A Machine-Learning-Based Global Atmospheric Forecast Model

Istvan Szunyogh[1,1], Troy Arcomano[1,1], Jaideep Pathak[2,2], Alexander Wikner[2,2], Brian Hunt[2,2], and Edward Ott[2,2]

[1]Texas A&M University
[2]University of Maryland

November 30, 2022

Abstract

The paper investigates the applicability of machine learning (ML) to weather prediction by building a reservoir-computing-based, low-resolution, global prediction model. The model is designed to take advantage of the massively parallel architecture of a modern supercomputer. The forecast performance of the model is assessed by comparing it to that of daily climatology, persistence, and a numerical (physics-based) model of identical prognostic state variables and resolution. Hourly resolution 20-day forecasts with the model predict realistic values of the atmospheric state variables at all forecast times for the entire globe. The ML model outperforms both climatology and persistence for the first three forecast days in the midlatitudes, but not in the tropics. Compared to the numerical model, the ML model performs best for the state variables most affected by parameterized processes in the numerical model.

A Machine-Learning-Based Global Atmospheric Forecast Model

Troy Arcomano[1], Istvan Szunyogh[1], Jaideep Pathak[2], Alexander Wikner[2], Brian R. Hunt[3], and Edward Ott[4]

[1]Department of Atmospheric Sciences, Texas A&M University, Texas, USA.
[2]Department of Physics, University of Maryland, College Park, Maryland, USA.
[3]Department of Mathematics, University of Maryland, College Park, Maryland, USA.
[4]Department of Physics and Department of Electrical and Computer Engineering, University of Maryland, College Park, Maryland, USA.

Key Points:

- A low-resolution, global, reservoir-computing-based machine learning (ML) model can forecast the atmospheric state.
- The training of the ML model is computationally efficient on a massively parallel computer.
- Compared to a numerical (physics-based) model, the ML model performs best for the state variables most affected by parameterized processes.

Corresponding author: Troy Arcomano, troyarcomano@tamu.edu

Abstract

The paper investigates the applicability of machine learning (ML) to weather prediction by building a reservoir-computing-based, low-resolution, global prediction model. The model is designed to take advantage of the massively parallel architecture of a modern supercomputer. The forecast performance of the model is assessed by comparing it to that of daily climatology, persistence, and a numerical (physics-based) model of identical prognostic state variables and resolution. Hourly resolution 20-day forecasts with the model predict realistic values of the atmospheric state variables at all forecast times for the entire globe. The ML model outperforms both climatology and persistence for the first three forecast days in the midlatitudes, but not in the tropics. Compared to the numerical model, the ML model performs best for the state variables most affected by parameterized processes in the numerical model.

1 Introduction

The ultimate goal of our research is to develop a *hybrid* (numerical-machine-learning) *weather prediction (HWP)* model. We hope to achieve this goal by implementing algorithms developed by *Pathak et al.* [2018a,b] and *Wikner et al.* [2020]: the first paper introduced an efficient ML algorithm for numerical-model-free prediction of large, spatiotemporal dynamical systems, based solely on the knowledge of past states of the system; the second paper showed how to combine a machine learning (ML) algorithm with an imperfect numerical model of a dynamical system to obtain a hybrid model that predicts the system more accurately than either component alone; while the third paper combined the techniques of the first two into a computationally efficient hybrid modeling approach. The present paper implements the parallel ML technique of *Pathak et al.* [2018a] to build a model that predicts the weather in the same format as a global numerical model. We train and verify the model on hourly ERA5 reanalysis data from the European Centre for Medium-Range Weather Forecasts (ECMWF) [*Hersbach et al.*, 2019].

The work presented here can also be considered an attempt to develop a ML model that can predict the evolution of the three-dimensional, multivariate, global atmospheric state. To the best of our knowledge, the only similar prior attempts were those by *Scher* [2018] and *Scher and Messori* [2019], but they trained their three-dimensional multivariate ML model on data that was produced by low-resolution numerical model simulations. In addition, *Dueben and Bauer* [2018] and *Weyn et al.* [2019, 2020] designed ML models to predict two-dimensional, horizontal fields of select atmospheric state variables. Similar to our verification strategy, they also verified the ML forecasts against reanalysis data. Compared to all of the aforementioned studies, an important new aspect of our work is that we employ *reservoir computing (RC)* [*Jaeger*, 2001; *Maass et al.*, 2002; *Lukoševičius and Jaeger*, 2009; *Lukoševičius*, 2012] rather than *deep-learning* [e.g. *Goodfellow et al.*, 2016], which is primarily motivated by the significantly lower computer wall-clock time required to train an RC-based model. This difference in training efficiency would allow for a larger number of experiments to tune the ML model at higher resolutions.

The structure of the paper is as follows. Section 2 describes the ML model, while section 3 presents the results of the forecast experiments, using as benchmarks persistence of the atmospheric state, climatology, as well as numerical forecasts from a physics-based model of identical prognostic state variables and resolution. Section 4 summarizes our conclusions.

2 The ML model

The N components of the state vector $\mathbf{v}^m(t)$ of the ML model are the grid-point values associated with the spatially discretized fields of the Eulerian dependent variables of the model. Training the model requires the availability of a discrete time series of past *observation-based estimates* (analyses) $\mathbf{v}^a(k\Delta t)$ $(k = -K, -K+1, \ldots, 0)$ of the atmospheric states that use the same N-dimensional representation of the state as the model. Beyond the training period, the analyses $\mathbf{v}^a(k\Delta t)$ $(k = 1, 2, \ldots)$ are used only to maintain the synchronization of the model state with the observed atmospheric state. An ML forecast can potentially be started at any analysis time $k\Delta t$ $(k = 0, 1, \ldots)$: the forecast is a discrete time series of model states $\mathbf{v}^m_k(k'\Delta t)$ $(k' = k+1, k+2, \ldots)$, where $k\Delta t$ is the *initial time*, $\mathbf{v}^a(k\Delta t)$ is the *initial state*, Δt is the *time-step*, and $(k'-k)\Delta t$ is the *forecast time*. The computational algorithm of the model is designed to take advantage of a massively parallel computer architecture.

2.1 Representation of the Model State

2.1.1 The Global State Vector

We define $\mathbf{v}^m(t)$ by the grid-based state vector of the physics-based numerical model SPEEDY [*Molteni*, 2003; *Kucharski et al.*, 2013]. While SPEEDY is a spectral transform model, it uses the grid-based state vector to represent the input and output state of the model, and to compute the nonlinear and parameterized terms of the physics-based prognostic equations. The horizontal grid spacing is $3.75^\circ \times 3.75^\circ$[1] and the model has $n_v = 8$ vertical σ-levels (at σ equals 0.025, 0.095, 0.20, 0.34, 0.51, 0.685, 0.835, and 0.95), where σ is the ratio of pressure to the pressure at the surface. The model has four three-dimensional dependent variables (the two horizontal coordinates of the wind vector, temperature, and specific humidity) and one two-dimensional dependent variable (the logarithm of surface pressure). Thus the number of variables per horizontal location is $n_t = 4 \times n_v + 1$. Because there are $n_h = 96 \times 48 = 36,864$ horizontal grid points, the total number of model variables is $N = n_t \times n_h = 1.52064 \times 10^5$. Before forming the state vector $\mathbf{v}^m(t)$, we standardize each state variable by subtracting its climatological mean and dividing by its standard deviation at the particular model level in the local region.

2.1.2 Local State Vectors

The global model domain is partitioned into $L = 1,152$ local regions. We use a Mercator (cylindrical) map projection to define the local regions, partitioning the three-dimensional model domain only in the two horizontal directions: each local region has the shape of a rectangular prism with a $7.5^\circ \times 7.5^\circ$ base (Fig. 1). The model state in local region ℓ $(\ell = 1, 2, \ldots, L)$ is represented by the *local state vector* $\mathbf{v}^m_\ell(t)$, whose components are defined by the $D_v = 4 \times n_t = 132$ components of the global state vector in the local region. The model computes the L evolved local state vectors $\mathbf{v}^m_\ell(t + \Delta t)$ from $\mathbf{v}^m(t)$ in parallel, and the evolved global state vector $\mathbf{v}^m(t + \Delta t)$ is obtained by piecing the L evolved local state vectors together.

2.2 The Computational Algorithm

2.2.1 RC

The computation of $\mathbf{v}^m_\ell(t+\Delta t)$ from $\mathbf{v}^m(t)$ requires the evaluation of a composite (chain) function for each local state vector. Because we use an RC algorithm, this

[1] This corresponds to about 300 km in the midlatitudes, which is an order of magnitude larger than the 10-30 km grid spacing of a state-of-the-art operational forecast model.

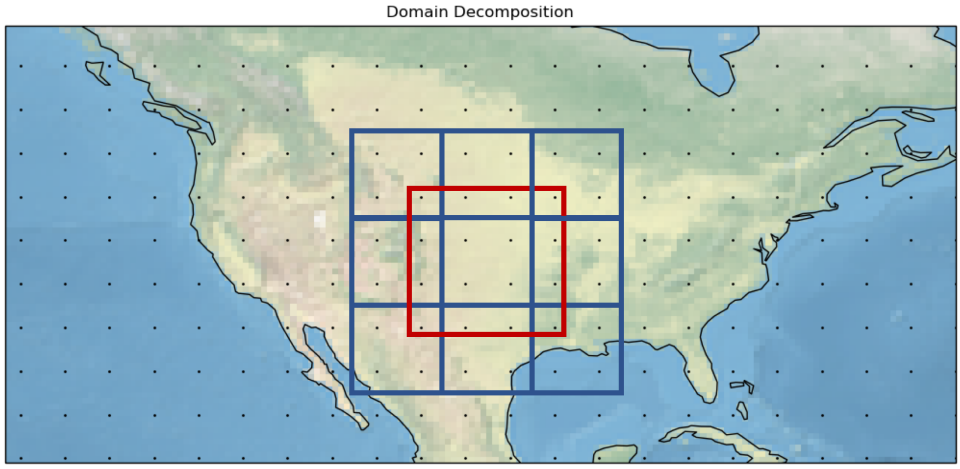

Figure 1. Illustration of the local regions. The local regions are defined on a Mercator map projection, where the black dots indicate the horizontal location of the grid-points of the model. The blue rectangles mark the boundaries of nine adjacent local regions. The red rectangle indicates the boundaries of the extended local region for the local region in the center.

composite function has only three layers: the *input layer, the reservoir,* and the *output layer.* A key feature of RC is that the trainable parameters of the model appear only in the output layer, which greatly simplifies the training process.

2.2.2 The Input Layer and Reservoir

The composite of the input layer and the reservoir is

$$\mathbf{r}_\ell(t+\Delta t) = \mathbf{G}_\ell\{\mathbf{r}_\ell(t), \mathbf{W}_{in,\ell}[\hat{\mathbf{v}}_\ell^m(t)]\}, \tag{1}$$

where the function $\mathbf{W}_{in,\ell}[\cdot]$ is the input layer. The dimension D_r of the *reservoir state vector* $\mathbf{r}_\ell(t) = (r_{\ell,1}, r_{\ell,2}, \ldots, r_{\ell,D_r})$ is much higher than the dimension $D_{\hat{v}}$ of the input vector $\hat{\mathbf{v}}_\ell^m(t)$. (The reservoir is a high-dimensional dynamical system.) The input vector $\hat{\mathbf{v}}_\ell^m(t)$ is an *extended local state vector* that represents the model state in an extended local region. In the present paper, we define $\hat{\mathbf{v}}_\ell^m(t)$ by the grid points of local region ℓ plus the closest grid points from the neighboring local regions (see Fig. 1 for an illustration). In the terminology of *Pathak et al.* [2018a], the *locality parameter* of our model is 1. Using a nonzero value of the locality parameter is essential, because otherwise no information can flow between the local regions. The dimension of the extended local state vectors is $D_{\hat{v}}$=16×n_t = 528 for most ℓ. The exceptions are the local regions nearest to the two poles, because for those, we add no extra grid points in the poleward direction. The dimension of the input vectors in these local regions is $D_{\hat{v}} = 12 \times n_t = 396$.

The 'local approach' of *Dueben and Bauer* [2018], which was introduced independently of the parallel technique of *Pathak et al.* [2018a], employs a localization strategy that is formally similar to the one described here. There is, however, an important difference between the two localization techniques: *Dueben and Bauer* [2018] trained a single common neural network for the different local regions, while we train a different reservoir for each local region.

The input layer is implemented as $\mathbf{W}_{in,\ell}[\hat{\mathbf{v}}_\ell^m(t)] = \mathbf{W}_{\hat{v},\ell}\hat{\mathbf{v}}_\ell^m(t)$, where $\mathbf{W}_{\hat{v},\ell}$ is a sparse $D_r \times D_{\hat{v}}$ random matrix, whose entries are drawn from a uniform probability

distribution in the interval $[-0.5, 0.5]$. The reservoir dynamics is defined by

$$\mathbf{G}_\ell\{\mathbf{r}_\ell(t), \mathbf{W}_{in,\ell}[\hat{\mathbf{v}}_\ell^m(t)]\} = \tanh\left[\mathbf{A}_\ell \mathbf{r}_\ell(t) + \mathbf{W}_{\hat{v},\ell}\hat{\mathbf{v}}_\ell^m(t)\right], \tag{2}$$

where $\tanh[\cdot]$ is the component-wise hyperbolic tangent function and $\mathbf{A}_\ell$ is a $D_r \times D_r$ *weighted adjacency matrix* that represents a low-degree, directed, random graph [*Gilbert*, 1959]. Each entry of $\mathbf{A}_\ell$ has a probability κ/D_r of being nonzero, so that the expected degree of each vertex is a prescribed number κ. Thus, κ is the average number of incoming connections (edges) per vertex. The nonzero entries of $\mathbf{A}_\ell$ are randomly drawn from a uniform distribution in the interval $(0, 1]$ and scaled so that the largest eigenvalue of $\mathbf{A}_\ell$ is a prescribed number ρ. The parameter ρ, which controls the length of the memory of the ML model dynamics, is called the *spectral radius*.

2.2.3 The Output Layer

The evolved local state vector is obtained by

$$\mathbf{v}_\ell^m(t + \Delta t) = \mathbf{W}_{out,\ell}[\mathbf{r}_\ell(t + \Delta t), \mathbf{P}_\ell], \tag{3}$$

where the function $\mathbf{W}_{out,\ell}[\cdot, \cdot]$ is the output layer. This function is chosen such that it is linear in the $D_v \times D_r$ *matrix of trainable parameters* $\mathbf{P}_\ell$. To be precise,

$$\mathbf{W}_{out,\ell}[\mathbf{r}_\ell(t + \Delta t), \mathbf{P}_\ell] = \mathbf{P}_\ell \tilde{\mathbf{r}}_\ell(t + \Delta t), \tag{4}$$

where $\tilde{\mathbf{r}}_\ell(t + \Delta t) = (r_{\ell,1}, r_{\ell,2}^2, r_{\ell,3}, r_{\ell,4}^2, \ldots, r_{\ell,D_r-1}, r_{\ell,D_r}^2)(t + \Delta t)$.

2.2.4 Synchronization and Training

We define the *local analysis* $\mathbf{v}_\ell^a(k\Delta t)$ by the components of the global analysis $\mathbf{v}^a(k\Delta t)$ $(k = -K, -K + 1, \ldots)$ that describe the state in local region ℓ. In other words, $\mathbf{v}_\ell^a(k\Delta t)$ is the observation-based estimate of the desired value of the model state $\mathbf{v}_\ell^m(k\Delta t)$. Likewise, we define the *extended local analysis* $\hat{\mathbf{v}}_\ell^a(k\Delta t)$ as the observation-based estimate of the extended local state vector $\hat{\mathbf{v}}_\ell^m(k\Delta t)$ $(k = -K, -K + 1, \ldots)$.

The synchronization and training of the ML model starts with feeding the past analyses to the reservoir, or more precisely, by substituting $\hat{\mathbf{v}}_\ell^a(k\Delta t)$ $(k = -K, -K + 1, \ldots, -1)$ for $\hat{\mathbf{v}}_\ell^m(k\Delta t)$ in Eq. (1). Thus the output layer, Eq. (3), is not needed to compute $\mathbf{r}_\ell(k\Delta t)$ for $k = -K + 1, -K + 2, \ldots, 0$: we generate $\mathbf{r}_\ell(-K\Delta t)$ randomly, discard the transient sequence $\mathbf{r}_\ell(k\Delta t)$, $k = -K, -K+1, \ldots, -K_t$, and define $\mathbf{v}_\ell^m(k\Delta t)$ for $k = -K_t + 1, -K_t + 2, \ldots, 0$ according to Eq. (1), with $\mathbf{P}_\ell$ as yet undetermined.

The goal of the training is to find the $\mathbf{P}_\ell$ that minimizes the cost function

$$J_\ell(\mathbf{P}_\ell) = \left[\sum_{k=-K_t+1}^{0} \|\mathbf{v}_\ell^a(k\Delta t) - \mathbf{v}_\ell^m(k\Delta t)\|^2\right] + \beta\|\mathbf{W}_{out,\ell}\|), \qquad \ell = 1, 2, \ldots, L, \tag{5}$$

where $\|\cdot\|$ is the Frobenius norm. The purpose of the Tikhonov regularization term $\beta\|\mathbf{W}_{out,\ell}\|)$ [*Tikhonov et al.*, 1977] of $J_\ell(\mathbf{P}_\ell)$ is to improve the numerical stability of the computations and prevent overfitting to the training data by choosing large values of the components of $\mathbf{W}_{out,\ell}$. Because $\mathbf{W}_{out,\ell}$ depends linearly on $\mathbf{P}_\ell$, the solutions of the L minimization problems can be obtained by a linear *ridge regression*. That is, $\mathbf{P}_\ell$ is computed by solving the linear problem

$$\mathbf{P}_\ell\left(\tilde{\mathbf{R}}_\ell\tilde{\mathbf{R}}_\ell^T + \beta\mathbf{I}\right) = \mathbf{V}_\ell^a\tilde{\mathbf{R}}_\ell^T, \qquad \ell = 1, 2, \ldots, L, \tag{6}$$

where the columns of $\tilde{\mathbf{R}}_\ell$ are $\tilde{\mathbf{r}}_\ell(k\Delta t)$ $(k = -K_t + 1, -K_t + 2, \ldots, 0)$ and the columns of $\mathbf{V}_\ell^a$ are $\mathbf{v}_\ell^a(k\Delta t)$ $(k = -K_t + 1, -K_t + 2, \ldots, 0)$. Notice that the dimension of the linear problem of Eq. (6) does not depend on the length K_t of the training period. To

conserve memory, the $D_r \times K_t$ matrix $\mathbf{R}_\ell$ need not be stored; the $D_r \times D_r$ matrix $\tilde{\mathbf{R}}_\ell \tilde{\mathbf{R}}_\ell^T$ and the $D_v \times D_r$ matrix $\mathbf{V}_\ell^a \tilde{\mathbf{R}}_\ell^T$ can be built incrementally, passing the training data through the reservoir time-step by time-step [e.g., *Lukoševičius and Jaeger*, 2009; *Lukoševičius*, 2012].

2.3 Implementation on ERA5 Reanalysis Data

2.3.1 Training

The global analyses $\mathbf{v}^a(k\Delta t)$ ($k = -K, -K+1, \dots$) are hourly ERA5 reanalyses interpolated to the computational grid and adjusted[2] to the topography of SPEEDY. The training starts at 0000 UTC[3] 1 January, 1981 and ends at 2000 UTC January 24, 2000 ($K \approx 1.66 \times 10^5$). We add a small-magnitude random noise $\boldsymbol{\varepsilon}(t)$ to $\hat{\mathbf{v}}_\ell^a(k\Delta t)$ ($k = -K, -K+1, \dots, -1$) before we substitute it for $\hat{\mathbf{v}}_\ell^m(t)$ in Eq. (1) in order to improve the robustness of the ML model to noise [*Jaeger*, 2001]. The transient sequence of $K - K_t$ discarded reservoir states corresponds to the first 43 days of training.

2.3.2 Code Implementation and Performance

The current computer code of the ML model is written in Fortran, using both MPI and OpenMP for parallelization and the LAPACK routine DGESV to solve the linear problem of Eq. (6). The computations of both the training and forecast phase are carried out on 1,152 Intel Xeon E5-2670 v2 processors. Training the model takes 67 minutes wall-clock time and requires 2.2 Gb of distributed memory per processor.[4] Our current code is designed to minimize the wall-clock execution time given the available memory on a particular supercomputer, but the memory usage could be reduced (e.g., by not keeping all training data in memory simultaneously, or using single- rather than double-precision arithmetic).

2.4 The Forecast Cycle

Beyond the training period, the analyses are used only to maintain the synchronization between the reservoirs and the atmosphere. We use the hourly reanalyses for synchronization, but start a new 20-day forecast only once every 48 hours. (Preparing a 20-day forecast takes about 1 minute of wall-clock time.) We prepare a total of 171 forecasts for the period from January 25, 2000 to 28 December, 2000. The forecast error statistics reported below are calculated based on these forecasts.

2.4.1 Selection of the Hyperparameters

The dimension D_r of the reservoir, rank κ of the random network, spectral radius ρ, random noise $\boldsymbol{\varepsilon}$, and regularization parameter β are the *hyperparameters* of the RC algorithm. We found suitable combinations of these parameters by numerical experimentation, monitoring the accuracy and stability of the forecasts. All results reported in this paper are for D_r=9,000, κ=6, $\beta = 10^{-5}$, while ρ monotonically increases from 0.3 at the equator to 0.7 at 45° and beyond. The components of $\boldsymbol{\varepsilon}$ are uncorrelated, normally distributed, random numbers with mean zero and standard deviation 0.28.

[2] The adjustment is done to the surface pressure field to insure that it is consistent with the SPEEDY orography [for the adjustment scheme, see *Baek et al.*, 2009].

[3] Universal Time Coordinated, known as Greenwich Mean Time (GMT) prior to 1972.

[4] The memory usage per processor is vastly smaller in the forecast phase.

For this combination of the hyperparameters, the ML model predicts realistic values of all state variables for the entire globe and 20-day forecast period.[5]

3 Forecast Verification Results

3.1 Benchmark Forecasts

We use daily climatology, persistence, and numerical forecasts for the evaluation of the ML model forecasts. Persistence is based on the assumption that the initial atmospheric state will persist for the entire time of the forecast. The numerical forecasts are prepared by Version 42 of the SPEEDY model. While SPEEDY has been developed for research applications rather than weather prediction, it can be considered a low-resolution version of today's NWP models. Most importantly, similar to all operational models, it solves the system of atmospheric primitive equations and has a realistic climate. It provides a good benchmark in the current stage of our research, in which the primary goal is to prove a concept rather than improve operational forecasts.

3.2 Results

We verify all forecasts against ERA5 reanalyses interpolated to the computational grid and adjusted to the SPEEDY orography. The magnitude of the forecast error is measured by the mean of the area-weighted root-mean-square difference between the forecasts and the verification data for all forecasts. Results are shown for selected variables in the Northern Hemisphere (NH) midlatitudes for the first 72 forecast hours (Fig. 2). In this region, the ML model outperforms both persistence and climatology by a large margin in the first 48 forecast hours. While the ML model forecasts remain more accurate than persistence in the next 24 forecast hours, their skill, with the exception of the temperature forecasts, degrades to that of climatology. In the tropics (results not shown) the accuracy of the ML model is very similar to that of persistence and climatology

The performance of the ML model compared to SPEEDY is mixed: the ML forecasts are more accurate for the specific humidity near the surface, especially at 24 h and 48 h forecast times, while the SPEEDY forecasts are more accurate for the wind, particularly at the jet level. The ML temperature forecasts are also more accurate in the tropics (results not shown), where the SPEEDY forecasts rapidly develop a large bias in the upper troposphere.

To better understand the behavior of the root-mean-square error, we decomposed it into a (square of) bias and variance component and also investigated the power spectrum of the variance in the NH midlatitudes with respect to the zonal wavenumber (results are not shown). On the positive side, the ML forecasts of the different variables have little or no bias, and the variance of the longer term forecasts saturates at a realistic level for zonal wave numbers larger than 6. On the negative side, the variance saturates at unrealistically high levels at the lower wave numbers, leading to an overprediction of the spatial variability of the forecast fields at the longer forecast times. The fast growth of the variance at the large scales, especially at wave number 4, is the main deficiency of ML model in the midlatitudes. Fixing this problem could extend the time range of forecast skill by days.

[5] A small improvement of the forecast accuracy for the first three days could be achieved at the expense of stability beyond 15 days.

3.3 Near-Surface Humidity and Tropical Temperature Profiles

The short-term forecast advantage of the ML model over SPEEDY has two sources. First, while the SPEEDY forecasts rapidly develop a near-surface humidity bias, the ML model forecasts are free of such bias. Second, the variance of the ML model forecast errors is also lower initially. As forecast time increases, the advantage of the ML model remains in terms of the bias, but vanishes in terms of the variance. Because the variance becomes the dominant component at the later forecast times, climatology breaks even with the ML model forecasts by 72 h forecast time (bottom right panel of Fig. 2). The spatial distribution of the difference of the errors (Fig. 3) suggests that the ML model performs better in regions where parameterized atmosphere-surface interactions play an important role in the moist processes in SPEEDY (e.g., regions of the ocean boundary currents). Likewise, the advantage of the ML model in predicting the tropical temperature profiles (not shown) is the result of large biases that are present only in the SPEEDY forecasts in the main regions of parameterized deep convection. Finally, it should be noted that while the current version of the ML model learns about atmosphere-surface interactions strictly from the atmospheric training data, SPEEDY uses a number of prescribed fields to describe the surface conditions (e.g., a spatio-temporally evolved sea-surface temperature analysis.)

3.4 Rossby Wave Propagation

The forecast variable for which SPEEDY clearly outperforms the ML model is the meridional component of the wind: while the accuracy of the wind forecasts by the two models is similar at 24 h, the error of the ML model forecasts grows more rapidly beyond that time. The difference between the errors of the two models grows the fastest in the layer around the jet streams of the Northern Hemisphere (NH) midlatitudes (between 400 hPa and 200 hPa). Because the variability of the meridional wind in this layer is dominated by dispersive synoptic-scale Rossby waves, the aforementioned result suggests that the ML model may be inferior to the numerical model in describing the Rossby wave dynamics. To investigate this possibility, we plot Hovmöller diagrams of the meridional wind for both forecasts and the verification data (Figure 4).

A pattern of negative (positive) values followed by a pattern of positive (negative) values indicate a trough (ridge). Because the eastward group velocity of the dispersive Rossby waves at the synoptic scales is larger than their eastward phase velocity, new troughs and ridges can develop downstream of the original wave. Such developments are marked by oriented dashed black lines in the figure. In the first three days, the ML model captures the dispersive dynamics of the wave packets accurately, but because the wave packets are composed of wave number 4-11 waves [e.g. *Zimin et al.*, 2003], the over-intensification of the wave number 4-6 components at the later forecast times leads to a gradual shift of the carrier wave number toward lower values and a deceleration of the group velocity.

4 Conclusions

We demonstrated that a RC-based parallel ML model can predict the global atmospheric state in the same gridded format as a numerical (physics-based) global weather prediction model. We found that the 20-day ML model forecasts predicted realistic values of all state variables at all forecast times for the entire globe. The ML model predicted the weather in the midlatitudes more accurately than either persistence or climatology for most of the first three forecast days. This time range could be significantly extended by eliminating, or at least reducing, the over-prediction of atmospheric spatial variability at the large scales (wave numbers lower than 7). The forecast variables for which the ML model performed best compared to a numerical

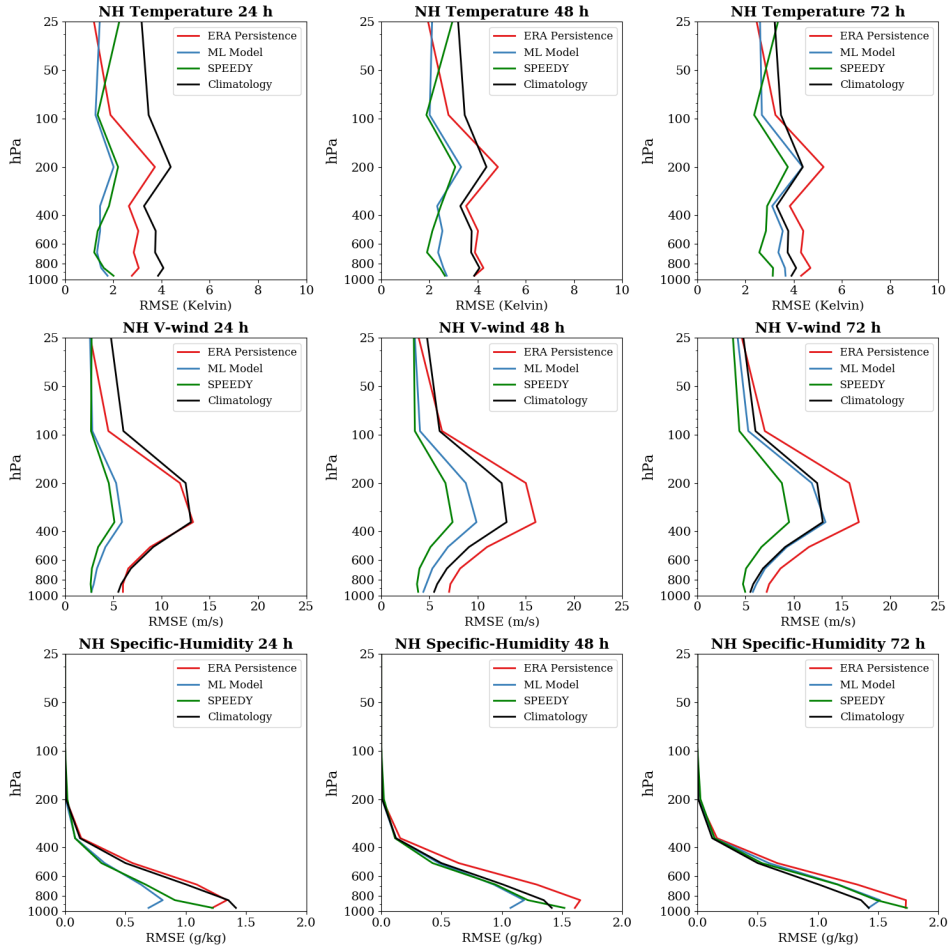

Figure 2. Forecast verification results for the NH midlatitudes (30°N and 70°N). Results are shown for (blue) the ML model, (green) SPEEDY, and (red) persistence. Shown is the area-weighted root-mean-square error at the different atmospheric levels for (top row) the temperature, (middle row) meridional wind, and (bottom row) specific humidity at (left column) 24 h forecast time, (middle column) 48 hour forecast time, and (right column) 72 h forecast time.

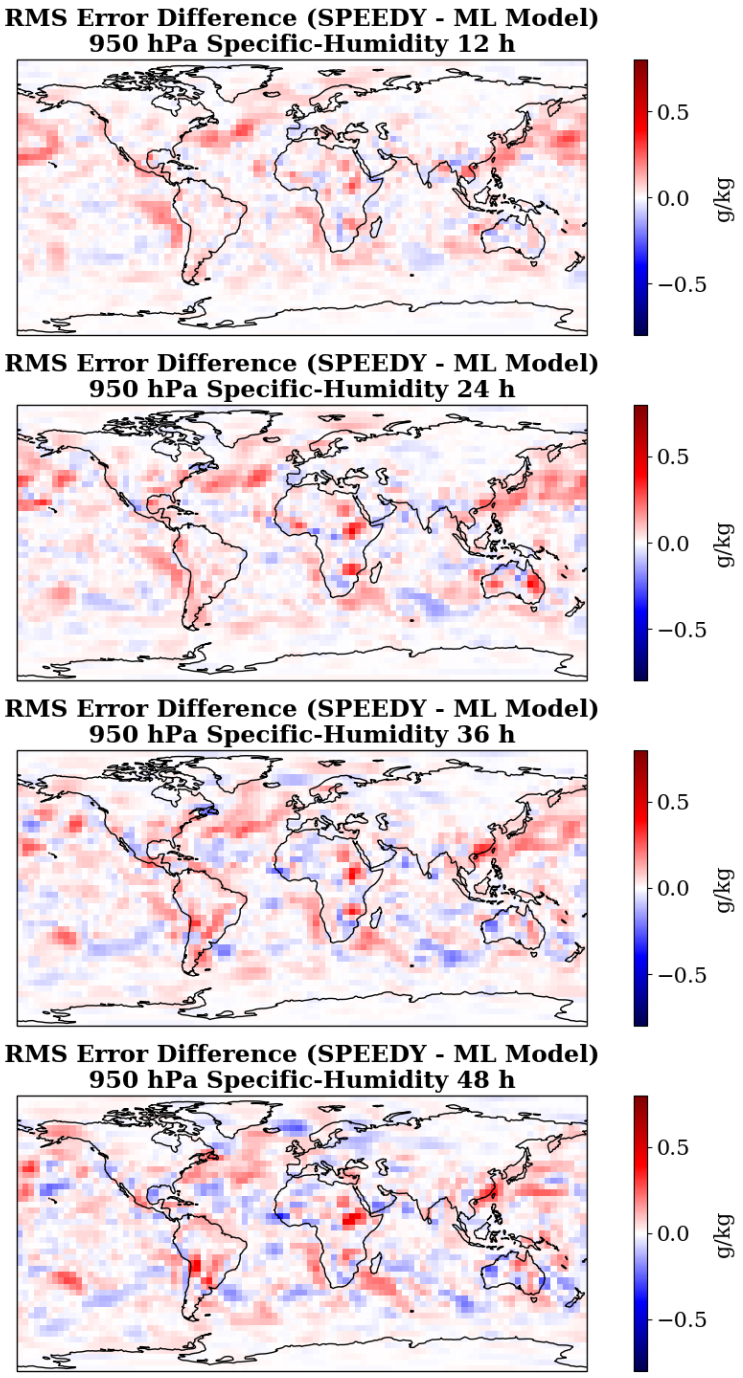

Figure 3. Comparison of the near-surface humidity forecast errors between the SPEEDY and ML model forecasts. Shown by color shades is the difference of the 925 hPa relative humidity root-mean-square errors between the SPEEDY and ML model forecasts at forecast times (top) 12 h, (second from top) 24 h, (second from bottom) 36 h, and (bottom) 48 h. Here, the mean is taken over all 171 forecasts. Positive (negative) values indicate locations where the ML model forecasts are more (less) accurate than the SPEEDY forecasts.

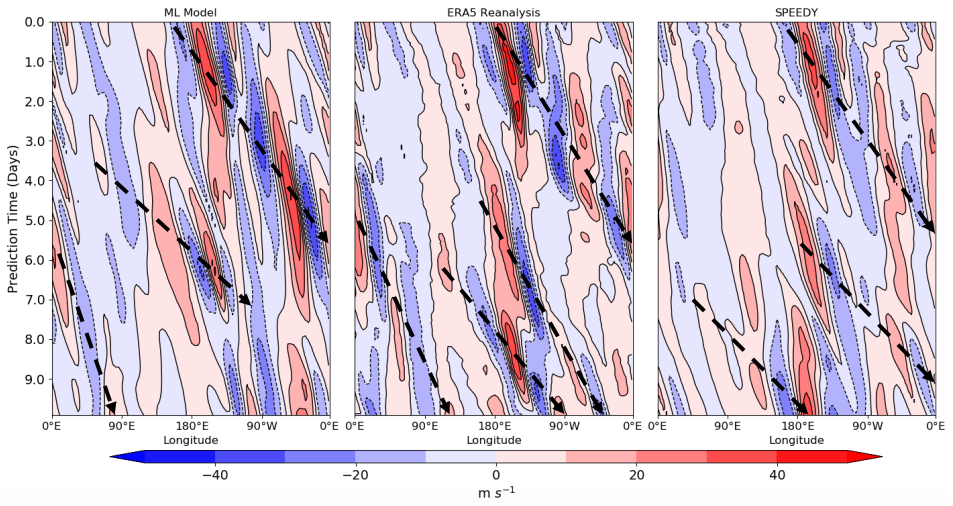

Figure 4. Rossby wave packets in the model forecasts and verification data. The Hovmöller diagrams show the propagation of waves packets in (left) a 10-day ML model forecast, (middle) related verification data, and (right) related 10-day SPEEDY forecast. Shown by color shades is the latitude-weighted meridional mean of the meridional coordinate of the wind for latitude band 30°N-60°N at 200 hPa. The forecasts start at 0000 UTC 2 December, 2000. The propagation of the wave packets are marked by the directed straight dashed lines.

(physics-based) model of identical prognostic state variables and resolution were the ones most affected by parameterized processes in the numerical model.

The results suggests that the current version of our ML model have potential in short-term weather forecasting. Because the parallel computational algorithm is highly scalable, it could be easily adapted to higher spatial resolutions on a larger supercomputer. As the algorithm is highly efficient in terms of wall-clock time, it could be used for rapid forecast applications and could also be implemented in a limited-area rather than a global setting. The ML modeling technique described here could also be applied to other geophysical fluid dynamical systems.

Acknowledgments

This work was supported by DARPA contract DARPA-PA-18-01 (HR111890044). The work of T. A. and I. S. was also supported by ONR award N00014-18-2509. This research were conducted with the advanced computing resources provided by Texas A&M High Performance Research Computing. This paper greatly benefitted from stimulating discussions with Sarthak Chandra, Michelle Girvan, Garrett Katz, and Andrew Pomerance. Peter Dueben of ECMWF and an anonymous reviewer provided valuable comments to improve the paper. The new data generated for the paper is available at http://doi.org/10.5281/zenodo.3712157.

References

Baek, S., Szunyogh, I., Hunt, B. R., and Ott, E. (2009). Correcting for surface pressure background bias in ensemble-based analyses. *Monthly Weather Review, 137,* 2349–2364.

Dueben, P. D., and Bauer, P. (2018). Challenges and design choices for global weather and climate models based on machine learning. *Geoscientific Model Development, 11,* 3999–4009.

Gilbert, E. (1959). Random graphs. *Annals of Mathematical Sciences 30,* 1141–1144.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning,* MIT Press, Cambridge, MA, USA.

Jaeger, H. (2001). *The "echo state" approach to analyzing and training recurrent neural networks.* GMD Report 148, German National Research Center for Information Technology.

Lukoševičius, M. (2012). A practical guide to applying echo state networks. *Neural Networks: Tricks of the Trade*, eds. G. Montavon, G. B. Orr, and K.-R. Müller, 2nd edition, Springer, 659–686.

Lukoševičius, M., and Jaeger, H. (2009). Reservoir computing approaches to recurrent neural network training. *Computer Science Review, 3,* 127–149.

Kucharski, F., Molteni, F., King, M. P., Farneti, R., Kang, I., Feudale, L. (2013). On the need of intermediate complexity general circulation models: A "SPEEDY" example. *Bulletin of the American Meteorological Society, 94,* 25–30.

Hersbach, H., Bell, B., Berrisford, P., Horányi, A., Sabater, J. M., Nicolas, J., Radu, R., Schepers, D., Simmons, A., Soci, C., Dee, D. (2019). Global reanalysis: goodby ERA-Interim, hello ERA. *ECMWF Newsletter 159,* 17–24.

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,* 2531–2560.

Molteni, F. (2003). Atmospheric simulations using a GCM with simplified parameterizations I: model climatology and variability in multi-decadal experiments. *Climate Dynamics, 20,* 175–191.

Pathak, J., Wikner, A., Fussell, R., Chandra, S., Hunt, B. R., Girvan, M., and Ott, E. (2018). Hybrid forecasting of chaotic processes: Using machine learning in conjunction with a knowledge-based model. *Chaos, 28,* 041101.

Pathak, J., Hunt, B. R., Girvan, M., Lu, Z., and Ott, E. (2018). Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach *Physical Review Letters, 120,* 024102.

Scher, S., and 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* 2797–2809.

Scher, S. (2018). Toward data-driven weather and climate forecasting: Approximating a simple general circulation model with deep learning. *Geoscientific Model Development, 12,* 2797–2809.

Tikhonov, A. N., Arsenin, V.I., and John, F. (1997). *Solutions of Ill-Posed Problems,* Winston, Washington, DC, USA.

Weyn, J. A., Durran, D. R., and Caruana, R. (2020). Improving data-driven global weather prediction using deep convolutional neural networks on a cubed sphere. https://doi.org/10.1002/essoar.10502543.1.

Weyn, J. A., Durran, D. R., and 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,* 2680–2693.

Wikner, A., Pathak J., Hunt, B., Girvan M., Arcomano, T., Szunyogh, I., Pomerance, A., and Ott, E. (2019). Combining machine learning with knowledge-based modeling for scalable forecasting and subgrid-scale closure of large, complex, spatiotemporal systems. *Chaos* (in press).

Zimin, A. V., Szunyogh, I., Patil, D. J., Hunt, B., and Ott, E. (2003). Extracting envelopes of Rossby wave packets *Monthly Weather Review, 131,* 1011–1017.