JAMES Journal of Advances in Modeling Earth Systems®

RESEARCH ARTICLE
10.1029/2023MS003792

Special Section:
Machine learning application to Earth system modeling

Key Points:

- Reducing training data temporal resolution by subsampling leads to overly dissipative small spatial scales in neural network emulators
- A quadratic autoregressive architecture is shown to be inadequate at capturing small scale turbulence, even when data are not subsampled
- Subsampling bias in Echo State Networks is mitigated but not eliminated by prioritizing kinetic energy spectrum during training

Supporting Information:

Supporting Information may be found in the online version of this article.

Correspondence to:
T. A. Smith,
tim.smith@noaa.gov

Citation:
Smith, T. A., Penny, S. G., Platt, J. A., & Chen, T.-C. (2023). Temporal subsampling diminishes small spatial scales in recurrent neural network emulators of geophysical turbulence. *Journal of Advances in Modeling Earth Systems*, *15*, e2023MS003792. https://doi.org/10.1029/2023MS003792

Received 1 MAY 2023
Accepted 2 NOV 2023

Temporal Subsampling Diminishes Small Spatial Scales in Recurrent Neural Network Emulators of Geophysical Turbulence

Timothy A. Smith[1,2], Stephen G. Penny[1,3], Jason A. Platt[4], and Tse-Chun Chen[5]

[1]Cooperative Institute for Research in Environmental Sciences (CIRES) at the University of Colorado Boulder, Boulder, CO, USA, [2]Physical Sciences Laboratory (PSL), National Oceanic and Atmospheric Administration (NOAA), Boulder, CO, USA, [3]Sofar Ocean, San Francisco, CA, USA, [4]University of California San Diego (UCSD), La Jolla, CA, USA, [5]Pacific Northwest National Laboratory, Richland, WA, USA

Abstract The immense computational cost of traditional numerical weather and climate models has sparked the development of machine learning (ML) based emulators. Because ML methods benefit from long records of training data, it is common to use data sets that are temporally subsampled relative to the time steps required for the numerical integration of differential equations. Here, we investigate how this often overlooked processing step affects the quality of an emulator's predictions. We implement two ML architectures from a class of methods called reservoir computing: (a) a form of Nonlinear Vector Autoregression (NVAR), and (b) an Echo State Network (ESN). Despite their simplicity, it is well documented that these architectures excel at predicting low dimensional chaotic dynamics. We are therefore motivated to test these architectures in an idealized setting of predicting high dimensional geophysical turbulence as represented by Surface Quasi-Geostrophic dynamics. In all cases, subsampling the training data consistently leads to an increased bias at small spatial scales that resembles numerical diffusion. Interestingly, the NVAR architecture becomes unstable when the temporal resolution is increased, indicating that the polynomial based interactions are insufficient at capturing the detailed nonlinearities of the turbulent flow. The ESN architecture is found to be more robust, suggesting a benefit to the more expensive but more general structure. Spectral errors are reduced by including a penalty on the kinetic energy density spectrum during training, although the subsampling related errors persist. Future work is warranted to understand how the temporal resolution of training data affects other ML architectures.

Plain Language Summary The computer models that govern weather prediction and climate projections are extremely costly to run, causing practitioners to make unfortunate tradeoffs between accuracy of the physics and credibility of their statistics. Recent advances in machine learning have sparked the development of neural network-based emulators, that is, low-cost models that can be used as drop-in replacements for the traditional expensive models. Due to the cost of storing large weather and climate data sets, it is common to subsample these fields in time to save disk space. This subsampling also reduces the computational expense of training emulators. Here, we show that this pre-processing step hinders the fidelity of the emulator. We offer one method to mitigate the resulting errors, but more research is needed to understand and eventually overcome them.

1. Introduction

Weather and climate prediction requires the numerical integration of one or more computational models derived from the fundamental equations of motion and initialized with an estimate of the present-day system state (e.g., temperature, wind speeds, etc.). Due to the high cost of these computational models, prediction systems typically require suboptimal tradeoffs. On one hand, it is desirable to increase the credibility of the underlying numerical model as much as possible, for instance by increasing model grid resolution (e.g., Hewitt et al., 2016) or by explicitly simulating as many coupled components (e.g., atmosphere, land, ocean, ice) as possible (e.g., Penny et al., 2017). On the other hand, knowledge of the model initial conditions is imperfect and the governing equations will always contain necessary, inexact approximations of reality. As a result, prediction systems employ statistical methods like ensemble based forecasting in order to represent this uncertainty. Producing an ensemble with statistical significance requires integrating the underlying numerical model many times; usually

$\mathcal{O}(10) - \mathcal{O}(100)$ in practice, but ideally $\mathcal{O}(1000)$ or greater (Evensen et al., 2022). Therefore, the resulting computational costs require practitioners to compromise between the fidelity of the numerical model and credibility of the statistical method.

An ongoing area of research that aims to enable statistical forecasting subject to the dynamics of an expensive numerical model is *surrogate modeling*. The general approach relies on using a model that represents or "emulates" the dynamics of the original numerical model with "sufficient accuracy" for the given application, while being computationally inexpensive to evaluate. Historically, surrogate models have been an important tool for nonlinear optimization (e.g., Bouhlel et al., 2020; Li et al., 2019), and in the Earth sciences have been developed with techniques such as Linear Inverse Models (e.g., principal oscillation or interaction patterns; Hasselmann, 1988; Penland, 1989; Moore et al., 2022), kriging (Cressie, 1993), or polynomial chaos techniques (Najm, 2009), to name only a few. More recently, advances in computing power, the rise of general purpose graphics processing units, and the explosion of freely available data has encouraged the exploration of more expensive machine learning methods like neural networks for the emulation task (Schultz et al., 2021). A number of data-driven, neural network architectures have been developed to generate surrogate models for weather forecasting and climate projection applications. Some examples include models based on feed forward neural networks (Dueben & Bauer, 2018), convolutional neural networks (CNNs; Scher, 2018; Scher & Messori, 2019; Rasp & Thuerey, 2021; Weyn et al., 2019, 2020, 2021), recurrent neural networks (RNNs; Arcomano et al., 2020; X. Chen et al., 2021; Nadiga, 2021), graph neural networks (Keisler, 2022; Lam et al., 2022), Fourier neural operators (Pathak et al., 2022), and encoder-transformer-decoder networks (Bi et al., 2023).

A significant advancement in surrogate modeling for weather and climate prediction has been the rapid increase in spatial resolution. To the best of our knowledge, the current highest resolution neural network emulators for global atmospheric dynamics is ~0.25° (~31 km) (Bi et al., 2023; Lam et al., 2022; Pathak et al., 2022), which is the same resolution as the ERA5 Reanalysis (Hersbach et al., 2020) used to train these models. At this resolution, General Circulation Models (GCMs) of the atmosphere are capable of explicitly capturing important small scale processes like low-level jets and interactions with mountainous topography (Orlanski, 1975). However, it is not yet clear that neural networks are able to represent the same dynamical processes as the training data. Instead, based on our own experimentation, we hypothesize that without careful architectural modifications, neural network emulators will effectively operate at a coarser resolution than the original data set used in training.

To make the discussion concrete, we present a sample prediction from our own surrogate model in Figure 1. The panels show the time evolution of Sea Surface Temperature (SST) in the Gulf of Mexico at 1/25° horizontal resolution, using data from a Navy HYCOM, 3D-Var-based reanalysis product as "Truth" (upper row; see Appendix B for data details). We generate the prediction (middle row) with an RNN architecture described more fully in Section 3.4. Generally speaking, the RNN captures the largest scales of the SST pattern over a 36 hr window. However, as time progresses, the SST pattern becomes overly smooth. The RNN is unable to capture the spatial details that are well resolved in the reanalysis data set, with the largest errors evolving along sharp SST fronts. We note that a similar smoothing behavior can be observed in other neural network based emulators, see for example, (Bi et al., 2023, Figure 3) (Pathak et al., 2022, Figures 4c and 4d) (Keisler, 2022, Figure 5).

There are a number of reasons that could cause this smoothing behavior to manifest in the predictions. As we show in Sections 4 and 5, the blurring of small scale features is a high frequency spectral bias, which has been studied in relation to the training of feedforward neural networks (Xu et al., 2022) and numerical instabilities of neural network predictions for turbulent flows (Chattopadhyay & Hassanzadeh, 2023). One potential reason that we observe spectral bias in our predictions is that the training uses a mean-squared error loss function, which is known to prioritize large over small scale features (Rossa et al., 2008). Here, we suggest that any blurring effect from such a loss function is exacerbated by more fundamental decisions in the experimental design. Our primary goal is to explore how temporal subsampling in the training data set adds to this blurring effect. We are motivated to study the impact of this subsampling because many existing emulators, including our example in Figure 1, rely on reanalysis products as training data (e.g., Arcomano et al., 2020; Bi et al., 2023; Keisler, 2022; Lam et al., 2022; Pathak et al., 2022; Weyn et al., 2021). While there are excellent reasons to leverage the existence of reanalysis products, namely that they are constrained to observational data, the shear size of the data requires some degree of temporal subsampling. It is therefore important to understand how this highly routine data reduction step impacts the performance of data-driven prediction methods when used for training.

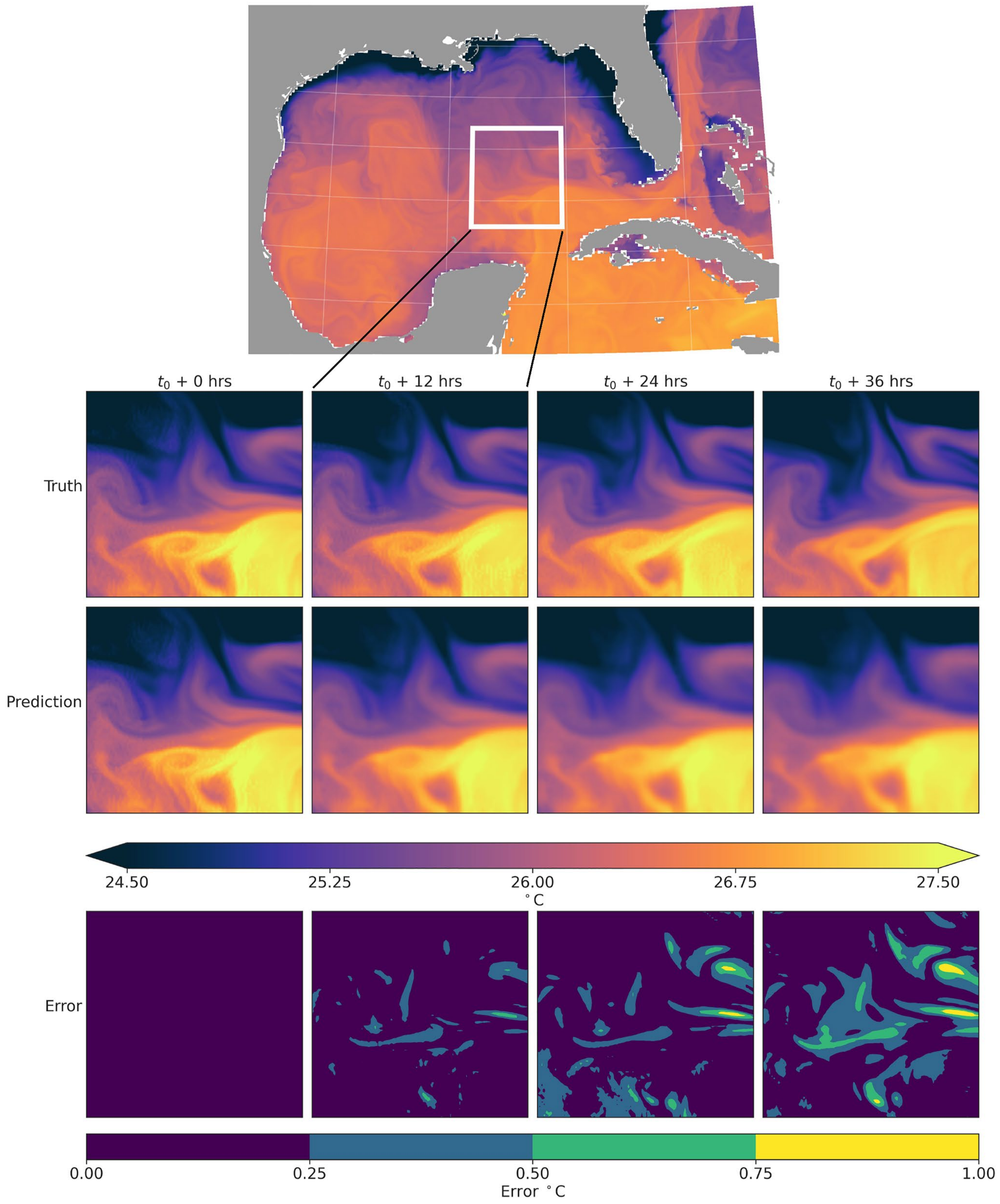

Figure 1. A sample prediction of sea surface temperatures in the Gulf of Mexico at 1/25° horizontal resolution. The upper row (Truth) shows the evolution of unseen test data from the Navy HYCOM reanalysis product, and the middle row shows a prediction from the Echo State Network architecture described in Section 3.4. The bottom row (Error) shows the absolute value of the difference between the two. See Appendix B for a description of the data set.

In our work, we explore the degree to which temporal subsampling impedes single layer autogregressive and recurrent neural network emulators from learning the true underlying dynamics of the system. In order to isolate this effect from the potential impacts of a data assimilation system and multivariate interactions, we do not rely on the Gulf of Mexico reanalysis data. Instead, we use a model for Surface Quasi-Geostrophic (SQG) turbulence (Blumen, 1978a; Held et al., 1995), which additionally gives us direct control over the data sets used for training, validation, and testing. The SQG model and data set generation is described more fully in Section 2.

The architectures that we use in this study stem from a broad class of machine learning techniques termed as reservoir computing (RC), which was independently discovered as Echo State Networks (ESNs; Jaeger, 2001), Liquid State Machines (Maass et al., 2002), and the Decorrelation Backpropagation Rule (Steil, 2004). One defining characteristic of RC models is that all internal connections are adjusted by global or "macro-scale" parameters, significantly reducing the number of parameters that need to be trained. The relatively simplified structure and training requirements of RC make it an attractive architecture for large scale prediction because it enables rapid development, and could be useful in situations requiring online learning. More importantly though, we are motivated to use RC because past studies have repeatedly shown that it can emulate low dimensional chaotic systems while often outperforming more complex RNNs such as those with Long Short-Term Memory units (LSTMs) (e.g., Griffith et al., 2019; Lu et al., 2018; Pathak et al., 2018; Platt et al., 2022; Vlachas et al., 2020). Additionally, Penny et al. (2022) showed that RC can be successfully integrated with a number of data assimilation algorithms, either by generating samples for ensemble based methods like the Ensemble Kalman Filter, or by generating the tangent linear model necessary for 4D-Var. Finally, we note that Gauthier et al. (2021) proposed a further simplification to the RC architecture based on insights from Bollt (2021) that unifies versions of RC with nonlinear vector autoregression (NVAR). For a variety of chaotic systems, this architecture has shown excellent prediction skill even with low order, polynomial-based feature vectors (Barbosa & Gauthier, 2022; T.-C. Chen et al., 2022; Gauthier et al., 2021), despite requiring a much smaller hidden state and less training data. Considering all of these advancements, we are motivated to use these simple yet powerful single layer NVAR and ESN architectures to emulate turbulent geophysical fluid dynamics, and study how they are affected by temporal subsampling (see Section 3 for architecture details).

2. Surface Quasi-Geostrophic Turbulence

Our goal in this study is to emulate turbulent motions relevant to realistic geophysical fluid dynamics, while avoiding the complications associated with the data assimilation system used to produce reanalysis data sets, including observational noise and error covariance estimates, and the intricate multivariate interactions inside atmosphere or ocean GCMs. Therefore, we aim to emulate a numerical model for SQG turbulence (Blumen, 1978a; Held et al., 1995) as outlined by Tulloch and Smith (2009). The model is formulated to represent the nonlinear Eady problem (Eady, 1949), following Blumen (1978b). The model simulates turbulence on an f plane with uniform stratification and shear, bounded by rigid surfaces $H = 10$ km apart. The motion is determined entirely by temperature advection on the boundaries $z = \{0 \text{ km}, 10 \text{ km}\}$ as follows,

$$\frac{\partial \hat{\theta}}{\partial t} + \hat{J}\left(\hat{\psi}, \hat{\theta}\right) + ik\left(U\hat{\theta} + \hat{\psi}\frac{\partial \Theta}{\partial y}\right) = 0 \qquad z = 0, 10\,\text{km},$$

where $z = 0$ km is the surface layer of the atmosphere, and $z = 10$ km is approximately at the top of the troposphere. Here, hatted variables denote spectral components, $\hat{J}$ is the Jacobian in spectral space, and the temperature streamfunction is

$$\hat{\psi}(z,t) = \frac{H}{\mu \sinh \mu}\left[\cosh\left(\mu\frac{z}{H}\right)\hat{\theta}(H,t) - \cosh\left(\mu\frac{z-H}{H}\right)\hat{\theta}(0,t)\right],$$

with $\mu = |\mathbf{K}|NH/f$ as the nondimensional wavenumber. We note that this model produces an approximate spectrum of $|\mathbf{K}|^{-5/3}$ without any break (Figure 2), as is expected in Eady turbulence. For more details on this model, see Tulloch and Smith (2009).

Our model configuration is discretized in space with $N_x = N_y = 64$ and $N_z = 2$, uses a periodic boundary in both horizontal directions, and uses a timestep of $\Delta t = 5$ min. To generate data sets for the neural networks, we initialize the model with Gaussian i. i.d. noise and spinup for 360 days, which we define as one model year. The spinup

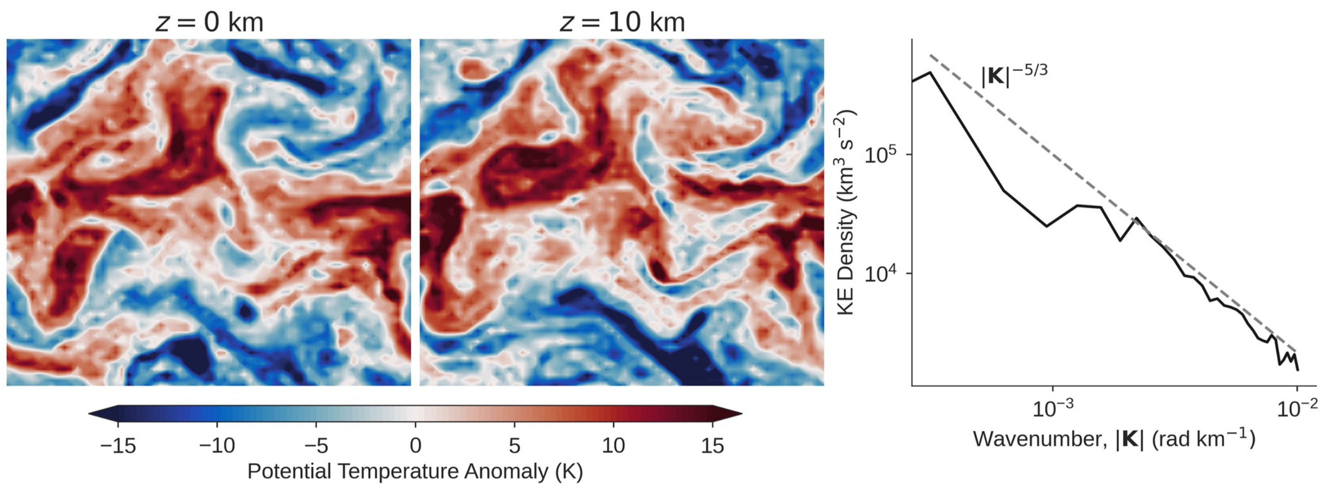

Figure 2. A reference snapshot from the SQG data set. The left and middle panels show snapshots of potential temperature anomaly at the surface and top-of-troposphere layers, respectively. The right panel shows the kinetic energy density spectrum associated with this snapshot (black line), compared to $|\mathbf{K}|^{-5/3}$ (dashed line).

period is discarded, and we then generate a 25 year-long data set that we partition into training (first 15 years), validation (next 5 years), and testing (final 5 years). For validation and testing, we randomly select without replacement 12 hr time windows from each respective data set.

3. Single Layer Autoregressive and Recurrent Neural Networks

Our goal is to develop an emulator that can reproduce the time evolution of a chaotic dynamical system, such that its future state can be predicted from an initial state estimate. Therefore we use the following generic, discrete-time equations for our recurrent and autoregressive models,

$$\begin{aligned}\mathbf{r}(n+1) &= f(\mathbf{r}(n), \mathbf{v}(n); \boldsymbol{\theta})\\ \hat{\mathbf{v}}(n+1) &= g(\mathbf{r}(n+1)),\end{aligned} \tag{1}$$

as by Goodfellow et al. (2016). Here $n \in \mathbb{Z}$ denotes a particular timestep $t = n\Delta\tau$, where $\Delta\tau = N_{\text{sub}}\Delta t$ is the time-step size of the neural network, which may be larger than $\Delta t = 5$ min, the step size of the original model described in Section 2. Here $\mathbf{v}(n) \in \mathbb{R}^{N_\mathbf{v}}$ is the state of the dynamical system and $\mathbf{r}(n) \in \mathbb{R}^{N_\mathbf{r}}$ is the hidden or internal state of the network, which is also referred to as the "reservoir" in RC or "feature vector" in NVAR. The generic function $f(\cdot)$ evolves this hidden state forward in time subject to the explicit influence of the current hidden and system states, as well as the macro-scale parameters $\boldsymbol{\theta}$. The output layer, $g(\cdot)$, or "readout" operation, maps the hidden state back to the original state space, giving an approximation of the target system.

During the training phase, $\mathbf{v}(n)$ is provided to the model at each timestep and the misfit between the approximation and data, $\hat{\mathbf{v}}(n+1) - \mathbf{v}(n+1)$, is used to train the weights in the output layer. After training, during the prediction phase, the network becomes an autonomous system:

$$\mathbf{r}(n+1) = f(\mathbf{r}(n), \hat{\mathbf{v}}(n); \boldsymbol{\theta}).$$

The neural network architectures that we use employ a common structure that is relevant to the readout operator and training procedure; we discuss these details in Section 3.1. Additionally, we employ a similar strategy to parallelize the architecture for high dimensional systems, and this is discussed in Section 3.2. Finally, the specific form of $f(\cdot)$ for the ESN and NVAR architectures is provided in Sections 3.3 and 3.4, respectively.

3.1. Linear Readout and Training

The neural networks that we use employ two simplifications relative to the generic form presented in Equation 1. First, any internal relationships encapsulated within $f(\cdot)$ are pre-defined by the macro-scale parameters, $\boldsymbol{\theta}$.

Therefore, no internal weights contained within $f(\cdot)$ are learned during the formal training process. Second, the readout operator is linear, such that

$$g(\mathbf{r}(n)) := \mathbf{W}_{\text{out}}\mathbf{r}(n),$$

where $\mathbf{W}_{\text{out}} \in \mathbb{R}^{N_{\mathbf{v}}\times N_{\mathbf{r}}}$ is a matrix. The result of these two assumptions is a cost function that is quadratic with respect to the elements of $\mathbf{W}_{\text{out}}$,

$$\mathcal{J}(\mathbf{W}_{\text{out}}) = \frac{1}{2N_{\text{train}}}\sum_{n=1}^{N_{\text{train}}} \|\mathbf{W}_{\text{out}}\mathbf{r}(n) - \mathbf{v}(n)\|_2^2 + \frac{\beta}{2}\|\mathbf{W}_{\text{out}}\|_{\text{F}}^2. \tag{2}$$

Here $\|\mathbf{A}\|_{\text{F}} := \sqrt{\text{Tr}(\mathbf{A}\mathbf{A}^T)}$ is the Frobenius norm, N_{train} is the number of time steps used for training, β is a Tikhonov regularization parameter (Tikhonov, 1963), chosen to improve numerical stability and prevent overfitting.

The hidden and target states can be expressed in matrix form by concatenating each time step "column-wise": $\mathbf{R} := (\mathbf{r}(1)\ \mathbf{r}(2)\ \cdots\ \mathbf{r}(N_{\text{train}}))$, and similarly, $\mathbf{V} := (\mathbf{v}(1)\ \mathbf{v}(2)\ \cdots\ \mathbf{v}(N_{\text{train}}))$. With this notation, the elements of $\mathbf{W}_{\text{out}}$ can be compactly written as the solution to the linear ridge regression problem

$$\mathbf{W}_{\text{out}} = \mathbf{V}\mathbf{R}^T\left(\frac{1}{N_{\text{train}}}\mathbf{R}\mathbf{R}^T + \beta\mathbf{I}\right)^{-1}, \tag{3}$$

although we do not form the inverse explicitly. We instead use the `solve` function from SciPy's linear algebra module (Virtanen et al., 2020), based on testing shown in Appendix C of Platt et al. (2022).

3.2. Parallelization Strategy

The model architectures that we use inherit the gridded structure of the target state being emulated, and often require hidden states that are $\mathcal{O}(10)$ to $\mathcal{O}(100)$ times larger than the target system dimension. Atmosphere and ocean GCMs typically propagate high dimensional state vectors, ranging from $\mathcal{O}(10^6)$ to $\mathcal{O}(10^9)$, so representing the system with a single hidden state would be intractable. Thus, we employ a parallelization strategy to distribute the target and hidden states across many semi-independent networks. Our strategy follows the algorithm introduced by Pathak et al. (2018), and follows a similar construction as Arcomano et al. (2020). We outline the procedure here and note an illustration of the process for the ESN architecture in Figure 3.

We subdivide the domain into N_g rectangular groups based on horizontal location, akin to typical domain decomposition techniques for atmosphere and ocean GCMs on structured grids. Each group contains $N_x^{\text{loc}} \times N_y^{\text{loc}}$ horizontal grid cells, and all N_z vertical grid cells at each horizontal location. The global state vector, $\mathbf{v}$, which consists of all state variables to be emulated at all grid cells, is partitioned into N_g local state vectors, $\mathbf{v}_k$. For example, Figure 3 shows a field $\mathbf{v}$ decomposed into nine groups, where each group is delineated by white lines. In our SQG predictions, we set $N_x^{\text{loc}} = N_y^{\text{loc}} = 8$, resulting in $N_g = 64$.

In order to facilitate interactions between nearby groups, each group has a designated overlap, or "halo," region which consists of N_o elements from its neighboring groups. The local group and overlapping points are illustrated in Figure 3 with a black box. The local state vectors, plus elements from the overlap region, are concatenated to form local input state vectors, $\mathbf{u}_k \in \mathbb{R}^{N_{\mathbf{u}}^{\text{loc}}}$. The result from the network is the local output state vector, $\mathbf{v}_k \in \mathbb{R}^{N_{\mathbf{v}}^{\text{loc}}}$, which is expanded to fill the target group as illustrated by the white box on the prediction shown in Figure 3. Here we set $N_o = 1$, so that $N_{\mathbf{u}}^{\text{loc}} = 200$ and $N_{\mathbf{v}}^{\text{loc}} = 128$, given that $N_x^{\text{loc}} = N_y^{\text{loc}} = 8$ and $N_z = 2$.

The local input vectors drive separate networks at each group, thereby generating distinct hidden states for each group as follows

$$\begin{aligned}\mathbf{r}_k(n+1) &= f(\mathbf{r}_k(n), \mathbf{u}_k(n); \boldsymbol{\theta})\\ \hat{\mathbf{v}}_k(n+1) &= \mathbf{W}_{\text{out}}^k\mathbf{r}_k(n+1).\end{aligned} \tag{4}$$

We make the assumption that the macro-scale parameters that determine internal connections within $f(\cdot)$ are globally fixed. Therefore, the only components that drive unique hidden states in each group are the local input vector $\mathbf{u}_k$ and the local readout matrix, $\mathbf{W}_{\text{out}}^k$.

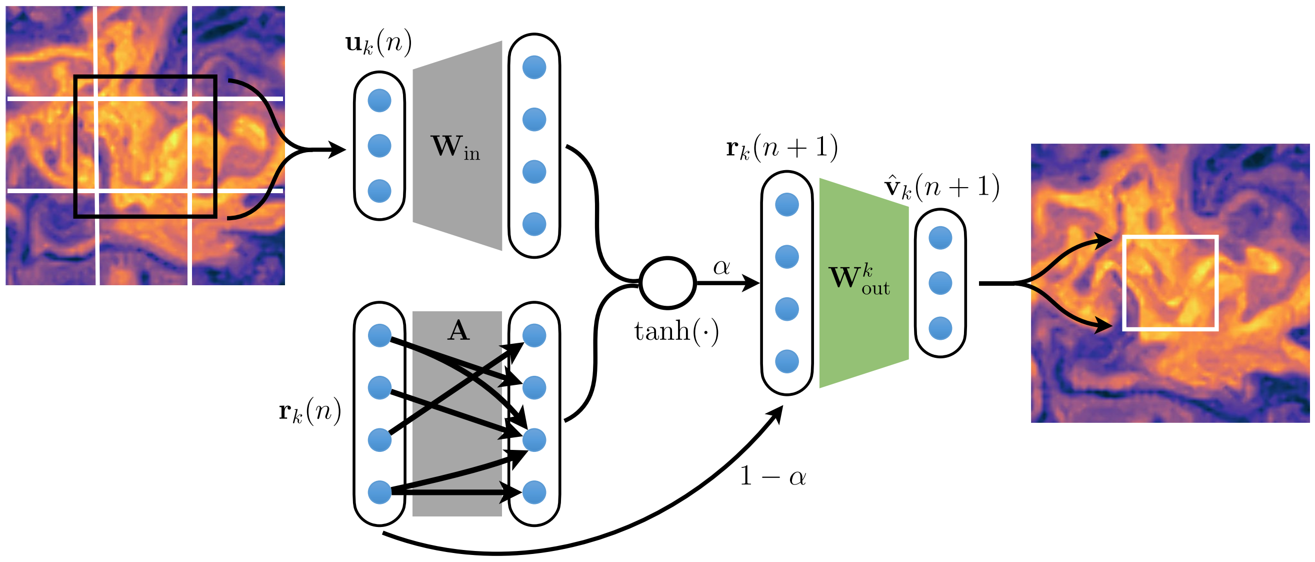

Figure 3. An illustration of the ESN architecture used, as it is applied to each local group throughout the domain. The domain is decomposed purely based on horizontal location, so the illustration shows a single horizontal slice, but note that each group contains all N_z vertical levels. In this example, there are nine groups delineated by the white lines on the 2D slice on the left. The black box denotes the group being operated on, which includes a region of width N_o that overlaps with neighboring groups. At timestep n, the group is flattened to make the input vector $\mathbf{u}_k(n)$, which is mapped into the ESN via $\mathbf{W}_{\text{in}}$. The output $\hat{\mathbf{v}}_k(n+1)$ is expanded to fill its position in the global domain. In ESNs, the matrices $\mathbf{A}$ and $\mathbf{W}_{\text{in}}$ (gray) are fixed, and only the readout matrix, $\mathbf{W}_{\text{out}}$ (green), is estimated from the training data.

During the training phase, each group acts completely independently from one another. Therefore, the training process is embarrassingly parallel and allows us to scale the problem to arbitrarily large state vectors across a distributed computing system, subject to resource constraints. During the prediction phase, neighboring elements must be passed between groups in order to fill each overlap region at each time step with the most accurate state estimate possible, to ensure spatial consistency across the domain.

3.3. Nonlinear Vector Autoregression Design

Following T.-C. Chen et al. (2022), we consider forming the hidden state by using polynomial combinations of the time-lagged input state. We explain this process with a simple example using a two variable system, $\mathbf{u}(n) = [u_0(n), u_1(n)]^T$, a maximum polynomial degree $p = 2$, and a generic maximum number of lagged states N_{lag}:

$$\begin{aligned}
\mathbf{r}_k(n+1) = [1,& \\
& u_0(n),\ u_1(n),\ u_0(n-1),\ u_1(n-1),\ \cdots\ u_0\left(n-N_{\text{lag}}\right),\ u_1\left(n-N_{\text{lag}}\right), \\
& u_0^2(n),\ u_1^2(n),\ u_0(n)u_1(n),\ u_0^2(n-1),\ \cdots\ u_1^2\left(n-N_{\text{lag}}\right) \\
& u_0(n)u_0(n-1),\ u_0(n)u_1(n-1),\ \cdots\ u_0\left(n-N_{\text{lag}}\right)u_1(n),\ \cdots] \\
\hat{\mathbf{v}}_k(n+1) = \mathbf{W}^k_{\text{out}}&\mathbf{r}_k(n+1).
\end{aligned} \tag{5}$$

Clearly, the size of the hidden state vector grows rapidly with p and N_{lag}, even for relatively low dimensional systems (see Supporting Information of T.-C. Chen et al., 2022, for explicit calculations). We therefore make a simplification to the generic polynomial NVAR model. That is, we only represent nonlinear interactions between points that lie within a given radius between one another, defined by the number of neighboring points, N_b. As a simple example, with $N_b = 1$ and $N_{\text{lag}} = 0$, the quadratic elements of a periodic, four variable system would be

$$u_0^2,\ u_1^2,\ u_2^2,\ u_3^2,\ u_0u_1,\ u_0u_3,\ u_1u_2,\ u_2u_3$$

ignoring "non-local" interactions such as u_0u_2. In order to make this parameter consistent with the overlap region in the parallelization scheme (Section 3.2), we set $N_b = N_o = 1$. Note, however, that we do model "non-local" linear interactions, up to the number of grid cells in each local group, that is, containing $\left(N_x^{\text{loc}} + 2N_o\right) \times \left(N_y^{\text{loc}} + 2N_o\right) \times N_z$ points.

All of the remaining macro-scale parameters that determine the NVAR performance are

$$\boldsymbol{\theta}_{NVAR} = \left\{p, N_{\text{lag}}, \beta\right\}.$$

By using the preconditioning scheme introduced by T.-C. Chen et al. (2022), we found results to be insensitive to the Tikhonov parameter β, and so we fix this to $\beta = 10^{-4}$. As noted earlier, we set $p = 2$. Our assumption behind this decision is that the NVAR model will be able to learn local quantities like gradients and fluxes between neighboring grid cells. Based on the results from T.-C. Chen et al. (2022), the NVAR model should then be able to use this information to construct arbitrarily complex time stepping schemes as a function of N_{lag}. Because of its explicit nature, we manually vary N_{lag} to understand how memory impacts NVAR prediction skill.

3.4. Echo State Network Design

Our ESN architecture is illustrated in Figure 3, and is defined as follows

$$\begin{aligned} \mathbf{r}_k(n+1) &= (1-\alpha)\mathbf{r}_k(n) + \alpha \tanh(\mathbf{A}\mathbf{r}_k(n) + \mathbf{W}_{\text{in}}\mathbf{u}_k(n) + \mathbf{b}) \\ \hat{\mathbf{v}}_k(n+1) &= \mathbf{W}^k_{\text{out}}\mathbf{r}_k(n+1). \end{aligned} \tag{6}$$

Here $\alpha \in [0, 1]$ is a leak parameter, $\mathbf{A} \in \mathbb{R}^{N_\mathbf{r} \times N_\mathbf{r}}$ is an adjacency matrix that determines the internal connections between the nodes of the hidden state, $\mathbf{W}_{\text{in}} \in \mathbb{R}^{N_\mathbf{r} \times N_\mathbf{u}}$ maps the input vector into the higher dimensional hidden state, and $\mathbf{b} \in \mathbb{R}^{N_\mathbf{r}}$ is the bias vector with elements $b_i \sim \mathcal{U}(-\sigma_b, \sigma_b)$. Unless otherwise specified, each ESN model uses a hidden layer width of $N_\mathbf{r} = 6,000$. Finally, we note that ESNs require a spinup period before generating predictions, so we specify a 10 day spinup period for all validation and testing samples.

Two scalar parameters, ρ and σ, are used to control the scaling of the adjacency and input matrices, respectively. These parameters have a dramatic influence on ESN prediction skill, since their values influence the network's memory and stability (Hermans & Schrauwen, 2010; Lukoševičius, 2012). Here we first normalize the matrices by their largest singular value, and then apply the scaling parameters as follows

$$\mathbf{A} := \frac{\rho}{\sigma_{\max}\left(\hat{\mathbf{A}}\right)}\hat{\mathbf{A}} \qquad \mathbf{W}_{\text{in}} := \frac{\sigma}{\sigma_{\max}\left(\hat{\mathbf{W}}_{\text{in}}\right)}\hat{\mathbf{W}}_{\text{in}}$$

where the elements of $\hat{\mathbf{W}}_{\text{in}}$ are initialized with elements $\hat{w}_{i,j} \sim \mathcal{U}(-1, 1)$. The initial adjacency matrix is generated similarly, except that the indices i, j are randomly chosen such that $\hat{\mathbf{A}}$ attains a specified sparsity. Here we set the matrix sparsity to $1 - \kappa/N_\mathbf{r}$, with $\kappa = 6$, following the success of very sparsely connected adjacency matrices as shown by Griffith et al. (2019). By first normalizing the matrices by the largest singular value, the parameters ρ and σ re-scale the induced 2-norm of the matrix. This normalization is not standard in the ESN literature, but we found that it helped improve prediction skill. We provide further discussion of this process in Appendix A.

In summary, the macro-scale parameters that determine the overall characteristics of the ESN are

$$\boldsymbol{\theta}_{ESN} = \{\rho, \sigma, \sigma_b, \alpha, \beta\}, \tag{7}$$

which are globally fixed for all groups. Due to the high sensitivity of ESN prediction skill to these parameter values, we follow the general optimization framework described by Platt et al. (2022) to determine approximately optimal values. We use the Bayesian Optimization algorithm outlined by Jones et al. (1998) and implemented by Bouhlel et al. (2019) to tune them. This process is discussed in Section 5. However, we first focus on prediction skill using the NVAR architecture in Section 4.

4. Nonlinear Vector Autoregression Prediction Skill

In this section we show the prediction skill of the polynomial based NVAR architecture described in Section 3.3. Note that we show the prediction skill of the ESN architecture in Section 5. To quantitatively evaluate each forecast, we compute the normalized root-mean-square error (NRMSE)

$$\text{NRMSE}(n) = \sqrt{\frac{1}{N_\mathbf{v}}\sum_{i=1}^{N_\mathbf{v}}\left(\frac{\hat{v}_i(n) - v_i(n)}{\text{SD}}\right)^2}, \tag{8}$$

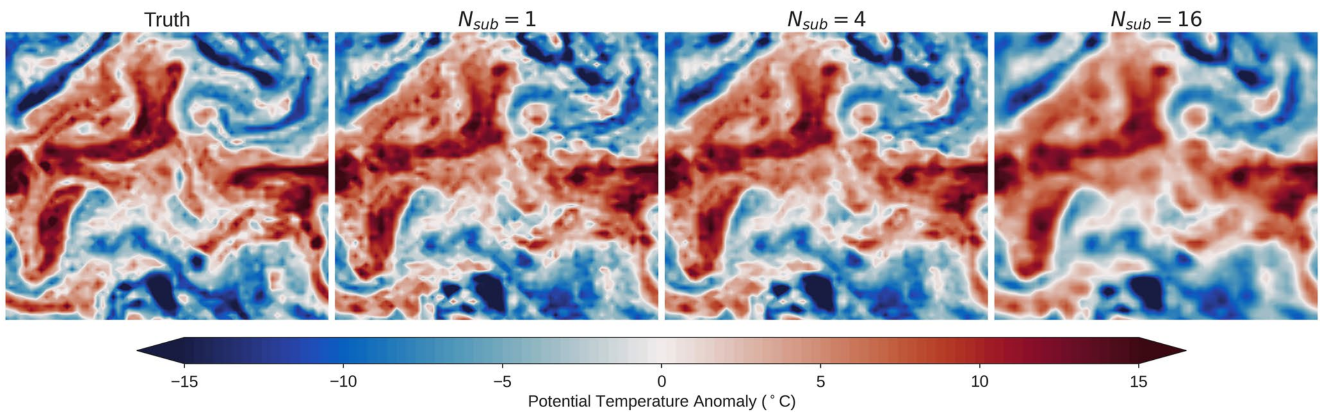

Figure 4. One sample NVAR prediction from the test data set for $N_{\text{sub}} = 1, 4, 16$; shown in the second, third, and fourth panels at a lead time of 4 hr. The corresponding truth is shown in the far left panel. As the temporal subsampling factor is increased, the small spatial scale features are diminished and predictions become blurrier. Here $N_{\text{lag}} = 1$ and only the surface level is shown.

which is averaged over each spatial dimension, succinctly represented as a summation over $N_{\mathbf{v}}$, and normalized by the standard deviation, SD, computed from the true trajectory over time and all spatial dimensions. Additionally, we compute the relative error in terms of the kinetic energy (KE) density spectrum,

$$\text{KE Relative Error}(n, k) = \frac{\hat{E}(n, k) - E(n, k)}{|E(n, k)|}, \tag{9}$$

where $E(n, k)$ and $\hat{E}(n, k)$ are the true and predicted KE density coefficients for each timestep n and wavenumber k, respectively (e.g., as in the right panel of Figure 2). Note that $|\cdot|$ denotes the absolute value operation, and we retain the sign of the error in order to show a sense of the spectral error in each prediction.

We compute these quantities based on 50 twelve-hour predictions initialized from a random set of initial conditions taken from an unseen test data set. To compactly visualize the skill over all samples, each lineplot in the following subsections shows a sample-average value with a solid line, and the 99% confidence interval with shading. We note that in some cases the model trajectory becomes unstable to the point that infinite values are produced. In the event that any single sample from a distribution has produced infinity, we take the more conservative approach and cut off any statistical averaging or confidence interval computation at that point in time and carry it no further. Therefore, some plots of NRMSE over time do not extend over the full 12 hr window, even though some sample trajectories are still valid, for example, Figure 5 (left).

4.1. Temporal Subsampling

Figure 4 shows a qualitative comparison of NVAR predictions as a function of N_{sub}, that is, how frequently the training data are sampled and the model makes predictions. For this figure, we set $N_{\text{lag}} = 1$, and note that both the NRMSE and a snapshot of the KE density relative error corresponding to this configuration are shown in Figure 5.

At the model timestep ($\Delta\tau = \Delta t = 5$ min; $N_{\text{sub}} = 1$), the NVAR predictions are qualitatively similar to the truth for short forecast lead times. That is, the NRMSE is near 0, and many of the small scale features that exist in the truth are also evident in the predictions. However, at longer lead times the predictions become unstable. NRMSE spikes rapidly at about 4 hr after numerical instabilities are generated, which causes the NVAR model to produce physically unrealistic results. For reference, Figure S1 shows a view of what these numerical instabilities look like at their onset.

As the temporal resolution of the data is reduced, that is, as N_{sub} increases, the predictions are generally stable for a longer period of time. Figure 5 shows that for $N_{\text{sub}} = 4$, predictions are stable for roughly 6 hr, and for $N_{\text{sub}} = 16$ no predictions generate numerical instabilities over the 12 hr window. However, this stability comes with a cost: as the temporal resolution is reduced, the model's representation of small scale features diminishes as these

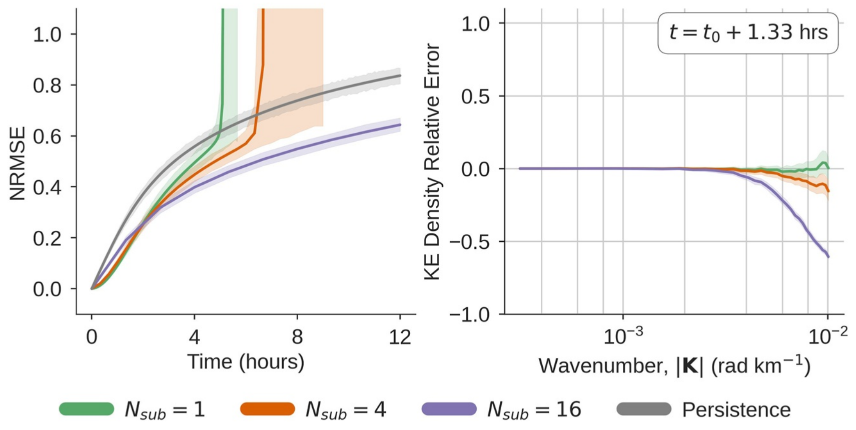

Figure 5. NRMSE (Equation 8; left) and KE density relative error (Equation 9; right) indicating prediction skill of the NVAR architecture using 50 samples from the test data set. Solid lines indicate averages and shading indicates 99% confidence interval. Here $N_{\text{lag}} = 1$, and the gray line indicates prediction skill of a persistent forecast, that is, where the initial condition does not change.

features become more blurry or smoothed. This blurring effect is apparent in Figure 4, where the prediction is qualitatively more blurry as N_{sub} increases in each panel from left to right.

This smoothing behavior is captured quantitatively in the right panel of Figure 5, which shows the KE relative error as in Equation 11. Here, we show the KE relative error after only 1.33 hr to show the behavior before instabilities dominate the $N_{\text{sub}} = 1$ predictions. The plot indicates the degree of spectral bias in each solution, which is largest at the smaller spatial scales, corresponding to higher wave numbers.

At $N_{\text{sub}} = 1$ there is a small positive bias at the smallest resolved spatial scales, indicating that this is when numerical instabilities are starting to generate. The subsampled runs, $N_{\text{sub}} = \{4, 16\}$, show a negative bias, which corresponds to a dampened energy spectrum at the scales that are not resolved in the qualitatively smooth predictions shown in Figure 4. This negative bias is clearly larger with higher subsampling, or reduced temporal resolution, suggesting that as the data are subsampled, the network becomes incapable of tracking the small scale dynamics. The result is an averaged view of what may be occurring in between each time stamp.

4.2. Prediction Skill as a Function of Memory

A key feature of RNNs and autoregressive models is that they retain memory of previous system states. Given the explicit nature of the NVAR architecture, we explore the effect of adding memory by increasing N_{lag}, the number of lagged states used to create the feature vector. We first summarize how memory impacts prediction skill in Figure 6, which shows the NRMSE as a function of N_{lag} (colors) for each subsampling factor $N_{\text{sub}} = \{1, 4, 16\}$ (panels). For any value of N_{sub}, adding memory (increasing N_{lag}) reduces the short term error. However, adding memory also tends to increase error by the end of the forecast, often leading to the development of numerical instabilities and an incoherent solution. Similarly, for any fixed value of N_{lag}, increasing the temporal resolution (decreasing N_{sub}) shows the same behavior.

To shed some light on how this additional memory impacts the solution, we show the KE relative error for the case of $N_{\text{sub}} = 16$ as a function of time (panels) and N_{lag} (colors) in Figure 7. For about the first 4 hr, increasing memory improves prediction skill at all spatial scales. However, beyond this point, the overall NRMSE grows rapidly, the improvement at small scales ($|\mathbf{K}| > 4 \cdot 10^{-3}$ rad km^{-1}) is more muted, and error is propagated rapidly into the larger spatial scales.

We surmise that adding memory degrades the long term prediction skill in the quadratic NVAR because the relationship between points further back in history are governed by higher order nonlinear interactions that are incorrectly represented by the simple local-quadratic relation that is used here. As more terms are added that are incorrectly represented, the model becomes more and more unstable. We make this supposition based on the fact that despite theoretical similarities between NVAR and ESNs as highlighted by Bollt (2021), we attain stable predictions using an ESN architecture with a hyperbolic tangent activation function in Section 5.

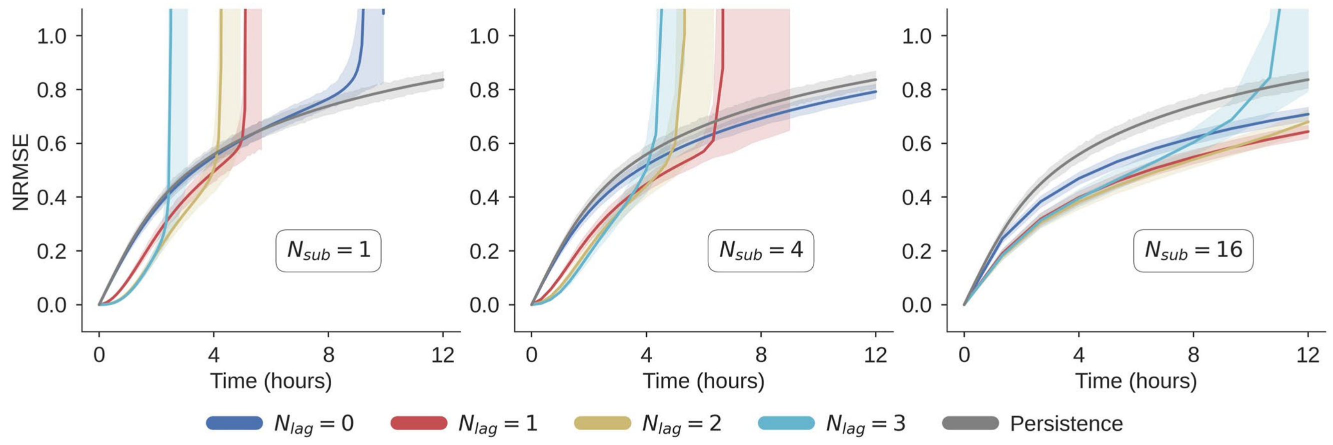

Figure 6. NRMSE computed using NVAR at various temporal resolutions (N_{sub}; columns) and with variable memory capacities (N_{lag}; colors). Decreasing the subsampling factor shows a similar effect as adding memory: error is at first reduced, but tends to produce more unstable forecasts.

The question for the NVAR architecture is therefore how to retain the short term benefit of added memory capacity throughout the forecast horizon while maintaining a stable trajectory. While it may seem natural to explore higher order polynomials to properly represent this history, we do not explore this further because the size of the feature vector grows dramatically with the polynomial order (T.-C. Chen et al., 2022). Another option would be to explore entirely different basis functions. While this could be a potential option for future work, we note the findings of Zhang and Cornelius (2022), who show the extreme sensitivity of NVAR to the form of nonlinearity imposed. Given that it is an entirely open question on how to represent the smallest scales of geophysical turbulence, we do not explore other basis functions, and instead turn to the more general ESN architecture.

5. Echo State Network Prediction Skill

In this section we show the prediction skill of the more general ESN architecture outlined in Section 3.4. Here we use similar metrics as in Section 4 to evaluate the ESN skill, except that we show time averaged quantitative metrics because all of the ESN predictions are stable for the full twelve-hour forecast horizon. That is, when shown as a single distribution rather than a time series, NRMSE is reported as

$$\text{NRMSE} = \sqrt{\frac{1}{N_{\text{time}} N_{\text{v}}} \sum_{n=1}^{N_{\text{time}}} \sum_{i=1}^{N_{\text{v}}} \left(\frac{\hat{v}_i(n) - v_i(n)}{\text{SD}} \right)^2}, \tag{10}$$

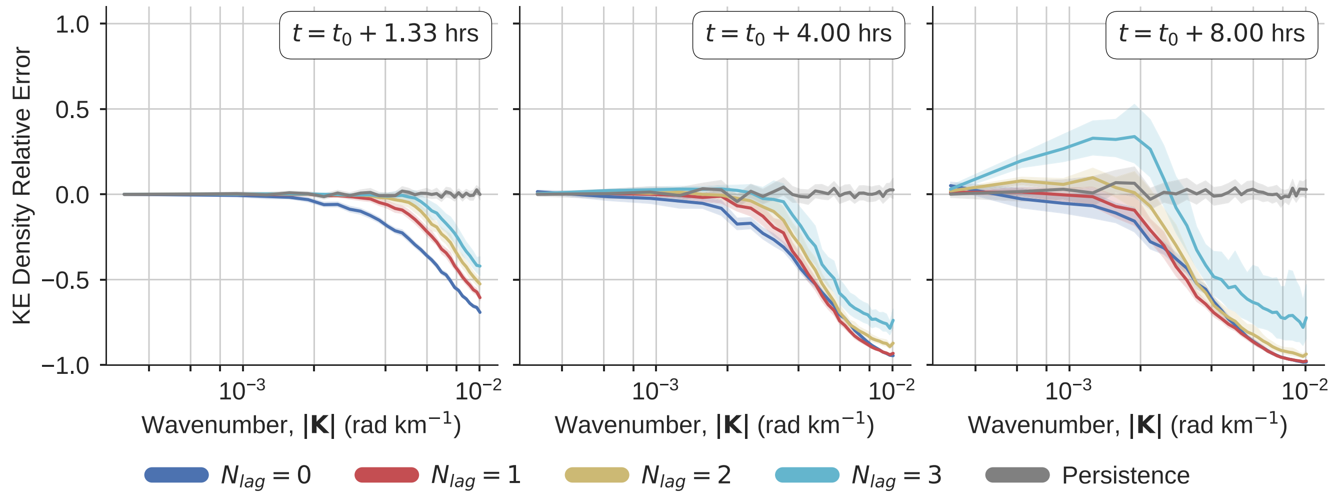

Figure 7. Kinetic energy density relative error with $N_{\text{sub}} = 16$ at various timesteps (columns) and memory capacity (N_{lag}; colors). Increasing memory at first reduces error at all spatial scales, but later on the error propagates more readily into the large scale.

where N_{time} consists of the number of timesteps in the trajectory. In order to characterize spectral error, we show the KE relative error as in Section 4. Additionally, we show the NRMSE in terms of the KE density spectrum as follows

$$\text{KE\_NRMSE} = \sqrt{\frac{1}{N_{\text{time}} N_K} \sum_{n=1}^{N_{\text{time}}} \sum_{k=1}^{N_K} \left(\frac{\hat{E}(n,k) - E(n,k)}{\text{SD}(k)} \right)^2}, \tag{11}$$

where N_K is the number of spectral coefficients and $\text{SD}(k)$ is the temporal standard deviation of each spectral coefficient throughout the test trajectory. As in Section 4, all distributions and lineplots indicate prediction skill from 50 randomly selected initial conditions from an unseen test data set.

5.1. Soft Constraints on Spectral Error

It is well known that ESN prediction skill is highly dependent on the global or "macro-scale" parameters noted in Equation 7 ($\boldsymbol{\theta}_{ESN}$, e.g., Platt et al., 2022; Lukoševičius, 2012). Following the success of previous studies in using Bayesian Optimization methods to systematically tune these parameters (Griffith et al., 2019; Penny et al., 2022; Platt et al., 2022), we use the Bayesian Optimization algorithm outlined by Jones et al. (1998) and implemented by Bouhlel et al. (2019) to find optimal parameter values.

More recently, Platt et al. (2023) showed that constraining these macro-scale parameters using global invariant properties of the underlying system leads the optimization algorithm to select parameters that generalize well to unseen test data. In that work, the authors were successful in using the largest positive Lyapunov exponent, and to a lesser extent the fractal dimension of the system. Because of the focus on resolved scales in this work, we take a similar approach, but test the effect of constraining the ESN to the KE density spectral coefficients. Specifically, we implement the following two-stage training process. At each step, the macro-scale parameters, $\boldsymbol{\theta}_{ESN}$, are fixed, and the "micro-scale" parameters $\mathbf{W}_{\text{out}}$ are obtained by minimizing Equation 2. This readout matrix is then used to make forecasts from randomly selected initial conditions from a validation data set. The skill of each of these forecasts is captured by the macro-scale cost function

$$\mathcal{J}_{\text{macro}}(\boldsymbol{\theta}_{ESN}) = \frac{1}{N_{\text{macro}}} \sum_{j=1}^{N_{\text{macro}}} \{\text{NRMSE}(j) + \gamma \text{KE\_NRMSE}(j)\}, \tag{12}$$

where NRMSE and KE_NRMSE are defined in Equations 10 and 11, N_{macro} is the number of forecasts used in the validation set, and γ is a hyperparameter that determines how much to penalize deviations from the true KE density spectrum. The value of $\mathcal{J}_{\text{macro}}$ is then used within the Bayesian Optimization algorithm, which reiterates the whole optimization process with new values for $\boldsymbol{\theta}_{ESN}$ until an optimal value is found or the maximum number of iterations is reached. Here, we use $N_{\text{macro}} = 10$, initialize the optimization with 20 randomly sampled points in the 5 dimensional parameter space, and run for 10 iterations. Note that we run this optimization procedure for each unique ESN configuration throughout Section 5 (i.e., for each N_{sub} and each γ value).

Figure 8 shows a qualitative view of how penalizing the KE density impacts ESN prediction skill when it operates at the original timestep of the SQG model (i.e., $N_{\text{sub}} = 1$). At $\gamma = 0$, the ESN parameters are selected based on NRMSE alone, and the prediction is relatively blurry. However, as γ increases to 10^{-1}, the prediction becomes sharper as the small scale features are better resolved.

Figure 9 gives a quantitative view of how the KE density penalty changes ESN prediction skill, once again with $N_{\text{sub}} = 1$. The first two panels show that there is a clear tradeoff between NRMSE and KE error: as γ increases the NRMSE increases but the spectral representation improves. The final panel in Figure 9 shows the spatial scales at which the spectral error manifests in these different solutions. When $\gamma = 0$, the macro-scale parameters are chosen to minimize NRMSE, leading to blurry predictions and a dampened spectrum at the higher wavenumbers, especially for $|\mathbf{K}| > 2 \cdot 10^{-3}$ rad km^{-1}. We note that Lam et al. (2022) report the same behavior when using a cost function that is purely based on mean-squared error. On the other hand, when $\gamma = 10^{-1}$, the global parameters are chosen to minimize both NRMSE and KE density error, where the latter treats all spatial scales equally. In this case, KE relative error is reduced by more than a factor of two and the spectral bias at higher wavenumbers is much more muted.

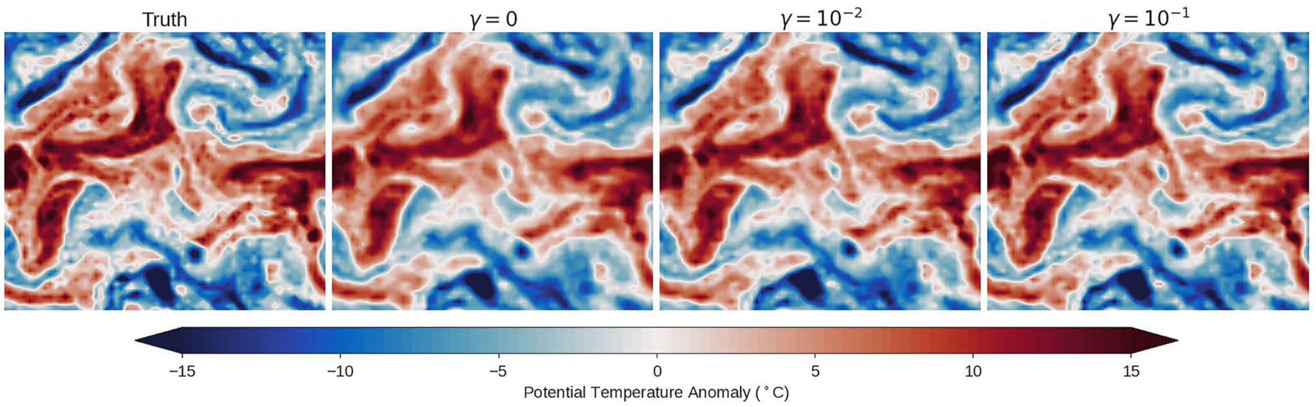

Figure 8. One sample prediction from the test data set, where each panel shows potential temperature in the truth (left) and subsequently for ESN predictions with parameters optimized using $\gamma = \{0, 10^{-2}, 10^{-1}\}$ in Equation 12. Each panel shows the prediction at a forecast lead time of 4 hr, using the same initial conditions as in Figure 4. As γ increases from left to right, the prediction becomes sharper (i.e., less blurry). Here, the ESN is evaluated at the SQG model timestep, that is, $N_{\text{sub}} = 1$.

Of course, the tradeoff for the reduced spectral error is larger NRMSE, resulting from slight mismatches in the position of small scale features in the forecast. However, our purpose is to generate forecasts that are as representative of the training data as possible. Overly smoothed forecasts are not desirable, because this translates to losing local extreme values, which are of practical importance in weather and climate. Additionally, a key aspect of ensemble forecasting is that the truth remains a plausible member of the ensemble (Kalnay et al., 2006). Therefore, representing the small scale processes, at least to some degree, will be critical for integrating an emulator into an ensemble based prediction system.

Finally, we note that using a cost function with only KE_NRMSE produced inconsistent results. Therefore, we consider it important to keep the NRMSE term in the cost function, as this prioritizes the position of small scale features. That is, it helps maintain phase information. Additionally, we note that there is some irreducible high wavenumber error, which is most clearly seen by comparing the prediction skill to a persistent forecast. While the sample median NRMSE for each γ value beats persistence, the KE_NRMSE is more than double, due to this error at the small spatial scales. Ideally, our forecasts would beat persistence in both of these metrics, but obtaining the "realism" in the small spatial scales necessary to dramatically reduce this spectral error should be addressed in future work.

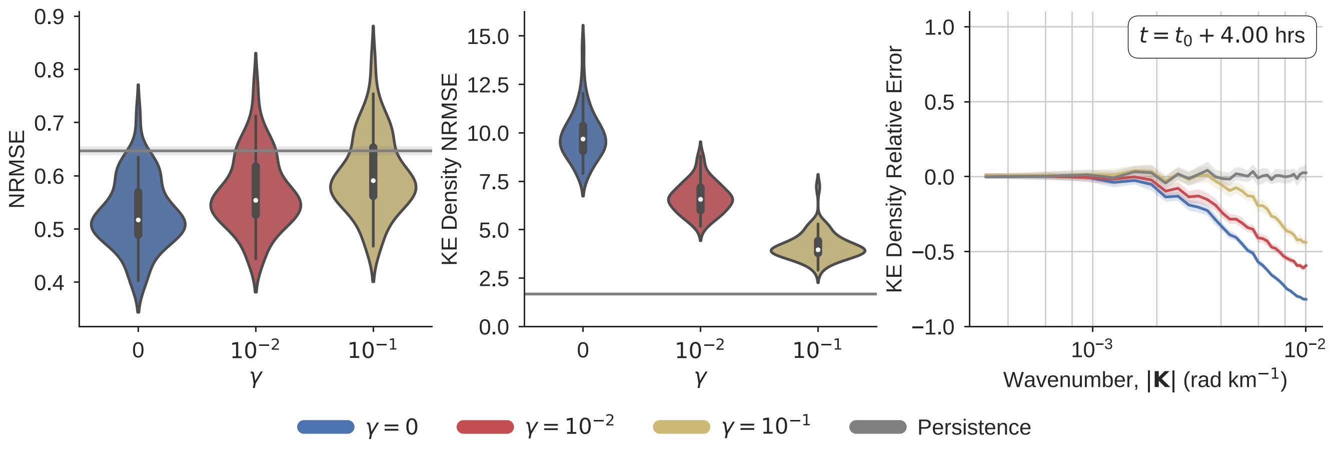

Figure 9. Quantitative comparison of ESN predictions at $N_{\text{sub}} = 1$ with macro-scale parameters chosen using different values of γ in Equation 12. NRMSE (Equation 10; left), KE_NRMSE (Equation 11; middle), and KE relative error (Equation 9; right) highlight the tradeoff between minimizing NRMSE and spectral error: as γ increases spectral error is reduced, but NRMSE increases. Note that the KE relative error is shown at 4 hr to provide direct comparison to the snapshots in Figure 8. In each plot, the solid gray line indicates the median skill of a persistent forecast.

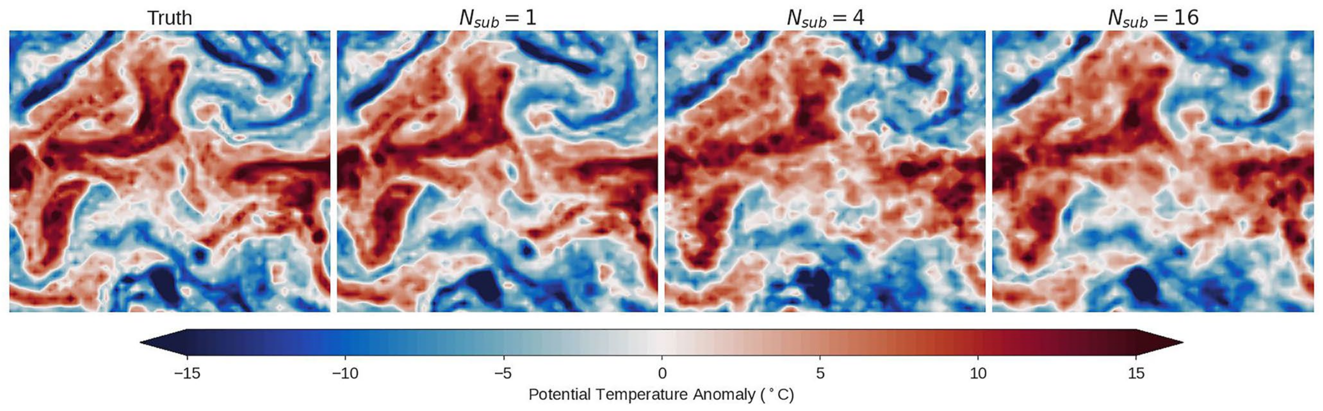

Figure 10. One sample prediction from the test data set, exactly as in Figure 8, except here $\gamma = 10^{-1}$ is fixed, and the temporal subsampling factor is varied: $N_{\text{sub}} = \{1, 4, 16\}$. As the temporal subsampling factor increases, the small spatial scale features are lost and the prediction becomes blurrier.

5.2. Temporal Subsampling

The NVAR predictions shown in Section 4.1 indicate that subsampling the training data systematically increases error at small spatial scales. However, the architecture was not specifically designed or constrained to have a good spectral representation of the underlying dynamics. On the other hand, the previous section (Section 5.1) showed that the spectral bias at high wavenumbers can be reduced by optimizing the global ESN parameters to the true KE density spectrum. Given these two results, we explore the following question: does temporal subsampling still increase spectral bias in the more general ESN framework, even when parameters are chosen to minimize this bias?

Figures 10 and 11 show that even when the macro-scale parameters are chosen to prioritize the KE density representation (i.e., $\gamma = 10^{-1}$ is fixed), temporal subsampling does lead to an apparently inescapable spectral bias. This effect is shown qualitatively in Figure 10, where the predictions become smoother as the temporal subsampling factor, N_{sub}, increases. The effect is similar to what was seen with NVAR except the blurring effect is less pronounced. Quantitatively, Figure 11b shows that as N_{sub} increases, error in KE density spectrum generally increases, while panel (c) shows that this KE error is concentrated in the small spatial scales, $|\mathbf{K}| > 2 \cdot 10^{-3}$ rad km^{-1}. We note that the degree of spectral bias at $N_{\text{sub}} = 16$ is smaller than what was achieved with NVAR for the same N_{sub} value, cf. Figure 7, indicating that the optimization was successful in reducing the spectral bias.

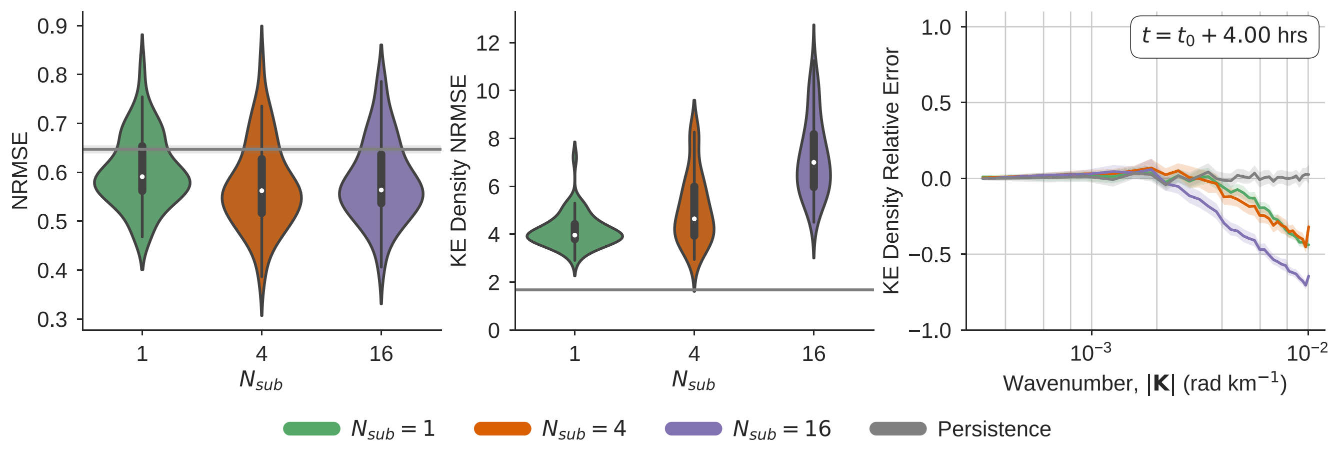

Figure 11. Quantitative comparison of ESN predictions, showing NRMSE (left), KE_NRMSE (middle), and KE relative error (right), exactly as in Figure 9, except here $\gamma = 10^{-1}$ is fixed, and the temporal subsampling factor is varied: $N_{\text{sub}} = \{1, 4, 16\}$. As the temporal subsampling factor increases, spectral errors increase. In each plot, the solid gray line indicates the median skill of a persistent forecast.

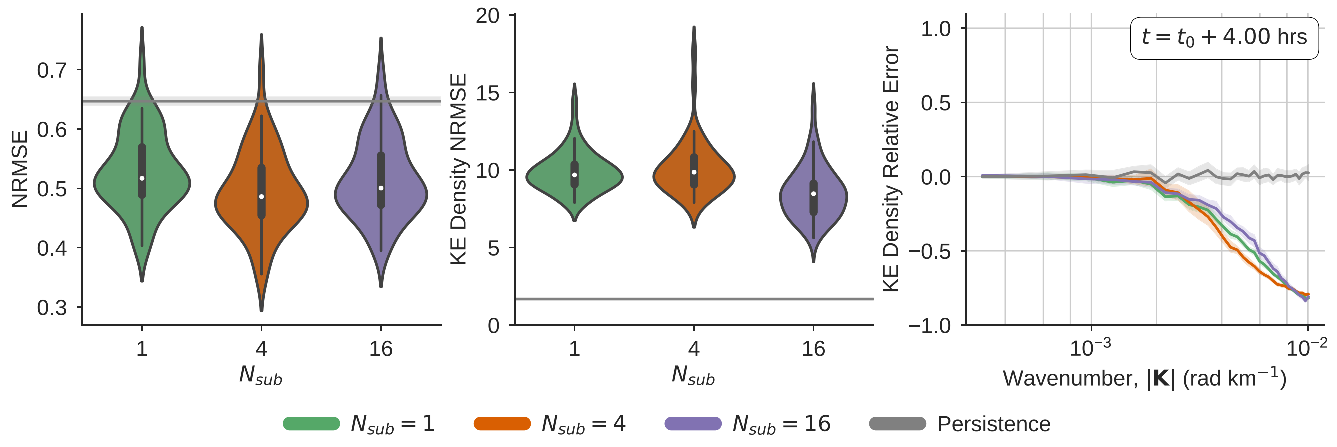

Figure 12. Same as Figure 11, except here $\gamma = 0$, indicating that only NRMSE is penalized in the cost function. The error is relatively similar, indicating that NRMSE alone is a suboptimal penalty for model selection. In each plot, the solid gray line indicates the median skill of a persistent forecast.

Interestingly, there is little difference between NRMSE obtained by the ESNs at different N_{sub} values. Additionally, Figure 12 shows that there is little difference in both NRMSE and KE_NRMSE when $\gamma = 0$, that is, when NRMSE is the only criterion for parameter selection. This result shows that NRMSE alone is not a good criterion for model selection, given that we have shown success in reducing spectral errors by prioritizing the spectrum appropriately.

5.3. Impact of the Hidden Layer Dimension

The dimension of the hidden layer, $N_{\mathbf{r}}$, also known as the reservoir size, determines the memory capacity available to the ESN (Jaeger, 2001; Lukoševičius, 2012). For systems with high dimensional input signals, it is crucial to use a sufficiently large hidden layer to afford the memory capacity necessary for accurate predictions (Hermans & Schrauwen, 2010). In all of the preceding sections we fixed $N_{\mathbf{r}} = 6,000$ for each local group, where for reference each local group has an input dimension of $N_{\mathbf{u}}^{\text{loc}} = 200$ and an output dimension of $N_{\mathbf{v}}^{\text{loc}} = 128$. Here, we briefly address the effect of doubling the hidden layer dimension, while keeping the input and output dimensions constant, in order to test how sensitive our conclusions are on this crucial hyperparameter. Due to the computational expense of the parameter optimization discussed in Section 5.1, we only perform this experiment for $N_{\text{sub}} = 16$.

The impact of doubling $N_{\mathbf{r}}$ on prediction skill is shown in Figure 13, where for the sake of brevity we only show results for the case when $\gamma = 10^{-1}$ in Equation 12. The left panel shows that the larger hidden layer actually increases the NRMSE slightly. However, the middle and right panels show that this increase is due to the improved spectral representation. The improvement in KE_NRMSE is nearly proportional to the improvement achieved by increasing the temporal resolution of the data. That is, doubling the hidden layer width reduces the average KE_NRMSE by 14%, while increasing the temporal resolution of the data by a factor of 4 reduces the KE_NRMSE by 30%. These results indicate a potential brute force approach to overcoming the subsampling related spectral errors. However, the larger hidden layer dimension has to be constrained with enough training data, and requires more computational resources.

5.4. Impact of Training Data Set Size

In all of the preceding experiments, the length of training time was fixed to 15 years, meaning that there are fewer training samples when the data are subsampled, that is, as N_{sub} grows. Specifically, 15 years of data at an original model timestep of 5 min means that there are approximately $1.6 \cdot 10^6$, $3.9 \cdot 10^5$, and $9.72 \cdot 10^4$ samples for each case previously shown: $N_{\text{sub}} = 1$, 4, and 16, respectively. Here, we show that even when the number of training samples is fixed, the subsampling related spectral errors are still present.

Figure 14 shows the prediction skill in terms of NRMSE and spectral errors when the number of training samples is fixed to $9.72 \cdot 10^4$. With this number of samples, the training data is exactly the same for $N_{\text{sub}} = 16$, but only

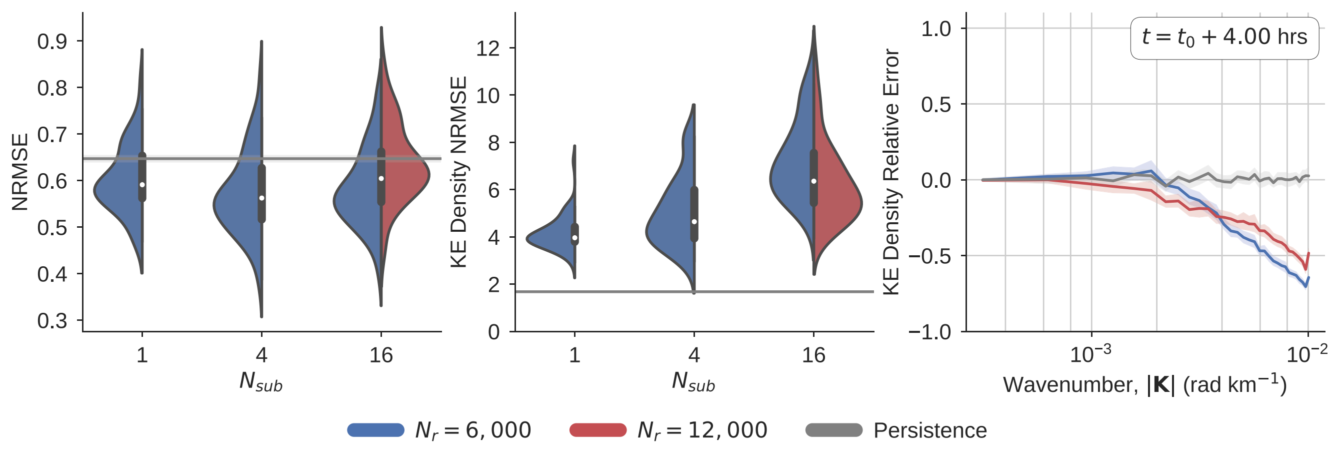

Figure 13. The impact of doubling the hidden layer dimension from $N_r = 6{,}000$ to $N_r = 12{,}000$ on NRMSE (left), KE_NRMSE (middle), and KE relative error (right). Increasing the hidden layer dimension is relatively proportional to reducing the temporal subsampling factor, indicating a potential brute force approach to reducing the subsampling related spectral errors. Here $\gamma = 10^{-1}$, and the solid gray line indicates the median skill of a persistent forecast.

spans 3.75 and 0.94 years for $N_{sub} = 4$ and $N_{sub} = 1$, respectively. However, we see the same general trend as before: subsampling the data improves NRMSE slightly but increases the KE_NRMSE. As before, the spectral error is largest in the higher wavenumbers, $|\mathbf{K}| > 2 \cdot 10^{-3}$ rad km^{-1}. We note that the difference in performance between $N_{sub} = 4$ and $N_{sub} = 16$ is marginal. The only notable difference between these two cases is that the ESN is less consistent, that is, the KE_NRMSE distribution is broader, when $N_{sub} = 16$. However, it is clear that spectral error is lowest when the data are not subsampled at all, even though less than a year of data is used. This result indicates that there could be a benefit to training a RNN on a relatively shorter model trajectory that is untouched, rather than a longer data set that is subsampled in time.

6. Discussion

Weather and climate forecasting necessitates the integration of expensive numerical models to make accurate predictions and projections. The computational cost of these models often results in tradeoffs, where practitioners must balance the spatial resolution of their model with other factors, such as the number of integrated model components or the ensemble size that can be afforded in the system. Model emulation or surrogate modeling aims to enable such predictions by emulating the dynamical system with adequate accuracy at a much lower computational expense. In this study, our primary interest was to shed light on the spatial scales that can be resolved by single layer autoregressive and recurrent neural network emulators in order to better understand the effective resolution that could be achieved in weather and climate applications. We used two relatively simple,

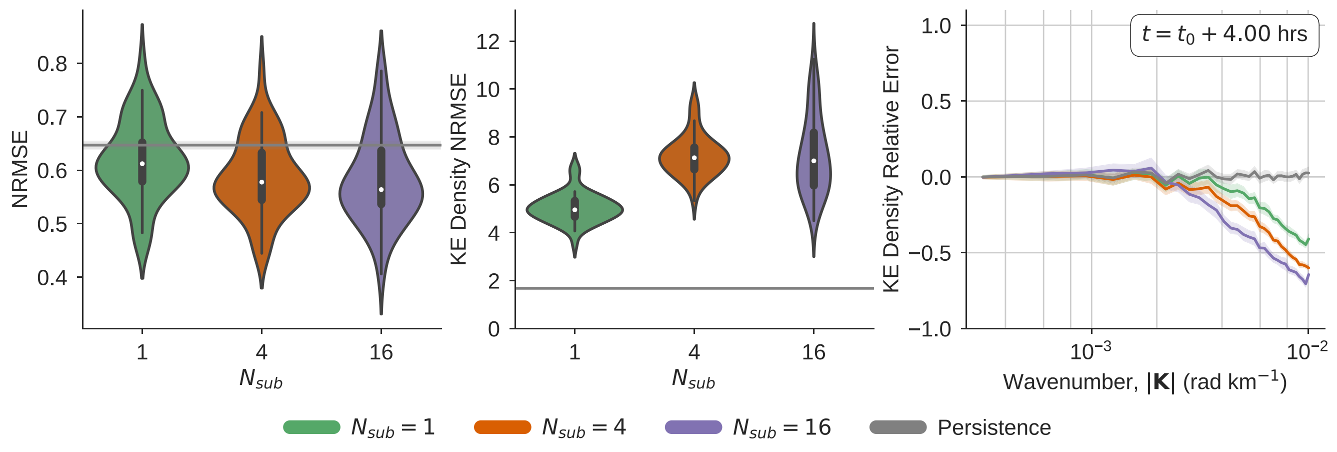

Figure 14. Subsampling related spectral errors persist even when the number of training samples is fixed. Here, the number of samples is fixed to 9.72×10^4 for all cases, and yet the temporal subsampling related spectral errors remain. Here, $\gamma = 10^{-1}$ and the solid gray line indicates the median skill of a persistent forecast.

single layer autoregressive and recurrent neural network architectures, mainly because it has been shown that they can successfully emulate low dimensional chaotic dynamics over multiple Lyapunov timescales (Gauthier et al., 2021; Pathak et al., 2017; Platt et al., 2022; Vlachas et al., 2020). We implemented a multi-dimensional parallelization scheme based on the concept introduced by Pathak et al. (2018) and similar to that of Arcomano et al. (2020) in order to scale up these architectures and test them in high dimensional systems. We note that an in-depth discussion of our software implementation using the task based scheduling system in python, Dask (Dask Development Team, 2016), will be covered in a forthcoming paper.

6.1. Main Result and Connections to Previous Work

Our main result is that we observe an inherent spectral bias that occurs when training data are subsampled in time, such that as the temporal resolution is reduced, the resolution of small scale features in NVAR and ESN predictions is diminished. High wavenumber spectral bias is a phenomenon that has been studied in the context of training feed forward neural networks (see Xu et al., 2022, for a comprehensive review on the topic). The authors show that while numerical Partial Differential Equation (PDE) solvers typically resolve small spatial scales first and iteratively refine the larger spatial scales, spectral biases arise while training neural networks because the reverse happens: the large scales are uncovered first and small spatial scales are slowly refined.

Here, we showed a similar bias that arises in NVAR and ESN architectures in relation to their temporal resolution. Given the sensitivity to model time step, this phenomenon bears resemblance to the Courant-Friedrich-Lewy (CFL) condition, which poses an upper bound on the time step size that can be used in the numerical solution of PDEs. The CFL condition is therefore a barrier to weather and climate model efficiency. However, sensitivity to the time step size manifests very differently in neural networks and numerical PDEs. While violating the CFL condition with too large of a time step leads to fundamental issues of numerical instability in numerical PDEs, here we see that increasing the time step adds a sort of numerical dissipation, which can actually stabilize an otherwise unstable model architecture (Section 4.1). Our results show that this occurs because the small scales are "lost" within the recurrent and autoregressive time stepping relations. Because of this, the models are trained to take on an interpolated or spatially averaged view of the intermediate dynamical behavior, which generates a blurred prediction.

We note that Bi et al. (2022) discuss a similar phenomenon relating to the timestepping of their autoregressive transformer model. Specifically, they devise a "Hierarchical Temporal Aggregation" scheme to make more stable and accurate forecasts (in terms of RMSE) over longer periods of time than they would potentially be able to if they were to use the original 1 hr cadence of the ERA5 data set. However, it is not clear how well small scale features are preserved with this approach. This is unclear first because they use a cost function that is purely based on RMSE. Second, the approach requires training multiple models at successively larger time intervals, and a forecast is made using the largest interval possibly available first. For instance, with trained models operating on 1 and 6 hourly increments, a 7 hr forecast would be made by first a 6 and then 1 hr prediction. Our results indicate that this could be problematic, as the model making the 6 hr prediction would filter out small scale features that would otherwise be captured by the second model, operating on a 1 hr timestep.

Finally, Chattopadhyay and Hassanzadeh (2023) show the connection between high wavenumber spectral bias and instabilities in neural network predictions of turbulent flows. Their focus was on achieving long term stability in neural network time stepping for climate applications, while the focus in our work has been on short term forecasting for weather applications - capturing the long term, climate statistics in turbulent geophysical fluid dynamics with an ESN or NVAR is future work. However, both works (a) draw some connection between high frequency spectral bias and the time stepping of the neural network, and (b) offer potential solutions by penalizing the solution's spectrum. In our work, we show that some of the spectral bias stems from the timestep size of the data used for training, while Chattopadhyay and Hassanzadeh (2023) devise a Runge-Kutta scheme to reduce the bias on subsampled data. Additionally, they use a spectral loss to train the internal weights of the network, along with the addition of a "corrector" network to make predictions of only the small scales. On the other hand, we use a spectral loss to guide the optimization of 5 "macro-scale" parameters, but the training of the network weights and operation of the network remain the same. Despite the differences in approach, the similarity of these two works indicates that the details of neural network time stepping schemes are crucial to their stability and accuracy in representing small scale processes. Additionally, it is clear that these small scale processes must be prioritized in some way, for instance through a loss function, and potentially additional "corrector" networks that propagate the small scales explicitly.

6.2. Implications for Training Data Sets in Weather and Climate

Our results have important implications for the rapidly developing field of neural network emulation for weather and climate forecasting because they show a potential limit to the effective resolution of an emulator relative to the original training data. If an emulator is used as a parameterization scheme for subgrid-scale dynamics, then a high wavenumber spectral bias will be detrimental to performance. Additionally, we surmise that such errors will reduce ensemble spread within data assimilation algorithms, which could limit their usefulness within a forecasting system (e.g., Kalnay et al., 2006). Our findings are pertinent to the field of neural network emulation development because of the widespread usage of reanalysis data sets for training. Currently, most existing neural network emulators in this field use the ERA5 reanalysis data set (Hersbach et al., 2020) for training (e.g., Arcomano et al., 2020; Bi et al., 2023; Keisler, 2022; Lam et al., 2022; Pathak et al., 2022; Weyn et al., 2021). Of course, reanalyses like ERA5 are an obvious choice for many reasons: the data sets are made freely available, they present a multi-decadal view of weather and climate, and, most importantly, they are constrained to observational data. However, we note that reanalysis products are imperfect for at least the following reasons: they contain jumps in the system state at the start of each DA cycle, they may contain inconsistencies reflective of changes in observational coverage, and they are only made available at large time intervals relative to the time step of the underlying integrated numerical model dynamics, due to the massive size of the data. Our study only addressed the latter of these issues, and showed that this simple space-saving step can have a negative impact on data-driven prediction methods. While we showed that adding spectral error as a weak constraint in the neural network training can reduce this time step related spectral bias, our results indicate that the underlying issue persists (Section 5.2). Moreover, as long as the data are not subsampled, we showed that ESNs perform only slightly worse when <1 year of data are used, compared to 15 years of training data (Section 5.4). This result suggests that it may be more effective to design an RNN-based emulator with a relatively short model trajectory that is not subsampled, rather than a long trajectory that is subsampled. In contrast to training the emulator on a reanalysis data set, a pure model-based emulator could then be used within a data assimilation system as by Penny et al. (2022) in order to additionally benefit from observational constraints.

6.3. Implications and Future Work Relating to Model Architecture

Due to the fact that RNNs require long, sequential data streams in order to learn the governing dynamics, it could be the case that RNNs suffer most dramatically from temporal subsampling. This hypothesis could be one reason for why the RNNs used by Agarwal et al. (2021) performed worse than other models on data that were subsampled every 10 days. Additionally, if RNNs are most dramatically affected by temporal subsampling, then they could be a suboptimal architecture choice for model emulators in cases where representing small scale dynamics is important but a coarse time step is required. This requirement is especially true when designing a parameterization scheme for subgrid-scale dynamics, where the emulator should ideally run at the same time step as the "large-scale" model.

However, given that we can qualitatively observe some degree of spectral error in a wide variety of neural network architectures that use subsampled data for training (e.g., Bi et al., 2023; Keisler, 2022; Lam et al., 2022; Pathak et al., 2022), the issue could be more general to other neural network architectures. Moreover, the similarities between our work and Chattopadhyay and Hassanzadeh (2023) as well as the reasons behind the hierarchical time stepping scheme introduced by Bi et al. (2022) (both discussed in Section 6.1) imply that the time stepping related spectral bias is a general issue. Therefore, future work should be directed at understanding the degree to which temporal resolution affects architectures other than RNNs. Potential avenues could include exploring how attention mechanisms (Dosovitskiy et al., 2021; Vaswani et al., 2017) handle this phenomenon. Additionally, in light of our results indicating that wider networks can mitigate the spectral bias at least to some degree (Section 5.3), it would be instructive to understand how successively adding layers to a neural network affects the spectral bias. Finally, we note the work of Duncan et al. (2022) who show success in using adversarial training to mitigate the spectral bias observed in FourCastNet, and suggest that such techniques deserve additional study to understand their robustness.

Of course, our neural network implementations are imperfect, and here we list some future avenues to improve their predictive capabilities. Both of the architectures relied on a mean-squared error micro-scale cost function to learn the readout matrix weights, even in the ESN models where the spectral errors were penalized in the macro-scale cost function. However, even when the spectrum was penalized and the data were not subsampled, the ESNs maintained a high wavenumber bias that resulted in KE_NRMSE far greater than that of a persistent forecast. While additional testing shows that a periodic sine activation function can reduce the high frequency bias in KE_NRMSE, following work by Sitzmann et al. (2020), the underlying problem still remains (see additional analysis in the Supporting Information S1). Therefore, in order to further reduce the high frequency bias, it may be necessary to move the spectral penalties to the micro-scale cost function, that is, to learn the readout matrix weights in the case of reservoir computing. The time stepping, spectral loss, and "small scale corrector network" employed by Chattopadhyay and Hassanzadeh (2023) would be appropriate starting points for such future work.

The NVAR architecture that we employed is incredibly simple. While we supposed that the local quadratic feature vector could learn quantities like derivatives and fluxes necessary to step the model forward in time, it is apparently not robust enough given the dramatic sensitivity to time step used. Future work could explore the possibility of using a larger library of analytic functions to improve the nonlinear expressions in the model, with the caution that this will lead to very high dimensional feature vectors. Such developments must sufficiently address the "Catch-22" described by Zhang and Cornelius (2022), who show that NVAR is inherently sensitive to the types of nonlinearity chosen. It is entirely possible, though, that an appropriate set of such basis functions exist for weather and climate emulation.

The ESN architecture that we employed is also relatively straightforward, and can undoubtedly be improved. In this work we took a somewhat brute force approach to emulate arbitrarily high dimensional systems by partitioning the system into subdomains and deploying parallel ESNs on each group. However, this process comes with overhead and can still lead to rather large networks on each group. The memory costs associated with these large networks coupled with any additional computational costs associated with timestepping, either by increasing the frequency or by using a more expensive method to represent small scale processes, will likely make the ESN implementation shown here too expensive to be considered for practical applications. Future work could explore dimension reduction techniques involving proper orthogonal decomposition (Jordanou et al., 2022), autoencoders (Heyder et al., 2022), or approaches involving self-organizing or scale invariant maps (Basterrech et al., 2011). Similarly, Whiteaker and Gerstoft (2022) show success in deriving a controllability matrix for the ESN, which leads to a reduced network size with minimal reduction in error. Finally, a number of studies claim to have developed ESN architectures that can capture dynamics occurring at many scales (Gallicchio et al., 2017, 2018; Ma et al., 2020; Malik et al., 2017; Moon et al., 2021), and these could be explored for geophysical turbulence emulation as well.

7. Conclusions

Recent advances in neural network based emulators of Earth's weather and climate indicate that forecasting centers could benefit greatly from incorporating neural networks into their future prediction systems. However, a common issue with these data-driven models is that they produce relatively blurry predictions, and misrepresent the small spatial scale features that can be resolved in traditional, physics-based forecasting models. Here, we showed that the simple space saving step of subsampling the training data used to generate recurrent neural network emulators accentuates this small scale error. While we show some success in mitigating the effects of this subsampling related, high wavenumber bias through an inner/outer loop optimization framework, the problem persists. Many neural network emulators use subsampled data sets for training, including most prominently the ERA5 Reanalysis. While our work suggests that there could be a benefit to using a training data set based on a relatively shorter model trajectory that is not subsampled, rather than a longer one that is, addressing the subsampling issue would provide more confidence in using already existing, freely available data sets like reanalyses. We therefore suggest that future work should focus on how other architectures and techniques like attention or adversarial training can address this subsampling related bias at the small spatial scales of turbulent geophysical fluid dynamics.

Appendix A: Matrix and Data Normalization for Echo State Networks

Here we describe several aspects of our ESN implementation that are unique with respect to previous works. Additionally, we provide some empirical justification for these choices, using the Lorenz96 model as a testbed (Lorenz, 1996), see Appendix A4 for a description of the data sets generated for these tests.

Our testing framework follows the general procedure laid out by Platt et al. (2022) to evaluate the architecture choices. For each design choice, we compute the Valid Prediction Time (VPT) of an ESN model over 100 randomly chosen initial conditions from a test data set. VPT is computed as

$$\mathrm{VPT} = \underset{n}{\operatorname{argmin}}\{\mathrm{NRMSE}(n) > \epsilon\}$$

$$\mathrm{NRMSE}(n) = \sqrt{\frac{1}{N_{\mathbf{v}}}\sum_{i=1}^{N_{\mathbf{v}}}\left(\frac{\hat{v}_i(n) - v_i(n)}{\mathrm{SD}_i}\right)^2},$$

where n is a time index, SD_i is the temporal standard deviation of the ith dimension, computed from the training data, and $\epsilon = 0.2$. To eliminate the dependence of the results on the randomly chosen adjacency and input matrices, we repeat the process for 10 different adjacency and input matrix pairs, initialized with different random number generator seeds. In total, we compare each design choice with a VPT distribution from 1,000 test samples. We note that we optimize the ESN parameters listed in Equation 7 for each design choice and each random matrix pair, following the procedure described in Section 5.1 with an NRMSE cost function. Of course, these tests are insufficient to definitively prove that these choices will translate perfectly to the SQG system. However, we consider this to be a bare minimum test that will catch downright bad design choices, while saving the computing resources necessary to train an emulator for larger problems.

A1. Input Matrix Scaling

Typically, $\mathbf{W}_{\text{in}}$ is filled with entries

$$\hat{w}_{i,j} \sim \mathcal{U}(-\sigma, \sigma) \qquad i = \{1, 2, \ldots, N_{\mathbf{r}}\}, j = \{1, 2, \ldots, N_{\mathbf{u}}\}$$

where σ determines the bounds of the uniform distribution. Here we found it to be advantageous to normalize the input matrix by the largest singular value. That is, we first compute $\hat{\mathbf{W}}_{\text{in}}$, with elements

$$\hat{w}_{i,j} \sim \mathcal{U}(-1, 1) \qquad i = \{1, 2, \ldots, N_{\mathbf{r}}\}, j = \{1, 2, \ldots, N_{\mathbf{u}}\}.$$

Then, we set $\mathbf{W}_{\text{in}}$ as

$$\mathbf{W}_{\text{in}} := \frac{\sigma}{\sigma_{\max}\left(\hat{\mathbf{W}}_{\text{in}}\right)}\hat{\mathbf{W}}_{\text{in}}$$

where $\sigma_{\max}(\cdot)$ is the largest singular value, and the parameter σ is the desired largest singular value of $\mathbf{W}_{\text{in}}$.

Our motivation for using this type of normalization is that we found it necessary to use very wide parameter optimization bounds for σ when using the standard input scaling strategy. Normalizing the matrix by the largest singular value compensates for the fact that the amplitude of the contributions to the reservoir, that is, the elements of the vector

$$\mathbf{p} = \mathbf{W}_{\text{in}}\mathbf{u} = \begin{pmatrix} \mathbf{w}_1^T\mathbf{u} \\ \mathbf{w}_2^T\mathbf{u} \\ \vdots \\ \mathbf{w}_{N_{\mathbf{r}}}^T\mathbf{u} \end{pmatrix}$$

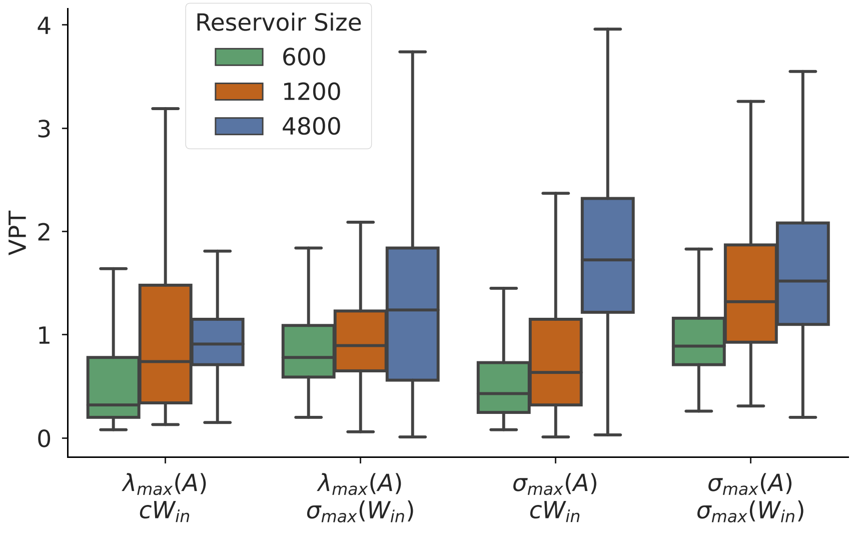

Figure A1. Valid Prediction Time (VPT) obtained with an ESN, using different normalization strategies for the adjacency and input matrices, **A** and $\mathbf{W}_{\text{in}}$. The normalization used for each matrix is indicated as follows: $\lambda_{\max}(\cdot)$ refers to the largest eigenvalue (i.e., spectral radius), $\sigma_{\max}(\cdot)$ refers to the largest singular value (i.e., induced 2 norm), while c implies that no normalization was used. The results are computed with the 20D Lorenz96 system, described in Appendix A4. The boxplots indicate prediction skill from 10 different adjacency and input matrices, achieved by changing the random number generator seed, with 100 initial conditions randomly sampled from the test data set for each set of matrices. The macro-scale parameters, including also the leak rate, bias, and Tikhonov parameter, were optimized for each unique matrix pair. Color indicates the size of the reservoir used.

grow with $N_{\mathbf{u}}$. By controlling for this growth, we were able to reduce the optimization search space and achieve more consistent prediction skill with fewer iterations.

Additionally, we found empirical evidence to suggest that this normalization is advantageous even for small systems. Figure A1 shows the VPT achieved with the 20-Dimensional Lorenz96 system (Appendix A4), using a variety of normalization strategies for the input and adjacency matrices. In Figure A1, the two schemes used for the input matrix are (a) no normalization (indicated by cW_{in}) and (b) normalization by the largest singular value (indicated by $\sigma_{\max}(W_{in})$). For a variety of reservoir sizes, $N_{\mathbf{r}}$, we found that using the largest singular value often performed better, usually by about 0.5 MTU.

A2. Adjacency Matrix Scaling

Typically, the reservoir adjacency matrix is normalized to achieve a desired spectral radius. That is, the matrix $\hat{\mathbf{A}}$ is generated with elements $\hat{a}_{i,j} \sim \mathcal{U}(-1,1)$, where i, j are random indices in order to satisfy the desired sparsity of the matrix (all other elements are 0). Then, **A** is set as

$$\mathbf{A} := \frac{\rho}{\lambda_{\max}\left(\hat{\mathbf{A}}\right)}\hat{\mathbf{A}},$$

where $\lambda_{\max}(\cdot)$ is the spectral radius, and ρ scales the matrix to achieve the desired spectral radius. A common guideline is to set $\rho \simeq 1$, as it is hypothesized that this puts the reservoir on the "edge of stability" so that it performs well in emulating nonlinear systems (e.g., as recommended by Lukoševičius, 2012). However, as originally described by Jaeger (2001), the spectral radius provides only a necessary, but insufficient, means to satisfy the required Echo State Property. On the other hand, using the largest singular value is a sufficient condition for satisfying the echo state property.

In our experimentation, we have found a slight benefit from using the largest singular value to normalize the adjacency matrix. Figure A1 shows that, for fixed input matrix normalization, using the largest singular value rather than spectral radius achieves similar and up to ∼0.3 longer valid predictions. While the improvement may seem subtle, we note that using the largest singular value has the following practical benefit for our python-based implementation: the singular values can be computed directly on a Graphical Processing Unit using CuPy (Okuta et al., 2017), while a general, non-symmetric eigenvalue decomposition is not readily available.

A3. Data Normalization

A key aspect in machine learning is normalizing input data before passing it to the model. Experiments from Platt et al. (2022) showed, however, that the standard approach to normalizing data can be detrimental to prediction skill. By "standard approach," we mean

$$v_i(n) = \frac{v_i(n) - \bar{v}_i}{\mathrm{SD}_i} \qquad i = \{1, 2, \ldots N_v\},$$

where

$$\bar{v}_i = \frac{1}{N_{\mathrm{train}}} \sum_{n=1}^{N_{\mathrm{train}}} v_i(n), \mathrm{SD}_i = \sqrt{\frac{\sum_{n=1}^{N_{\mathrm{train}}} (v_i(n) - \bar{v}_i)^2}{N_{\mathrm{train}} - 1}}$$

that is, $\bar{v}_i$ and SD_i are the mean and standard deviation taken from the training data separately over each channel of data, indexed by i. The key takeaway from Platt et al. (2022) is that by using separate normalization values for each channel, the covarying relationships between the data are destroyed and the reservoir cannot learn the true dynamics. The authors propose to normalize with the average and range of the data, computed over the length of the training data and over all channels

$$v_i(n) = \frac{v_i(n) - \mu}{v_{\max} - v_{\min}} \qquad i = \{1, 2, \ldots N_v\}, \tag{A1}$$

where

$$\mu = \frac{1}{N_v} \sum_{i=1}^{N_v} \bar{v}_i, \qquad v_{\max} = \max_{\substack{i=\{1,\ldots,N_v\} \\ n=\{1,\ldots,N_{\mathrm{train}}\}}} (v_i(n)), \qquad v_{\min} = \min_{\substack{i=\{1,\ldots,N_v\} \\ n=\{1,\ldots,N_{\mathrm{train}}\}}} (v_i(n)). \tag{A2}$$

Here, we propose to replace the range in the denominator with the standard deviation computed over all channels and timesteps in the training data,

$$v_i(n) = \frac{v_i(n) - \mu}{\mathrm{SD}} \qquad i = \{1, 2, \ldots N_v\}, \tag{A3}$$

with

$$\mathrm{SD} = \sqrt{\frac{\sum_{i=1}^{N_v} \sum_{n=1}^{N_{\mathrm{train}}} (v_i(n) - \mu)^2}{(N_{\mathrm{train}} - 1)(N_v - 1)}}.$$

Figure A2 compares the prediction skill when these two normalization strategies are used. Using the standard deviation normalization as in Equation A3 leads to an average VPT increase of 2 MTU. We suggest that this improvement is due to the fact that when the data are normalized by the full range, then all values are in the range $[-1, 1]$. In this case, once the input is mapped into the hidden space, it is more likely to lie on the linear regime of the $\tanh(\cdot)$ activation function. While a large enough input scaling could eliminate this problem, it is apparently not easily obtained during the Bayesian optimization.

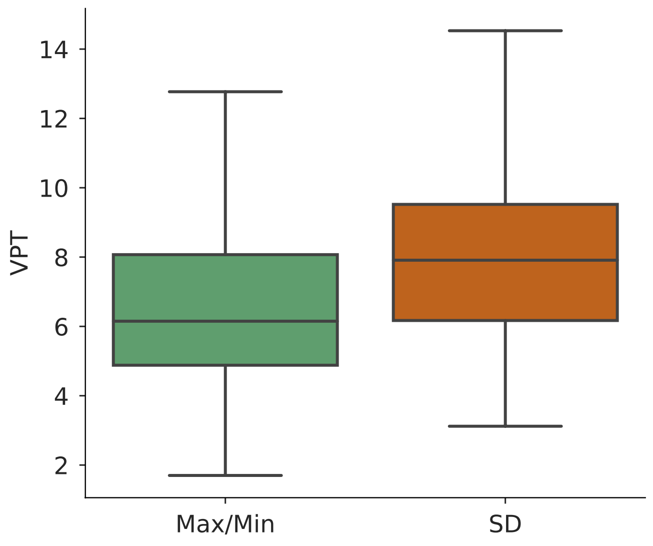

Figure A2. Valid Prediction Time (VPT) with an ESN, using the Max/Min normalization strategy shown in Equation A1 and standard deviation (SD) normalization strategy as in Equation A3. The results are computed with the 6D Lorenz96 system, described in Appendix A4. The boxplots indicate prediction skill from 10 different adjacency and input matricesm, achieved by changing the random number generator seed, with 100 initial conditions randomly sampled from the test data set for each set of matrices. All macro-scale parameters were optimized for each unique matrix pair.

A4. Lorenz96 Data Sets

The Lorenz96 data set used for these supplemental experiments were generated by the following set of equations introduced by Lorenz (1996),

$$\frac{dv_i(t)}{dt} = v_{i-1}(t)(v_{i+1}(t) - v_{i-2}(t)) - v_i(t) + F,$$

where $i = 1, 2, \ldots, N_l$, and the domain is periodic. $F = 8$ is a fixed parameter that generates chaotic dynamics. We use $N_l = 20$ for the tests in Appendices A1 and A2 and $N_l = 6$ for the tests in Appendix A3. Each data set was generated by stepping the model forward with a fourth order Runge-Kutta scheme with $\Delta t = 0.01$ Model Time Units (MTU). Each data set consisted of a 10 MTU spinup period that was discarded, 420 MTU of training data, a 60 MTU validation period, and a 120 MTU test period. Each randomly chosen validation and test trajectory were 1 MTU and 15 MTU, respectively, and the ESN spinup period was 5 MTU.

Appendix B: Gulf of Mexico Data Set and ESN Prediction

The Gulf of Mexico reanalysis data set used to generate the prediction in Figure 1 was provided by HYCOM (2016). The data consists of 3 hourly snapshots of 2D sea surface height and 3D temperature, salinity, and zonal and meridional velocities, covering 1993–2012 (inclusive). We used only the top level of temperature, and used the first 18 years as training, and the last 2 years as test data. Here we apply a parallelized ESN architecture, using $N_x^{\text{loc}} = N_y^{\text{loc}} = 4$, $N_o = 1$, and $N_{\mathbf{r}} = 6,000$. Because we use only the top level of temperature, $N_z = 1$, and therefore $N_{\mathbf{u}}^{\text{loc}} = 25$, $N_{\mathbf{v}}^{\text{loc}} = 16$. The grid cells that represent continental land are ignored in the input and output vectors, and in the corresponding rows of $\mathbf{W}_{\text{out}}$. Therefore, the effect of the boundary conditions on the neighboring grid cells is implicitly learned from the data.

Data Availability Statement

The model configurations used to generate the results in this manuscript can be found at https://github.com/timothyas/rc-gfd, and specifically used the version at Smith (2023). The repository uses a modified version of the SQG Turbulence model code developed by Jeffrey S. Whitaker at https://github.com/jswhit/sqgturb.

Acknowledgments
T.A. Smith and S.G. Penny acknowledge support from NOAA Grant NA20OAR4600277. S.G. Penny and J.A. Platt acknowledge support from the Office of Naval Research (ONR) Grants N00014-19-1-2522 and N00014-20-1-2580. T.A. Smith thanks Nora Loose for comments and discussion that improved the manuscript. The authors thank Jeffrey S. Whitaker for developing the open source SQG Turbulence model code which was the basis for our training, validation, and testing data set. The authors thank three anonymous reviewers, the associate editor, and editor Stephen Griffies for comments that improved the manuscript.

References

Agarwal, N., Kondrashov, D., Dueben, P., Ryzhov, E., & Berloff, P. (2021). A comparison of data-driven approaches to build low-dimensional ocean models. *Journal of Advances in Modeling Earth Systems*, *13*(9), e2021MS002537. https://doi.org/10.1029/2021MS002537

Arcomano, T., Szunyogh, I., Pathak, J., Wikner, A., Hunt, B. R., & Ott, E. (2020). A machine learning-based global atmospheric forecast model. *Geophysical Research Letters*, *47*(9), e2020GL087776. https://doi.org/10.1029/2020GL087776

Barbosa, W. A. S., & Gauthier, D. J. (2022). Learning spatiotemporal chaos using next-generation reservoir computing. arXiv:2203.13294 [nlin]. Retrieved from http://arxiv.org/abs/2203.13294

Basterrech, S., Fyfe, C., & Rubino, G. (2011). Self-organizing maps and scale-invariant maps in echo state networks. In *2011 11th international conference on intelligent systems design and applications* (pp. 94–99). https://doi.org/10.1109/ISDA.2011.6121637

Bi, K., Xie, L., Zhang, H., Chen, X., Gu, X., & Tian, Q. (2022). Pangu-weather: A 3D high-resolution model for fast and accurate global weather forecast. arXiv. Retrieved from http://arxiv.org/abs/2211.02556

Bi, K., Xie, L., Zhang, H., Chen, X., Gu, X., & Tian, Q. (2023). Accurate medium-range global weather forecasting with 3D neural networks. *Nature*, *619*(7970), 533–538. https://doi.org/10.1038/s41586-023-06185-3

Blumen, W. (1978a). Uniform potential vorticity flow: Part I. Theory of wave interactions and two-dimensional turbulence. *Journal of the Atmospheric Sciences*, *35*(5), 774–783. https://doi.org/10.1175/1520-0469(1978)035⟨0774:UPVFPI⟩2.0.CO;2

Blumen, W. (1978b). Uniform potential vorticity flow: Part II. A model of wave interacions. *Journal of the Atmospheric Sciences*, *35*(5), 784–789. https://doi.org/10.1175/1520-0469(1978)035⟨0784:UPVFPI⟩2.0.CO;2

Bollt, E. (2021). 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*(1), 013108. https://doi.org/10.1063/5.0024890

Bouhlel, M. A., He, S., & Martins, J. R. R. A. (2020). Scalable gradient–enhanced artificial neural networks for airfoil shape design in the subsonic and transonic regimes. *Structural and Multidisciplinary Optimization*, *61*(4), 1363–1376. https://doi.org/10.1007/s00158-020-02488-5

Bouhlel, M. A., Hwang, J. T., Bartoli, N., Lafage, R., Morlier, J., & Martins, J. R. R. A. (2019). A Python surrogate modeling framework with derivatives. *Advances in Engineering Software*, *135*, 102662. https://doi.org/10.1016/j.advengsoft.2019.03.005

Chattopadhyay, A., & Hassanzadeh, P. (2023). Long-term instabilities of deep learning-based digital twins of the climate system: The cause and a solution. arXiv. Retrieved from http://arxiv.org/abs/2304.07029

Chen, T.-C., Penny, S. G., Smith, T. A., & Platt, J. A. (2022). "Next generation" reservoir computing: An empirical data-driven expression of dynamical equations in time-stepping form. https://doi.org/10.48550/arXiv.2201.05193

Chen, X., Nadiga, B. T., & Timofeyev, I. (2021). Predicting shallow water dynamics using echo-state networks with transfer learning. arXiv:2112.09182 [physics]. Retrieved from http://arxiv.org/abs/2112.09182

Cressie, N. (1993). Statistics for spatial data.

Dask Development Team. (2016). Dask: Library for dynamic task scheduling [Computer software manual]. Dask Development Team. Retrieved from https://dask.org

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., et al. (2021). An Image is Worth 16x16 Words: Transformers for image recognition at scale. arXiv. Retrieved from http://arxiv.org/abs/2010.11929

Dueben, P. D., & Bauer, P. (2018). Challenges and design choices for global weather and climate models based on machine learning. *Geoscientific Model Development*, *11*(10), 3999–4009. https://doi.org/10.5194/gmd-11-3999-2018

Duncan, J., Subramanian, S., & Harrington, P. (2022). Generative modeling of high-resolution global precipitation forecasts. https://doi.org/10.48550/arXiv.2210.12504

Eady, E. T. (1949). Long waves and cyclone waves. *Tellus*, *1*(3), 33–52. https://doi.org/10.1111/j.2153-3490.1949.tb01265.x

Evensen, G., Vossepoel, F. C., & van Leeuwen, P. J. (2022). *Data assimilation fundamentals: A unified formulation of the state and parameter estimation problem*. Springer International Publishing. https://doi.org/10.1007/978-3-030-96709-3

Gallicchio, C., Micheli, A., & Pedrelli, L. (2017). Deep reservoir computing: A critical experimental analysis. *Neurocomputing*, *268*, 87–99. https://doi.org/10.1016/j.neucom.2016.12.089

Gallicchio, C., Micheli, A., & Pedrelli, L. (2018). Design of deep echo state networks. *Neural Networks*, *108*, 33–47. https://doi.org/10.1016/j.neunet.2018.08.002

Gauthier, D. J., Bollt, E., Griffith, A., & Barbosa, W. A. S. (2021). Next generation reservoir computing. *Nature Communications*, *12*(1), 5564. https://doi.org/10.1038/s41467-021-25801-2

Goodfellow, I., Yoshua, B., & Aaron, C. (2016). Sequence modeling: Recurrent and recursive nets. In *Deep learning*. MIT Press. Retrieved from https://www.deeplearningbook.org/

Griffith, A., Pomerance, A., & Gauthier, D. J. (2019). Forecasting chaotic systems with very low connectivity reservoir computers. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, *29*(12), 123108. https://doi.org/10.1063/1.5120710

Hasselmann, K. (1988). PIPs and POPs: The reduction of complex dynamical systems using principal interaction and oscillation patterns. *Journal of Geophysical Research*, *93*(D9), 11015–11021. https://doi.org/10.1029/JD093iD09p11015

Held, I. M., Pierrehumbert, R. T., Garner, S. T., & Swanson, K. L. (1995). Surface quasi-geostrophic dynamics. *Journal of Fluid Mechanics*, *282*, 1–20. https://doi.org/10.1017/S0022112095000012

Hermans, M., & Schrauwen, B. (2010). Memory in reservoirs for high dimensional input. In *The 2010 international joint conference on neural networks (IJCNN)* (pp. 1–7). IEEE. https://doi.org/10.1109/IJCNN.2010.5596884

Hersbach, H., Bell, B., Berrisford, P., Hirahara, S., Horányi, A., Muñoz-Sabater, J., et al. (2020). The ERA5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, *146*(730), 1999–2049. https://doi.org/10.1002/qj.3803

Hewitt, H. T., Roberts, M. J., Hyder, P., Graham, T., Rae, J., Belcher, S. E., et al. (2016). The impact of resolving the Rossby radius at mid-latitudes in the ocean: Results from a high-resolution version of the Met Office GC2 coupled model. *Geoscientific Model Development*, *9*(10), 3655–3670. https://doi.org/10.5194/gmd-9-3655-2016

Heyder, F., Mellado, J. P., & Schumacher, J. (2022). Generalizability of reservoir computing for flux-driven two-dimensional convection. *Physical Review*, *106*(5), 055303. https://doi.org/10.1103/PhysRevE.106.055303

HYCOM. (2016). HYCOM + NCODA Gulf of Mexico 1/25° reanalysis, (GOMu0.04/expt_50.1). (Data retrieved from HYCOM, Retrieved from https://www.hycom.org/data/gomu0pt04/expt-50pt1)

Jaeger, H. (2001). The "echo state" approach to analysing and training recurrent neural networks—With an Erratum note. *German National Research Center for Information Technology GMD Technical Report*, *148*, 13.

Jones, D. R., Schonlau, M., & Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, *13*(4), 455–492. https://doi.org/10.1023/A:1008306431147

Jordanou, J. P., Antonelo, E. A., Camponogara, E., & Gildin, E. (2022). Investigation of proper orthogonal decomposition for echo state networks. arXiv. Retrieved from http://arxiv.org/abs/2211.17179

Kalnay, E., Hunt, B., Ott, E., & Szunyogh, I. (2006). Ensemble forecasting and data assimilation: Two problems with the same solution. *Predictability of weather and climate*, *157*, 180.

Keisler, R. (2022). Forecasting global weather with graph neural networks. arXiv:2202.07575 [physics]. Retrieved from http://arxiv.org/abs/2202.07575

Lam, R., Sanchez-Gonzalez, A., Willson, M., Wirnsberger, P., Fortunato, M., Pritzel, A., et al. (2022). GraphCast: Learning skillful medium-range global weather forecasting. https://doi.org/10.48550/arXiv.2212.12794

Li, J., Bouhlel, M. A., & Martins, J. R. R. A. (2019). Data-based approach for fast airfoil analysis and optimization. *AIAA Journal*, *57*(2), 581–596. https://doi.org/10.2514/1.J057129

Lorenz, E. (1996). Predictability—A problem partly solved. In *Proceedings of a seminar held at ECMWF on predictability*.

Lu, Z., Hunt, B. R., & Ott, E. (2018). Attractor reconstruction by machine learning. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, *28*(6), 061104. https://doi.org/10.1063/1.5039508

Lukoševičius, M. (2012). A practical guide to applying echo state networks. In G. Montavon, G. B. Orr, & K.-R. Müller (Eds.), *Neural networks: Tricks of the trade* (2nd ed., pp. 659–686). Springer. https://doi.org/10.1007/978-3-642-35289-8_36

Ma, Q., Shen, L., & Cottrell, G. W. (2020). DeePr-ESN: A deep projection-encoding echo-state network. *Information Sciences*, *511*, 152–171. https://doi.org/10.1016/j.ins.2019.09.049

Maass, W., Natschläger, T., & Markram, H. (2002). Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, *14*(11), 2531–2560. https://doi.org/10.1162/089976602760407955

Malik, Z. K., Hussain, A., & Wu, Q. J. (2017). Multilayered echo state machine: A novel architecture and algorithm. *IEEE Transactions on Cybernetics*, *47*(4), 946–959. https://doi.org/10.1109/TCYB.2016.2533545

Moon, J., Wu, Y., & Lu, W. D. (2021). Hierarchical architectures in reservoir computing systems. *Neuromorphic Computing and Engineering*, *1*(1), 014006. https://doi.org/10.1088/2634-4386/ac1b75

Moore, A. M., Fiechter, J., & Edwards, C. A. (2022). A linear stochastic emulator of the California Current system using balanced truncation. *Ocean Modelling*, *174*, 102023. https://doi.org/10.1016/j.ocemod.2022.102023

Nadiga, B. T. (2021). Reservoir computing as a tool for climate predictability studies. *Journal of Advances in Modeling Earth Systems*, *13*(4), e2020MS002290. https://doi.org/10.1029/2020MS002290

Najm, H. N. (2009). Uncertainty quantification and polynomial chaos techniques in computational fluid dynamics. *Annual Review of Fluid Mechanics*, *41*(1), 35–52. https://doi.org/10.1146/annurev.fluid.010908.165248

Okuta, R., Unno, Y., Nishino, D., Hido, S., & Loomis, C. (2017). Cupy: A numpy-compatible library for nvidia gpu calculations. In *Proceedings of workshop on machine learning systems (learningsys) in the thirty-first annual conference on neural information processing systems (NIPS)*. Retrieved from http://learningsys.org/nips17/assets/papers/paper_16.pdf

Orlanski, I. (1975). A Rational subdivision of scales for atmospheric processes. *Bulletin of the American Meteorological Society*, *56*(5), 527–530. Retrieved from https://www.jstor.org/stable/26216020

Pathak, J., Hunt, B., Girvan, M., Lu, Z., & Ott, E. (2018). Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. *Physical Review Letters*, *120*(2), 024102. https://doi.org/10.1103/PhysRevLett.120.024102

Pathak, J., Lu, Z., Hunt, B. R., Girvan, M., & Ott, E. (2017). Using machine learning to replicate chaotic attractors and calculate Lyapunov exponents from data. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, *27*(12), 121102. https://doi.org/10.1063/1.5010300

Pathak, J., Subramanian, S., Harrington, P., Raja, S., Chattopadhyay, A., Mardani, M., et al. (2022). FourCastNet: A global data-driven high-resolution weather model using adaptive fourier neural operators. arXiv:2202.11214 [physics]. Retrieved from http://arxiv.org/abs/2202.11214

Penland, C. (1989). Random forcing and forecasting using principal oscillation pattern analysis. *Monthly Weather Review*, *117*(10), 2165–2185. https://doi.org/10.1175/1520-0493(1989)117⟨2165:RFAFUP⟩2.0.CO;2

Penny, S. G., Akella, S., Alves, O., Craig, B., Buehner, M., Chevallier, M., et al. (2017). *Coupled data assimilation for integrated earth system analysis and prediction: Goals, challenges and recommendations (Tech. Rep.)*. World Meteorological Organization.

Penny, S. G., Smith, T. A., Chen, T.-C., Platt, J. A., Lin, H.-Y., Goodliff, M., & Abarbanel, H. D. I. (2022). Integrating recurrent neural networks with data assimilation for scalable data-driven state estimation. *Journal of Advances in Modeling Earth Systems*, *14*(3), e2021MS002843. https://doi.org/10.1029/2021MS002843

Platt, J. A., Penny, S. G., Smith, T. A., Chen, T.-C., & Abarbanel, H. D. I. (2022). A systematic exploration of reservoir computing for forecasting complex spatiotemporal dynamics. *Neural Networks*, *153*, 530–552. https://doi.org/10.1016/j.neunet.2022.06.025

Platt, J. A., Penny, S. G., Smith, T. A., Chen, T.-C., & Abarbanel, H. D. I. (2023). Constraining Chaos: Enforcing dynamical invariants in the training of recurrent neural networks. https://doi.org/10.48550/arXiv.2304.12865

Rasp, S., & Thuerey, N. (2021). Data-driven medium-range weather prediction with a Resnet pretrained on climate simulations: A new model for WeatherBench. *Journal of Advances in Modeling Earth Systems*, *13*(2), e2020MS002405. https://doi.org/10.1029/2020MS002405

Rossa, A., Nurmi, P., & Ebert, E. (2008). Overview of methods for the verification of quantitative precipitation forecasts. In S. Michaelides (Ed.), *Precipitation: Advances in measurement, estimation and prediction* (pp. 419–452). Springer. https://doi.org/10.1007/978-3-540-77655-0_16

Scher, S. (2018). Toward data-driven weather and climate forecasting: Approximating a simple general circulation model with deep learning. *Geophysical Research Letters*, *45*(22), 12616–12622. https://doi.org/10.1029/2018GL080704

Scher, S., & Messori, G. (2019). Weather and climate forecasting with neural networks: Using general circulation models (GCMs) with different complexity as a study ground. *Geoscientific Model Development*, *12*(7), 2797–2809. https://doi.org/10.5194/gmd-12-2797-2019

Schultz, M. G., Betancourt, C., Gong, B., Kleinert, F., Langguth, M., Leufen, L. H., et al. (2021). Can deep learning beat numerical weather prediction? *Philosophical Transactions of the Royal Society A: Mathematical, Physical & Engineering Sciences*, *379*(2194), 20200097. https://doi.org/10.1098/rsta.2020.0097

Sitzmann, V., Martel, J. N. P., Bergman, A. W., Lindell, D. B., & Wetzstein, G. (2020). Implicit neural representations with periodic activation functions. arXiv. Retrieved from http://arxiv.org/abs/2006.09661

Smith, T. (2023). *timothyas/rc-gfd: Revision 1*. Zenodo. https://doi.org/10.5281/zenodo.8368225

Steil, J. (2004). Backpropagation-decorrelation: Online recurrent learning with O(N) complexity. In *2004 IEEE international joint conference on neural networks (IEEE Cat. No.04CH37541)* (Vol. 2, pp. 843–848). https://doi.org/10.1109/IJCNN.2004.1380039

Tikhonov, A. N. (1963). Solution of incorrectly formulated problems and the regularization method. *Soviet Math. Dokl*.

Tulloch, R., & Smith, K. S. (2009). A note on the numerical representation of surface dynamics in Quasigeostrophic turbulence: Application to the nonlinear Eady model. *Journal of the Atmospheric Sciences*, *66*(4), 1063–1068. https://doi.org/10.1175/2008JAS2921.1

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al. (2017). Attention is all you need. In *Advances in neural information processing systems* (Vol. 30). Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., et al. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods*, *17*(3), 261–272. https://doi.org/10.1038/s41592-019-0686-2

Vlachas, P. R., Pathak, J., Hunt, B. R., Sapsis, T. P., Girvan, M., Ott, E., & Koumoutsakos, P. (2020). Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics. *Neural Networks*, *126*, 191–217. https://doi.org/10.1016/j.neunet.2020.02.016

Weyn, J. A., Durran, D. R., & Caruana, R. (2019). Can machines learn to predict weather? Using deep learning to predict gridded 500-hPa geopotential height from historical weather data. *Journal of Advances in Modeling Earth Systems*, *11*(8), 2680–2693. https://doi.org/10.1029/2019MS001705

Weyn, J. A., Durran, D. R., & Caruana, R. (2020). Improving data-driven global weather prediction using deep convolutional neural networks on a cubed sphere. *Journal of Advances in Modeling Earth Systems*, *12*(9), e2020MS002109. https://doi.org/10.1029/2020MS002109

Weyn, J. A., Durran, D. R., Caruana, R., & Cresswell-Clay, N. (2021). Sub-seasonal forecasting with a large ensemble of deep-learning weather prediction models. *Journal of Advances in Modeling Earth Systems*, *13*(7), e2021MS002502. https://doi.org/10.1029/2021MS002502

Whiteaker, B., & Gerstoft, P. (2022). Reducing echo state network size with controllability matrices. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, *32*(7), 073116. https://doi.org/10.1063/5.0071926

Xu, Z.-Q. J., Zhang, Y., & Luo, T. (2022). Overview frequency principle/spectral bias in deep learning. arXiv. Retrieved from http://arxiv.org/abs/2201.07395

Zhang, Y., & Cornelius, S. P. (2022). A catch-22 of reservoir computing. arXiv. Retrieved from http://arxiv.org/abs/2210.10211