkip, $(\log_2(D_r), B)$ are indicated on the top of each plot. The $*$-symbol indicates the model with the best mean VPT within each architecture.
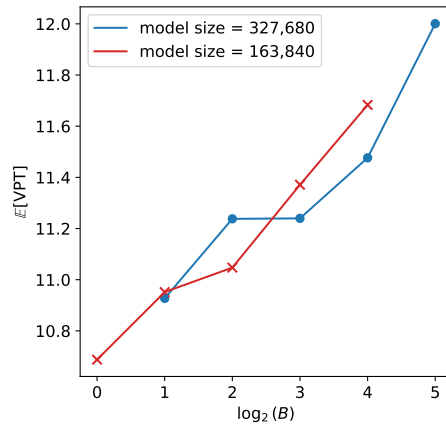


FIGURE 6. Mean VPT for DeepSkip as a function of depth $B$ for constant model size $S$ for the L63 system (9). Along each curve the model size $S = (3D + 1)D_r B$ remains constant and the width $D_r$ decreases with depth.

8

In general, finer temporal resolution is beneficial for learning the dynamics. In Figure 7 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 forecast skill for various models of nearly similar size. The deep model outperforms the shallow models by $\sim 4.8$ 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 $\Delta t$ is changed from 0.01 to 0.1. Smaller sampling times $\Delta t$ allow for a better approximation of temporal derivatives and therefore SkipRFM outperforms RFM for small $\Delta t$. This advantage, however, vanishes at higher $\Delta t$ and both perform equally. For RFM, SkipRFM, and DeepSkip, the mean VPT drops by 49.5%, 53.8% and 15.8%, respectively, indicating that the deep architectures are the least susceptible to the temporal resolution of the training data.
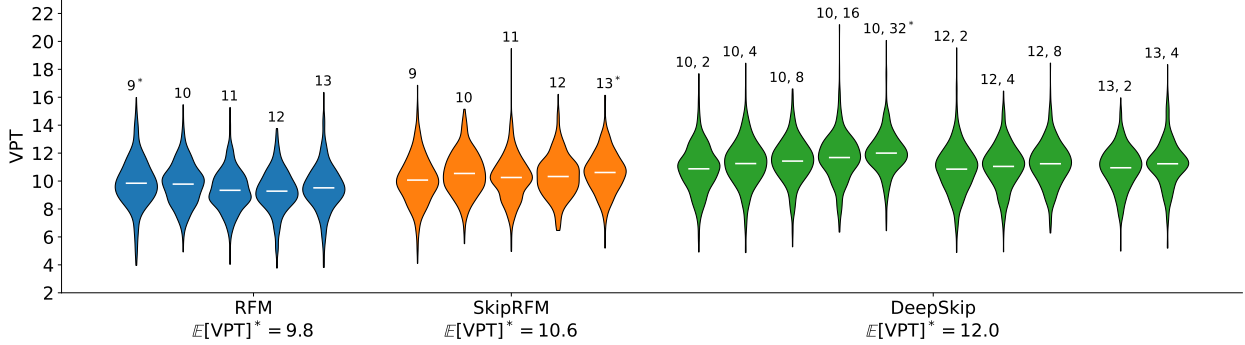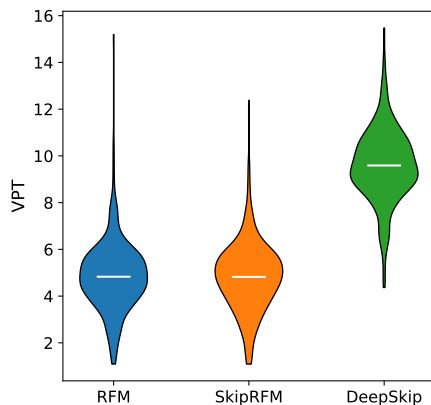


FIGURE 7. Kernel density plots of VPT for the L63 system (9) for $(N, \Delta t, \varepsilon) = (5 \times 10^4, 0.1, 0.3)$. Sizes of the models are $S = 114,688$, $S = 114,688$ and $S = 114,560$, from left to right, with, $D_r = 16,384$, $D_r = 16,384$ and $D_r = 716$ respectively. The DeepSkip model has depth $B = 16$.

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 8 compares the marginal densities estimated from the invariant measures of the original L63-system (9) 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.
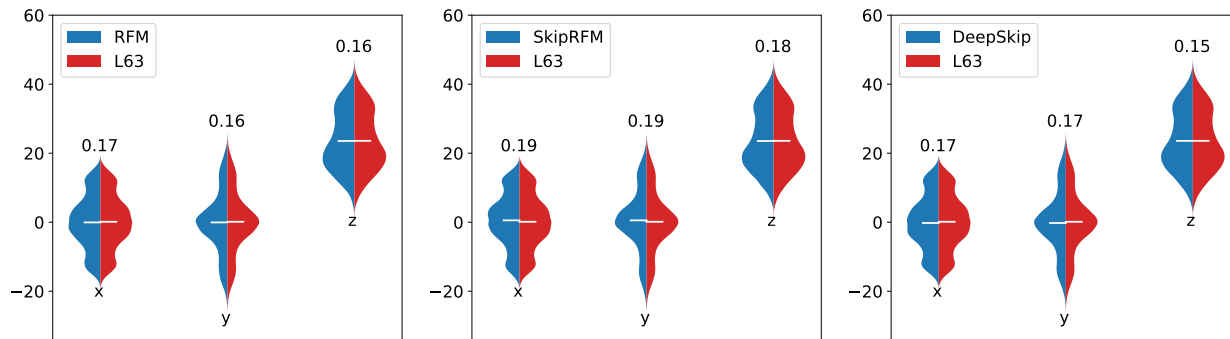


FIGURE 8. Marginal densities for $x$, $y$ and $z$ estimated from the invariant measures of the L63 system (9) and the surrogate models RFM, SkipRFM and DeepSkip. The respective Wasserstein $W_2$ distances are indicated at the top of the kernel density plots. We used the best models marked with the $*$-symbol in Figure 5 to generate the data.

3.2. **Lorenz-96.** In this section we demonstrate the forecast skill for the 40-dimensional Lorenz-96 (L96) system

$$\frac{dx_i}{dt} = (x_{i+1} - x_{i-2})x_{i-1} - x_i + F, \ \ i = 1, 2, \cdots, D, \tag{10}$$

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 $\Lambda = 2.27$ [9]. 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.

Figure 9 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 Appendix 6.3 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 plateau quickly with increasing model size whereas we run out of GPU memory before observing saturation in the forecast 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 Figure 10 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 $B$ implies a decrease of the width $B$, the deeper networks are not able to resolve the dynamics to sufficient accuracy at each layer.

Table 2 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 the Appendix 6.2 a comparison of our variants of the random feature model is shown in Tables 6 and 7 for different model sizes, reporting on the mean, median, standard deviation of the VPT as well as the maximal and minimal values.
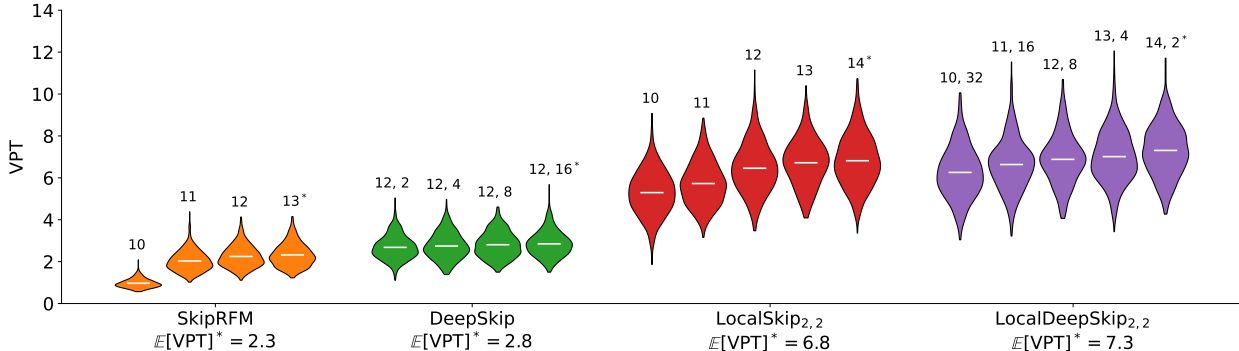


FIGURE 9. Kernel density plots of VPT for the L96 system (10) for $(N, \Delta t, \varepsilon) = (10^5, 0.01, 0.5)$. For shallow variants $\log_2(D_r)$ is indicated on the top of the plots. For deep variants $(\log_2(D_r), B)$ are indicated on the top of each plot. The $*$-symbol indicates the model with the best mean VPT within each architecture.

Figure 11 compares the empirical marginal densities estimated from the invariant measures of the original L96-system (10) 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.
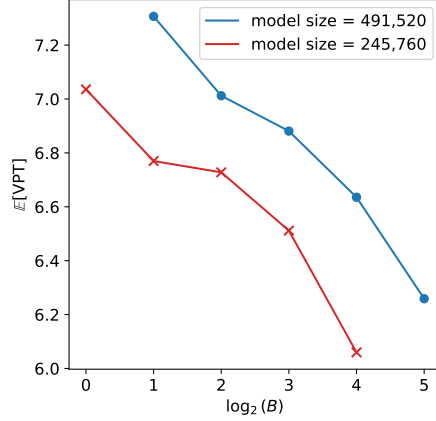
FIGURE 10. Mean VPT for LocalDeepSkip as a function of depth $B$ for constant model size $S$ and localization scheme $(G, I) = (2, 2)$. Along each curve the model size $S = (\hat{D} + G + 1)D_r B$ with $\hat{D} = 2G(I + 1)$ remains constant and the width $D_r$ decreases with depth.

| Source | Model | $\log_{10}$(model size) | $\mathbb{E}[\text{VPT}]$ | $N$ | $\Delta t$ | $\varepsilon$ |
|---|---|---|---|---|---|---|
| Penny et al. (2022) [37, 12][3]. | RC | 7.56 | $2.5 - 2.8$ | $2 \times 10^5$ | 0.01 | 0.5 |
| Vlachas et al. (2022) [9] | Localized LSTM | 5.95 | 3.9 | $10^5$ | 0.01 | 0.5 |
| Platt et al. (2022) [12] | Localized RC | 7.03 | $6.5 - 6.8$ | $4 \times 10^4$ | 0.01 | 0.5 |
| Our work | LocalDeepSkip RFM | 5.69 | 7.3 | $10^5$ | 0.01 | 0.5 |

TABLE 2. Comparison of mean VPT and corresponding model sizes from recent benchmark results for forecasting the L96 system (10). The result corresponding to the best mean VPT $\mathbb{E}[\text{VPT}]$ is reported for each source. The largest $\mathbb{E}[\text{VPT}]$ and the corresponding smallest model size are highlighted by red shading.
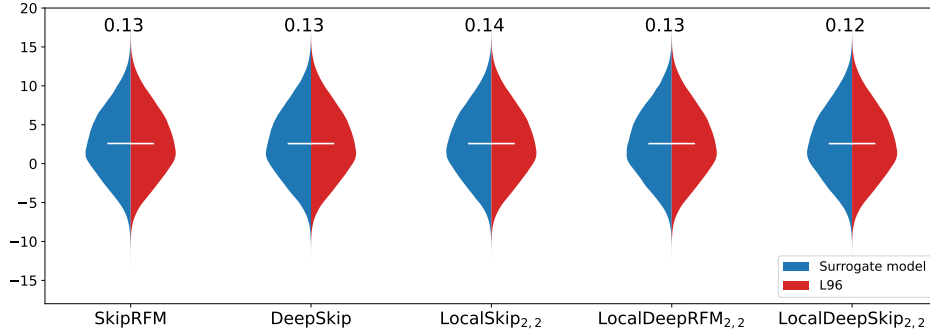


FIGURE 11. Marginal densities for a component of the L96 system, estimated from the invariant measures of the original L96 system (10) and various surrogate models. We employ translational symmetry and use all 40 components to estimate the densities. The respective Wasserstein $W_2$ distances are indicated at the top of the kernel density plots. We used the best models marked with the $*$-symbol in Figure 9 to generate the data.

3.3. **Kuramoto-Sivashinsky.** We further consider the Kuramoto-Sivashinsky (KS) equation

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + \frac{\partial^2 u}{\partial x^2} + \frac{\partial^4 u}{\partial x^4} = 0 \tag{11}$$
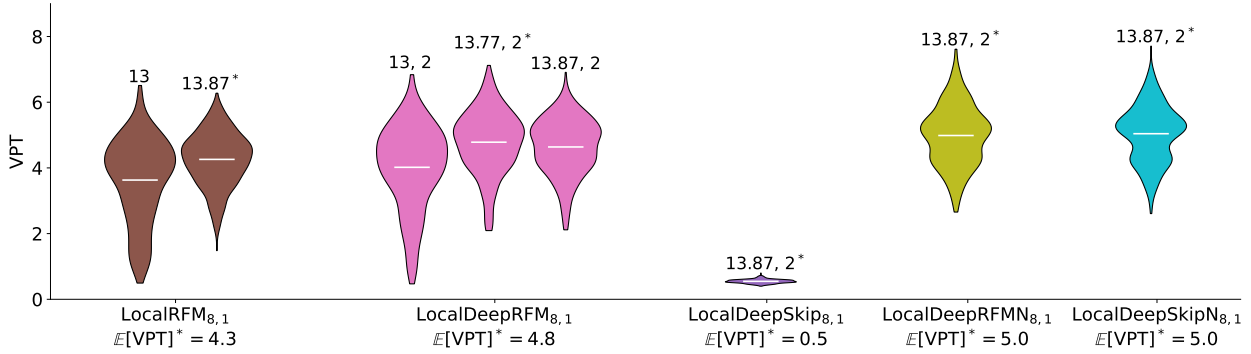
FIGURE 12. Kernel density plots of VPT for the KS system (11) for $(N, \Delta t, \varepsilon) = (10^5, 0.25, 0.5)$ for various localized surrogate models. For all models $(\log_2(D_r), B)$ are indicated on the top of each plot. The $*$-symbol indicates the model with the best mean VPT within each architecture. The maximal value of $D_r$ allowed by our GPU is $15,000$ (i.e. $\log_2(D_r) \approx 13.87$).

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 [38, 39]. The maximal Lyapunov exponent is estimated to be $\Lambda = 0.094$ [9]. We solve equation 11 on a domain of length $L = 200$ with a uniform grid of 512 nodes using the ETDRK4 method [40] with a time step of $h = 0.001$. 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. 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 here a localization scheme of $(G, I) = (8, 1)$; see Appendix 6.3 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 $\sim 10^{15}$. As discussed in Section 2.6, 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. The LocalDeepRFM and LocalDeepSkip models tested on this problem have outer weight matrices with condition numbers $\sim 950$ and $\sim 1,350$ respectively. Due to the larger condition number, LocalDeepSkip performs much worse than LocalDeepRFM, with $\mathbb{E}[VPT] = 0.5$ for LocalDeepSkip and $\mathbb{E}[VPT] = 4.8$ for LocalDeepRFM (cf. Figure 12). 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 high condition numbers of the $\mathbf{W}$ matrix for skip connections may be the following. Note that 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 against the detrimental effect of large condition numbers, we add zero-mean Gaussian noise with standard deviation $10^{-3}$ to the training data. LocalDeepSkip models trained on artificially noisy data are able forecast up to 5 Lyapunov units on average, as shown in Figure 12. 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 of [9], where the same experimental setup was used, is reported in Table 3 and shows that LocalDeepSkip trained on artificially noised data achieves marginally better results with a mean VPT of $\mathbb{E}[VPT] = 5.0$ but with an approximately 2.7 orders of magnitude smaller model.

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

| Source | Model | $\log_{10}$(model size) | $\mathbb{E}$[VPT] | $N$ | $\Delta t$ | $\varepsilon$ |
|---|---|---|---|---|---|---|
| Vlachas et al. (2022) [9] | 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 |

TABLE 3. Comparison of mean VPT and corresponding model sizes from recent benchmark results for forecasting the KS system (11) with 512 spatial grid points on a domain of length $L = 200$. The result corresponding to best $\mathbb{E}$[VPT] is reported for each source. The largest $\mathbb{E}$[VPT] and the corresponding smallest model size are indicated with coloring.

are able to reproduce the invariant measure of the KS equation, as seen in Figure 13. 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.
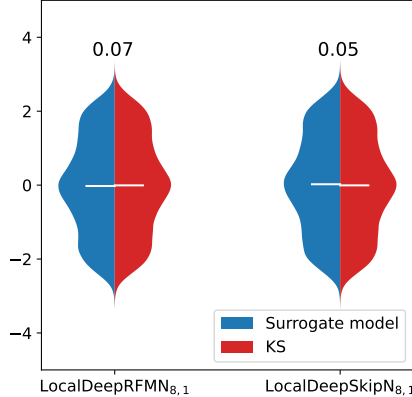


FIGURE 13. Marginal densities for a component of the invariant measure of the KS system, estimated from the original KS system (11) and various surrogate models. We employ translational symmetry and use all 512 components to estimate the densities. The respective Wasserstein $W_2$ distances are indicated at the top of the kernel density plots. We used the best models marked with the $*$-symbol in Figure 12 to generate the data.

## 4. 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 behaviour. 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 the hit-and-run algorithm developed in [19]. 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 the 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 Kuramoto-Sivashinsky equation with more than an order of magnitude smaller model size when compared against recent benchmark results.

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 [12], 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. [12] reports 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 was used in [9].

We considered here noise-free and complete observations. RFMs can be used for noisy observations in conjunction with data assimilation procedures such as the ensemble Kalman filter to control the noise [16]. They can also be used for partial observations by formulating the learning task in time-delay coordinates [17].

## 5. Acknowledgments

## 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). URL https://www.science.org/doi/abs/10.1126/science.adi2336.

[3] Price, I. *et al.* Probabilistic weather forecasting with machine learning. *Nature* (2024).

[4] Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural Computation* **9**, 1735–1780 (1997). URL https://doi.org/10.1162/neco.1997.9.8.1735.

[5] Maass, W., Natschläger, T. & Markram, H. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation* **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] Pathak, J., Hunt, B., Girvan, M., Lu, Z. & Ott, E. Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. *Physical Review Letters* **120**, 024102 (2018).

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

[10] Nakajima, K. & Fischer, I. *Reservoir Computing* (Springer, 2021).

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

[12] 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 Networks* **153**, 530–552 (2022).

[13] Gauthier, D. J., Bollt, E., Griffith, A. & Barbosa, W. A. S. Next generation reservoir computing. *Nature Communications* **12**, 5564 (2021).

[14] 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: An Interdisciplinary Journal of Nonlinear Science* **31**, 013108 (2021). URL `https://doi.org/10.1063/5.0024890`.

[15] 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). URL `http://papers.nips.cc/paper/3182-random-features-for-large-scale-kernel-machines.pdf`.

[16] Gottwald, G. A. & Reich, S. Supervised learning from noisy observations: Combining machine-learning techniques with data assimilation. *Physica D: Nonlinear Phenomena* **423**, 132911 (2021).

[17] Gottwald, G. A. & Reich, S. Combining machine learning and data assimilation to forecast dynamical systems from noisy partial observations. *Chaos: An Interdisciplinary Journal of Nonlinear Science* **31**, 101103 (2021). URL `https://doi.org/10.1063/5.0066080`.

[18] Nelsen, N. H. & Stuart, A. M. The random feature model for input-output maps between Banach spaces. *SIAM Journal on Scientific Computing* **43**, A3212–A3243 (2021). URL `https://doi.org/10.1137/20M133957X`.

[19] Mandal, P. & Gottwald, G. A. On the choice of the non-trainable internal weights in random feature maps. *arXiv preprint arXiv:2408.03626* (2024).

[20] Cybenko, G. Approximation by superposition of a sigmoidal function. *Math. Contr., Sign., and Syst.* **2**, 303–314 (1989).

[21] Park, J. & Sandberg, I. Universal approximation using radial-basis-function networks. *Neural Computation* **3**, 246–257 (1991).

[22] Barron, A. Universal approximation bounds for superposition of a sigmoidal function. *IEEE Trans. on Inform. Theory* **39**, 930–945 (1993).

[23] 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).

[24] Oliver R. A. Dunbar, M. M., Nicholas H. Nelsen. Hyperparameter optimization for randomized algorithms: A case study for random features. *arXiv preprint arXiv:2407.00584* (2024).

[25] 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).

[26] Ceni, A. & Gallicchio, C. Residual Echo State Networks: Residual recurrent neural networks with stable dynamics and fast learning. *Neurocomputing* **597**, 127966 (2024). URL `https://www.sciencedirect.com/science/article/pii/S0925231224007379`.

[27] E, W. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics* **5**, 1–11 (2017).

[28] Akiyama, T. & Tanaka, G. Computational efficiency of multi-step learning echo state networks for nonlinear time series prediction. *IEEE Access* **10**, 28535–28544 (2022).

[29] Ding, S., Zhang, N., Xu, X., Guo, L. & Zhang, J. Deep extreme learning machine and its application in EEG classification. *Mathematical Problems in Engineering* **2015**, 129021 (2015). URL `https://onlinelibrary.wiley.com/doi/abs/10.1155/2015/129021`.

[30] Uzair, M., Shafait, F., Ghanem, B. & Mian, A. Representation learning with deep extreme learning machines for efficient image set classification. *Neural Computing and Applications* **30**, 1211–1223 (2018).

[31] Gottwald, G. & Reich, S. Localized Schrödinger bridge sampler. *arXiv:2409.07968* (2024).

[32] Spielman, D. A. & Teng, S.-H. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *Journal of the ACM (JACM)* **51**, 385–463 (2004).

[33] 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).

[34] Lorenz, E. N. Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences* **20**, 130–141 (1963).

[35] Köster, F., Patel, D., Wikner, A., Jaurigue, L. & Lüdge, K. Data-informed reservoir computing for efficient time-series prediction. *Chaos: An Interdisciplinary Journal of Nonlinear Science* **33** (2023).

[36] Lorenz, E. N. Predictability: A problem partly solved. In *Proc. Seminar on predictability*, vol. 1 (Reading, 1996).

[37] Penny, S. G. *et al.* Integrating recurrent neural networks with data assimilation for scalable data-driven state estimation. *Journal of Advances in Modeling Earth Systems* **14**, e2021MS002843 (2022).

[38] Kuramoto, Y. Diffusion-induced chaos in reaction systems. *Progress of Theoretical Physics Supplement* **64**, 346–367 (1978).

[39] Sivashinsky, G. Nonlinear analysis of hydrodynamic instability in laminar flames—i. derivation of basic equations. In *Dynamics of Curved Fronts*, 459–488 (Elsevier, 1988).

[40] Kassam, A.-K. & Trefethen, L. N. Fourth-order time-stepping for stiff PDEs. *SIAM Journal on Scientific Computing* **26**, 1214–1233 (2005).

## 6. Appendix

### 6.1. **One-shot hit-and-run algorithm to draw internal weights for random feature maps.**

---

**Algorithm 1** Hit-and-run sampling for a row of the internal augmented matrix $\mathbf{W}_{\text{in}}|\mathbf{b}_{\text{in}}$

---

1: Input: training data $\mathbf{U} = [\mathbf{u_1}, \mathbf{u_2}, \cdots, \mathbf{u}_N]$. Boundaries for the good range of the tanh-function $L_{0,1}$.
   Here $L_0 = 0.4$ and $L_1 = 3.5$.
2: Sample $b$ uniformly from $(L_0, L_1)$, the "good" part of the domain of tanh.
3: Select a sign vector $\mathbf{s}$ uniformly randomly from $\{-1, 1\}^D$.
4: **for** $i = 1, \cdots, D$ **do**
5:     **if** $\mathbf{s}_i = 1$ **then**
6:         $\mathbf{x}_{-,i} \leftarrow \min\limits_{1 \le n \le N} \mathbf{u}_{n,i}$
7:         $\mathbf{x}_{+,i} \leftarrow \max\limits_{1 \le n \le N} \mathbf{u}_{n,i}$
8:     **else**
9:         $\mathbf{x}_{-,i} \leftarrow \max\limits_{1 \le n \le N} \mathbf{u}_{n,i}$
10:        $\mathbf{x}_{+,i} \leftarrow \min\limits_{1 \le n \le N} \mathbf{u}_{n,i}$
11:     **end if**
12: **end for**
13: $V \leftarrow \{\mathbf{w} \in \mathbb{R}^D : \text{sgn}(\mathbf{w}_i) \in \{\mathbf{s}_i, 0\} \ \forall \, i = 1, 2, \ldots, D\}$
14: Randomly select a unit vector $\mathbf{d} \in V$.
15: $c_0 \leftarrow 0$.
16: $c_1 \leftarrow \inf\left(\left\{\frac{L_0 - b}{\mathbf{d} \cdot \mathbf{x}_-}, \frac{L_1 - b}{\mathbf{d} \cdot \mathbf{x}_+}\right\} \cap \left(\mathbb{R}_{>0} \cup \{+\infty\}\right)\right)$ with the convention $\inf \varnothing = +\infty$.
17: Sample $c$ uniformly from $(c_0, c_1)$.
18: $\mathbf{w} \leftarrow c\mathbf{d}$
19: Uniformly sample a scalar $z$ from $\{-1, 1\}$.
20: **if** $z = 1$ **then**
21:     $(\mathbf{w}, b)$ is our final row sample.
22: **else**
23:     $-(\mathbf{w}, b)$ is our final row sample.
24: **end if**

---

6.2. **Additional numerical details.** In this section we document additional numerical details corresponding to the experimental results in Section 3 for the Lorenz-63 system, the Lorenz-96 system and the Kuramoto-Sivashinsky equation. We present tables summarizing our results. Each row summarizes the results for 500 samples that differ in their training data, testing data, and the non-trainable random weights and biases of the corresponding model. Along with the model details and mean, standard deviation, median, minimum and maximum of VPT, each row also shows the corresponding value of the regularization hyperparameter $\beta$ used in the experiments as well as the average training time in seconds which includes the run-time of algorithm 1. For all models trained on noisy data, a zero-mean Gaussian noise with standard deviation $10^{-3}$ was used. For each architecture, the best performing model has been highlighted by a red shading.

6.2.1. *Lorenz-63.* We use two different setups for the L63 system. Table 4 documents results for $(N, \Delta t, \varepsilon) = (5 \times 10^4, 0.01, 0.3)$ which is also used in [12]. Table 5 documents results for $(N, \Delta t, \varepsilon) = (2 \times 10^4, 0.02, \sqrt{0.05})$ corresponding to the setup used in [16, 19]. A similar setup with $\varepsilon = 0.4$ appears in [28]. To generate the training and testing data for L63, we use a burn-in period of 40 model time units.

6.2.2. *Lorenz-96.* For the 40-dimensional L96 system with forcing $F = 10$ we use $(N, \Delta t, \varepsilon) = (10^5, 0.01, \sqrt{0.5})$ corresponding to the setup used in [12, 9]. Tables 6 and 7 document the results for non-localized and localized architectures, respectively. Note that [12] uses $F = 8$ and [9] uses both $F = 8$ and 10. Section 4 of [9] shows that trained surrogate models demonstrate similar forecasting skill for both values of $F$. This justifies comparing our results with [12, 9]. To generate the training and testing data for the L96 system, we use a

| Model | | | | VPT | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| architecture | $D_r$ | $B$ | model size | mean | std | median | min | max | $\beta$ | $\mathbb{E}[t_{\text{train}}](s)$ |
| RFM | 512 | 1 | 3,584 | 9.8 | 1.8 | 9.8 | 4.0 | 16.0 | 3.52e-09 | 1.1e-02 |
| | 1,024 | 1 | 7,168 | 9.8 | 1.5 | 9.8 | 4.9 | 15.5 | 6.40e-09 | 1.6e-02 |
| | 2,048 | 1 | 14,336 | 9.3 | 1.5 | 9.3 | 4.0 | 15.3 | 4.96e-08 | 4.4e-02 |
| | 4,096 | 1 | 28,672 | 9.3 | 1.5 | 9.3 | 3.8 | 13.8 | 8.20e-08 | 1.2e-01 |
| | 8,192 | 1 | 57,344 | 9.5 | 1.7 | 9.5 | 3.8 | 16.3 | 6.76e-08 | 4.4e-01 |
| | 16,384 | 1 | 114,688 | 9.5 | 1.5 | 9.6 | 4.9 | 18.9 | 8.92e-08 | 2.1e+00 |
| SkipRFM | 512 | 1 | 3,584 | 10.1 | 1.7 | 10.0 | 4.1 | 16.9 | 3.88e-09 | 7.2e-03 |
| | 1,024 | 1 | 7,168 | 10.5 | 1.5 | 10.5 | 5.5 | 15.1 | 6.40e-09 | 1.6e-02 |
| | 2,048 | 1 | 14,336 | 10.3 | 1.6 | 10.3 | 5.0 | 19.5 | 3.16e-08 | 4.4e-02 |
| | 4,096 | 1 | 28,672 | 10.3 | 1.5 | 10.3 | 6.5 | 16.2 | 7.12e-08 | 1.2e-01 |
| | 8,192 | 1 | 57,344 | 10.6 | 1.5 | 10.7 | 5.2 | 16.1 | 6.76e-08 | 4.4e-01 |
| | 16,384 | 1 | 114,688 | 10.4 | 1.6 | 10.4 | 5.0 | 17.2 | 2.44e-07 | 2.1e+00 |
| DeepSkip | 1,024 | 1 | 10,240 | 10.1 | 1.7 | 10.0 | 4.0 | 17.0 | 4.96e-09 | 1.6e-02 |
| | 1,024 | 2 | 20,480 | 10.9 | 1.7 | 11.0 | 4.9 | 17.7 | 4.96e-09 | 3.2e-02 |
| | 1,024 | 4 | 40,960 | 11.3 | 1.7 | 11.3 | 4.9 | 18.4 | 4.96e-09 | 6.2e-02 |
| | 4,096 | 1 | 40,960 | 9.9 | 1.6 | 9.9 | 3.9 | 18.6 | 5.32e-08 | 1.2e-01 |
| | 1,024 | 8 | 81,920 | 11.4 | 1.6 | 11.5 | 5.3 | 16.6 | 4.96e-09 | 1.2e-01 |
| | 4,096 | 2 | 81,920 | 10.9 | 1.7 | 11.0 | 4.9 | 19.5 | 5.32e-08 | 2.4e-01 |
| | 8,192 | 1 | 81,920 | 10.3 | 1.6 | 10.3 | 4.9 | 16.8 | 6.76e-08 | 4.4e-01 |
| | 1,024 | 16 | 163,840 | 11.7 | 1.7 | 11.8 | 6.3 | 21.2 | 4.96e-09 | 2.4e-01 |
| | 2,048 | 8 | 163,840 | 11.4 | 1.7 | 11.4 | 5.7 | 16.7 | 2.19e-08 | 3.2e-01 |
| | 4,096 | 4 | 163,840 | 11.0 | 1.6 | 11.1 | 4.9 | 16.4 | 5.32e-08 | 4.9e-01 |
| | 8,192 | 2 | 163,840 | 11.0 | 1.5 | 11.1 | 5.0 | 16.0 | 6.76e-08 | 9.0e-01 |
| | 16,384 | 1 | 163,840 | 10.7 | 1.6 | 10.7 | 5.6 | 17.8 | 8.92e-08 | 2.1e+00 |
| | 1,024 | 32 | 327,680 | 12.0 | 1.5 | 12.0 | 6.4 | 20.1 | 4.96e-09 | 4.6e-01 |
| | 2,048 | 16 | 327,680 | 11.5 | 1.6 | 11.4 | 5.7 | 17.5 | 2.19e-08 | 6.4e-01 |
| | 4,096 | 8 | 327,680 | 11.2 | 1.6 | 11.3 | 6.3 | 18.4 | 5.32e-08 | 1.0e+00 |
| | 8,192 | 4 | 327,680 | 11.2 | 1.6 | 11.3 | 5.2 | 18.3 | 6.76e-08 | 1.8e+00 |
| | 16,384 | 2 | 327,680 | 10.9 | 1.5 | 10.9 | 5.0 | 17.3 | 8.92e-08 | 4.3e+00 |

TABLE 4. Results for the L63 system with $N = 5 \times 10^4$, $\Delta t = 0.01$ and $\varepsilon = 0.3$ for various surrogate models.

burn-in period of $1,000$ model time units. The training data matrix for the L96 system is well-conditioned, so introducing noise does not enhance the quality of the trained surrogate model (cf. the last row of Table 7 where we report results for LocalDeepSkip$_{2,2}$ trained on artificially noisy data).

6.2.3. *Kuramoto-Sivashinsky.* For the KS equation with domain length $L = 200$ and 512 spatial grid points we use $(N, \Delta t, \varepsilon) = (10^5, 0.25, 0.5)$, corresponding to the setup used in [9]. To generate the training and testing data for KS, we use a burn-in period of $2.5 \times 10^4$ model time units. We used the following initial condition,

$$u(x,0) = \cos\left(\frac{2\pi x}{L}\right)\left(1 + \sin\left(\frac{2\pi x}{L}\right)\right). \tag{12}$$

Table 8 documents the results.

6.3. **Localization schemes.** In this section we discuss the efficacy of various localization schemes for the 40-dimensional L96 system and the 512-dimensional discretization of the KS equation. Figures 14 and 15 show crude estimates of the mean VPT as a function of the regularization parameter $\beta$ for these systems, respectively. These estimates were computed by averaging over 5 samples differing in the training data, the testing data and the non-trainable internal weights and biases for each value of $\beta$. The data shown in this

| Model | | | | VPT | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| architecture | $D_r$ | $B$ | model size | mean | std | median | min | max | $\beta$ | $\mathbb{E}[t_{\text{train}}]$(s) |
| SkipRFM | 512 | 1 | 3,584 | 10.1 | 1.7 | 10.0 | 4.6 | 16.7 | 6.04e-10 | 8.8e-03 |
| | 1,024 | 1 | 7,168 | 10.4 | 1.4 | 10.4 | 4.8 | 16.1 | 8.74e-10 | 1.1e-02 |
| | 2,048 | 1 | 14,336 | 10.0 | 1.5 | 10.1 | 4.7 | 17.5 | 4.24e-09 | 2.6e-02 |
| | 4,096 | 1 | 28,672 | 10.1 | 1.4 | 10.1 | 4.8 | 15.7 | 9.46e-09 | 6.2e-02 |
| | 8,192 | 1 | 57,344 | 10.1 | 1.5 | 10.1 | 4.7 | 15.9 | 2.26e-08 | 2.2e-01 |
| | 16,384 | 1 | 114,688 | 10.3 | 1.5 | 10.2 | 4.7 | 15.9 | 2.62e-08 | 8.9e-01 |
| DeepSkip | 1,024 | 1 | 10,240 | 9.7 | 1.6 | 9.7 | 4.6 | 16.0 | 9.46e-10 | 1.1e-02 |
| | 1,024 | 2 | 20,480 | 10.9 | 1.6 | 10.7 | 4.8 | 17.3 | 9.46e-10 | 2.1e-02 |
| | 1,024 | 4 | 40,960 | 11.0 | 1.6 | 10.8 | 5.6 | 17.7 | 9.46e-10 | 4.2e-02 |
| | 4,096 | 1 | 40,960 | 9.8 | 1.6 | 9.8 | 4.6 | 16.9 | 9.28e-09 | 6.2e-02 |
| | 1,024 | 8 | 81,920 | 11.3 | 1.5 | 11.2 | 4.8 | 17.5 | 9.46e-10 | 7.7e-02 |
| | 4,096 | 2 | 81,920 | 10.6 | 1.5 | 10.6 | 4.7 | 17.1 | 9.28e-09 | 1.3e-01 |
| | 8,192 | 1 | 81,920 | 9.2 | 1.4 | 9.2 | 4.6 | 14.2 | 3.70e-08 | 2.2e-01 |
| | 1,024 | 16 | 163,840 | 11.7 | 1.6 | 11.6 | 6.4 | 18.2 | 9.46e-10 | 1.5e-01 |
| | 4,096 | 4 | 163,840 | 10.8 | 1.6 | 10.7 | 4.8 | 18.2 | 9.28e-09 | 2.5e-01 |
| | 8,192 | 2 | 163,840 | 10.4 | 1.5 | 10.4 | 5.4 | 17.8 | 3.70e-08 | 4.4e-01 |
| | 16,384 | 1 | 163,840 | 9.5 | 1.4 | 9.6 | 4.7 | 14.5 | 5.14e-08 | 8.9e-01 |
| | 1,024 | 32 | 327,680 | 11.8 | 1.5 | 11.7 | 7.8 | 18.2 | 9.46e-10 | 3.1e-01 |
| | 4,096 | 8 | 327,680 | 11.0 | 1.4 | 11.0 | 5.6 | 18.2 | 9.28e-09 | 5.1e-01 |
| | 8,192 | 4 | 327,680 | 10.6 | 1.5 | 10.6 | 5.5 | 15.2 | 3.70e-08 | 9.0e-01 |
| | 16,384 | 2 | 327,680 | 10.5 | 1.5 | 10.5 | 5.5 | 18.2 | 5.14e-08 | 1.8e+00 |

TABLE 5. Results for the L63 system with $N = 2 \times 10^4$, $\Delta t = 0.02$ and $\varepsilon = \sqrt{0.05} \approx 0.224$ for various surrogate models.

| Model | | | | VPT | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| architecture | $D_r$ | $B$ | model size | mean | std | median | min | max | $\beta$ | $\mathbb{E}[t_{\text{train}}]$(s) |
| SkipRFM | 512 | 1 | 41,472 | 0.3 | 0.1 | 0.3 | 0.2 | 0.7 | 3.52e-09 | 1.6e-02 |
| | 1,024 | 1 | 82,944 | 1.0 | 0.2 | 0.9 | 0.6 | 2.1 | 6.40e-09 | 2.4e-02 |
| | 2,048 | 1 | 165,888 | 2.0 | 0.5 | 2.0 | 1.0 | 4.4 | 4.60e-08 | 6.6e-02 |
| | 4,096 | 1 | 331,776 | 2.2 | 0.5 | 2.2 | 1.1 | 4.1 | 3.16e-07 | 2.3e-01 |
| | 8,192 | 1 | 663,552 | 2.3 | 0.6 | 2.3 | 1.2 | 4.2 | 3.16e-07 | 1.0e+00 |
| DeepSkip | 4,096 | 1 | 495,616 | 2.3 | 0.5 | 2.2 | 1.1 | 4.8 | 1.72e-07 | 2.4e-01 |
| | 4,096 | 2 | 991,232 | 2.7 | 0.6 | 2.6 | 1.1 | 5.0 | 1.72e-07 | 5.0e-01 |
| | 4,096 | 4 | 1,982,464 | 2.7 | 0.6 | 2.7 | 1.4 | 5.0 | 1.72e-07 | 1.0e+00 |
| | 4,096 | 8 | 3,964,928 | 2.8 | 0.6 | 2.8 | 1.5 | 4.6 | 1.72e-07 | 2.0e+00 |
| | 4,096 | 16 | 7,929,856 | 2.8 | 0.6 | 2.8 | 1.5 | 5.7 | 1.72e-07 | 4.0e+00 |

TABLE 6. Results for non-localized architectures for the L96 system with $N = 10^5$, $\Delta t = 0.01$ and $\varepsilon = 0.5$ for various surrogate models.

section correspond to a fixed training data size $N = 10^5$. Figure 14 shows that $(G, I) = (1, 4)$ and $(2, 2)$ are the best performing localization schemes for L96 and Figure 15 shows that overall $(G, I) = (8, 1)$ is the best performing localization scheme for KS. Comparing the first and second panels of Figure 15 we see that the optimal localization scheme varies for different values of $D_r$.

While choosing a localization scheme, practitioners should consider several factors such as hardware e.g. available GPU memory, model size, $D_r$, amount of training data $N$, inferences drawn from the decay of the spatial correlation of the system, physical intuitions about the underlying dynamical system etc. If $G_2 > G_1$ then for the same architecture and $D_r$, the model using scheme $(G_2, I_2)$ will typically have larger

| Model | | | | VPT | | | | | $\beta$ | $\mathbb{E}[t_{\text{train}}](s)$ |
|---|---|---|---|---|---|---|---|---|---|---|
| architecture | $D_r$ | $B$ | model size | mean | std | median | min | max | | |
| LocalSkip$_{2,2}$ | 512 | 1 | 6,656 | 4.4 | 0.9 | 4.3 | 2.0 | 7.6 | 3.16e-09 | 6.3e-02 |
| | 1,024 | 1 | 13,312 | 5.3 | 1.1 | 5.3 | 1.9 | 9.1 | 3.16e-08 | 7.8e-02 |
| | 2,048 | 1 | 26,624 | 5.7 | 1.0 | 5.7 | 3.2 | 8.9 | 8.92e-08 | 1.2e-01 |
| | 4,096 | 1 | 53,248 | 6.5 | 1.2 | 6.4 | 3.5 | 11.1 | 1.00e-07 | 2.8e-01 |
| | 8,192 | 1 | 106,496 | 6.7 | 1.1 | 6.8 | 4.1 | 10.4 | 4.24e-07 | 1.1e+00 |
| | 16,384 | 1 | 212,992 | 6.8 | 1.2 | 6.8 | 3.4 | 10.7 | 7.48e-07 | 4.4e+00 |
| LocalDeepRFM$_{2,2}$ | 512 | 4 | 30,720 | 4.8 | 0.9 | 4.7 | 2.3 | 7.4 | 5.32e-09 | 4.7e-02 |
| | 1,024 | 4 | 61,440 | 5.9 | 1.1 | 5.9 | 2.8 | 9.2 | 1.72e-08 | 9.1e-02 |
| | 2,048 | 4 | 122,880 | 6.2 | 1.1 | 6.2 | 2.9 | 9.5 | 1.36e-07 | 2.7e-01 |
| | 4,096 | 4 | 245,760 | 6.6 | 1.2 | 6.6 | 3.5 | 10.0 | 1.72e-07 | 9.4e-01 |
| | 8,192 | 2 | 245,760 | 6.9 | 1.2 | 7.0 | 4.0 | 11.1 | 3.16e-07 | 2.0e+00 |
| | 11,586 | 2 | 347,580 | 7.1 | 1.3 | 7.0 | 3.9 | 11.5 | 3.52e-07 | 4.3e+00 |
| | 8,192 | 4 | 491,520 | 7.0 | 1.3 | 7.0 | 3.7 | 11.2 | 3.16e-07 | 4.2e+00 |
| | 16,384 | 2 | 491,520 | 7.2 | 1.3 | 7.1 | 3.9 | 11.3 | 3.88e-07 | 9.5e+00 |
| | 11,586 | 4 | 695,160 | 7.1 | 1.3 | 7.0 | 4.0 | 11.2 | 3.52e-07 | 8.8e+00 |
| LocalDeepSkip$_{1,4}$ | 16,384 | 2 | 393,216 | 6.9 | 1.3 | 6.9 | 3.7 | 11.1 | 6.40e-07 | 9.4e+00 |
| LocalDeepSkip$_{2,2}$ | 1,024 | 1 | 15,360 | 5.5 | 1.1 | 5.5 | 2.7 | 8.7 | 9.64e-09 | 9.7e-02 |
| | 1,024 | 2 | 30,720 | 5.8 | 1.2 | 5.8 | 2.5 | 10.4 | 9.64e-09 | 1.2e-01 |
| | 2,048 | 1 | 30,720 | 5.8 | 1.1 | 5.7 | 2.3 | 9.2 | 4.96e-08 | 1.3e-01 |
| | 1,024 | 4 | 61,440 | 6.0 | 1.2 | 5.9 | 2.5 | 9.3 | 9.64e-09 | 1.6e-01 |
| | 2,048 | 2 | 61,440 | 6.3 | 1.2 | 6.2 | 2.7 | 11.1 | 4.96e-08 | 2.0e-01 |
| | 4,096 | 1 | 61,440 | 6.0 | 1.1 | 5.9 | 3.5 | 10.0 | 3.88e-07 | 3.2e-01 |
| | 1,024 | 8 | 122,880 | 6.0 | 1.1 | 6.0 | 2.9 | 9.0 | 9.64e-09 | 2.5e-01 |
| | 2,048 | 4 | 122,880 | 6.4 | 1.2 | 6.4 | 3.4 | 10.5 | 4.96e-08 | 3.4e-01 |
| | 4,096 | 2 | 122,880 | 6.6 | 1.2 | 6.6 | 3.0 | 10.1 | 3.88e-07 | 5.8e-01 |
| | 8,192 | 1 | 122,880 | 6.2 | 1.1 | 6.2 | 3.5 | 9.6 | 9.64e-07 | 1.2e+00 |
| | 1,024 | 16 | 245,760 | 6.1 | 1.2 | 6.0 | 3.3 | 10.5 | 9.64e-09 | 4.3e-01 |
| | 2,048 | 8 | 245,760 | 6.5 | 1.2 | 6.5 | 2.6 | 10.2 | 4.96e-08 | 6.1e-01 |
| | 4,096 | 4 | 245,760 | 6.7 | 1.2 | 6.7 | 3.7 | 10.4 | 3.88e-07 | 1.1e+00 |
| | 8,192 | 2 | 245,760 | 6.8 | 1.3 | 6.7 | 3.4 | 10.4 | 9.64e-07 | 2.3e+00 |
| | 16,384 | 1 | 245,760 | 7.0 | 1.2 | 7.0 | 3.9 | 11.1 | 3.88e-07 | 4.5e+00 |
| | 1,024 | 32 | 491,520 | 6.3 | 1.2 | 6.2 | 3.0 | 10.1 | 9.64e-09 | 7.8e-01 |
| | 2,048 | 16 | 491,520 | 6.6 | 1.2 | 6.6 | 3.2 | 11.5 | 4.96e-08 | 1.1e+00 |
| | 4,096 | 8 | 491,520 | 6.9 | 1.2 | 6.8 | 4.1 | 10.7 | 3.88e-07 | 2.1e+00 |
| | 8,192 | 4 | 491,520 | 7.0 | 1.2 | 6.9 | 3.4 | 12.1 | 9.64e-07 | 4.5e+00 |
| | 16,384 | 2 | 491,520 | 7.3 | 1.2 | 7.2 | 4.3 | 11.7 | 3.88e-07 | 9.1e+00 |
| | 2,048 | 32 | 983,040 | 6.7 | 1.2 | 6.7 | 4.1 | 10.8 | 4.96e-08 | 2.2e+00 |
| | 4,096 | 16 | 983,040 | 7.0 | 1.3 | 7.0 | 3.7 | 11.1 | 3.88e-07 | 4.1e+00 |
| | 8,192 | 8 | 983,040 | 7.1 | 1.3 | 7.0 | 3.9 | 12.0 | 9.64e-07 | 8.9e+00 |
| | 16,384 | 4 | 983,040 | 7.2 | 1.2 | 7.2 | 4.1 | 11.8 | 3.88e-07 | 1.8e+01 |
| LocalDeepSkipN$_{2,2}$ | 16,384 | 2 | 491,520 | 7.1 | 1.3 | 7.1 | 3.8 | 10.8 | 3.88e-07 | 9.5e+00 |

TABLE 7. Results for localized architectures for the L96 system with $N = 10^5$, $\Delta t = 0.01$ and $\varepsilon = 0.5$ for various surrogate models.

size compared to the model using scheme $(G_1, I_1)$. Since the GPU memory used during training is primarily a function of $ND_r$, for the same $N$ both models occupy roughly the same amount of memory on the GPU during training, despite the model using scheme $(G_1, I_1)$ having smaller size. Therefore, if our goal is to fit the largest possible model on the GPU during training, we should opt for the localization scheme with larger

| Model | | | | VPT | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| architecture | $D_r$ | $B$ | model size | mean | std | median | min | max | $\beta$ | $\mathbb{E}[t_{\text{train}}]$(s) |
| LocalRFM$_{8,1}$ | 8,192 | 1 | 270,336 | 3.6 | 1.3 | 3.9 | 0.5 | 6.5 | 8.56e-06 | 1.2e+00 |
| | 15,000 | 1 | 495,000 | 4.3 | 0.8 | 4.3 | 1.5 | 6.3 | 2.80e-05 | 4.1e+00 |
| LocalDeepRFM$_{8,1}$ | 8,192 | 2 | 671,744 | 4.0 | 1.3 | 4.2 | 0.5 | 6.8 | 4.60e-06 | 2.1e+00 |
| | 8,192 | 3 | 1,007,616 | 4.5 | 1.0 | 4.6 | 1.0 | 7.1 | 3.52e-05 | 3.1e+00 |
| | 14,000 | 2 | 1,148,000 | 4.8 | 1.0 | 4.9 | 2.1 | 7.1 | 2.00e-05 | 6.5e+00 |
| | 15,000 | 2 | 1,230,000 | 4.6 | 0.9 | 4.7 | 2.1 | 6.9 | 4.24e-05 | 7.4e+00 |
| | 15,000 | 3 | 1,845,000 | 4.7 | 1.0 | 4.8 | 1.7 | 7.3 | 3.88e-05 | 1.1e+01 |
| | 13,308 | 5 | 2,728,140 | 4.6 | 1.0 | 4.7 | 1.9 | 7.0 | 9.55e-05 | 1.5e+01 |
| LocalDeepSkip$_{8,1}$ | 15,000 | 2 | 1,230,000 | 0.5 | 0.1 | 0.5 | 0.4 | 0.8 | 2.00e-05 | 7.9e+00 |
| LocalRFMN$_{8,1}$ | 15,000 | 1 | 495,000 | 4.3 | 0.9 | 4.4 | 2.0 | 6.4 | 4.24e-05 | 3.8e+00 |
| LocalSkipN$_{8,1}$ | 15,000 | 1 | 495,000 | 4.3 | 0.8 | 4.4 | 2.0 | 6.4 | 4.24e-05 | 3.8e+00 |
| LocalDeepRFMN$_{8,1}$ | 14,000 | 2 | 1,148,000 | 4.9 | 0.9 | 5.1 | 2.7 | 7.0 | 2.00e-05 | 6.3e+00 |
| | 15,000 | 2 | 1,230,000 | 5.0 | 0.9 | 5.0 | 2.7 | 7.6 | 2.00e-05 | 7.6e+00 |
| LocalDeepSkipN$_{8,1}$ | 15,000 | 2 | 1,230,000 | 5.0 | 0.9 | 5.1 | 2.6 | 7.7 | 2.00e-05 | 7.9e+00 |

TABLE 8. Results for the KS equation with $N = 10^5$, $\Delta t = 0.25$ and $\varepsilon = 0.5$ for various surrogate models.
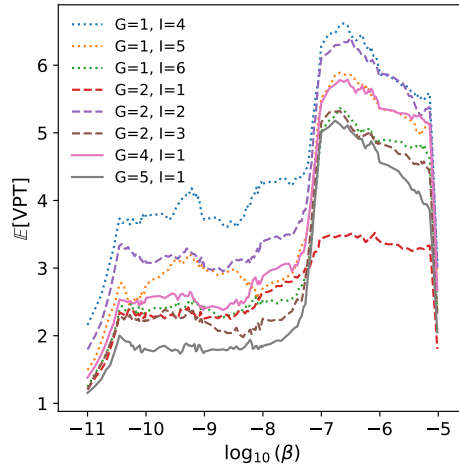


FIGURE 14. Estimates of the mean VPT as a function of the regularization hyperparameter $\beta$ for different localization schemes for the L96 system. The models depicted here are LocalSkip with $D_r = 4,096$.

$G$. These considerations lead us to choose $(G, I) = (2, 2)$ for L96 and $(G, I) = (8, 1)$ for KS. These choices are consistent with those employed in [12, 9].

6.4. **Effect of depth on training time.** In this section we demonstrate that deeper models train faster using the L63 system (9) as an example. Figure 16 shows that for both non-localized and localized architectures, making a model deeper while keeping its model size $S$ fixed leads to faster training times. In fact, for both cases we see that the training time can be reduced by an order of magnitude by increasing the depth $B$. This is achieved because the linear regression problem occupies smaller space on the GPU for deeper models, as discussed in Section 2.4. Note that the training time shown here includes the run-time of the sampling algorithm 1, which accounts for only a small fraction of the total time.
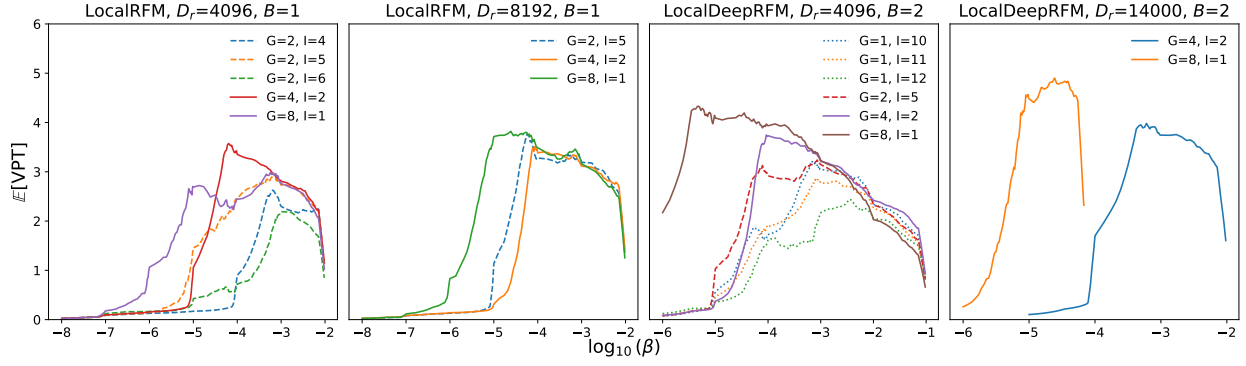
FIGURE 15. Estimates of the mean VPT as a function of the regularization hyperparameter $\beta$ for different localization schemes for the KS equation for various localized random feature models.
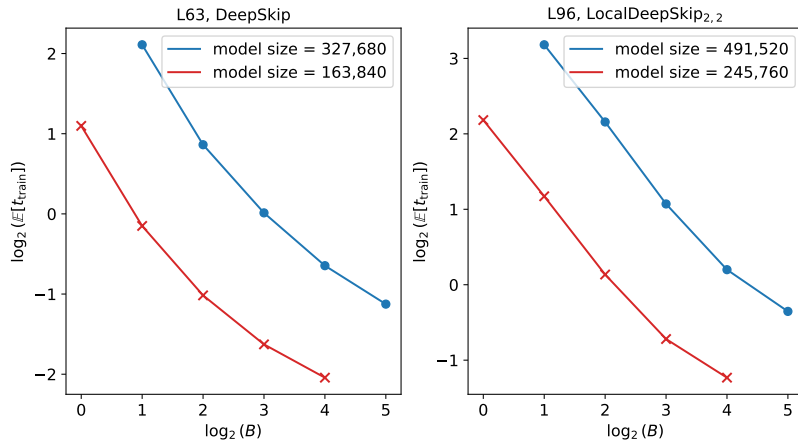


FIGURE 16. Average training time in seconds as a function of depth $B$. The left and right panels show results for DeepSkip and LocalDeepSkip taken from Tables 4 and 7, respectively. Along each curve the model size $S$ remains constant and the width $D_r$ decreases with depth.