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 reservoir-computing-based low-resolution machine learning model can skillfully predict the global atmospheric state for 3-5 days.
- The short-term ML model forecasts are more accurate than the physics-based model forecasts for the low-level humidity.
- The training of the ML model is computationally highly efficient.

Corresponding author: Troy Arcomano, troyarcomano@tamu.edu

Abstract

The paper investigates the applicability of machine learning (ML) to weather prediction by building a low-resolution ML model for global weather prediction. The forecast performance of the ML model is assessed by comparing it to that of persistence and a numerical physics-based model, whose prognostic state variables and resolution are identical to those of the ML model. The ML model typically provides realistic prediction of the weather for the entire globe for about five forecast days. For the first three forecast days, the ML model outperforms persistence in the extratropics. While the relative performance of the ML model compared to the physics-based model is mixed, the ML forecasts are more accurate for the specific humidity in the extratropics and the specific humidity and temperature in the tropics. These results suggests that ML has a potential to improve the prediction of state variables most affected by parameterized processes in numerical models.

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] 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] 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 is a potential key advantage at higher spatial 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, as well as numerical forecasts from a physics-based model of the same 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*.

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 $3.75^\circ \times 3.75^\circ$ and the model has eight 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. On this computational grid, the three-dimensional field of an Eulerian dependent variable is represented by $96 \times 48 \times 8 = 36{,}864$ grid-point variables. The model has four three-dimensional dependent variables: the two horizontal coordinates of the wind vector, temperature, and specific humidity. The logarithm of surface pressure is also a dependent model variable, but it is represented by a two-dimensional (horizontal) field. Thus the number of variables per horizontal location is $n = 4 \times 8 + 1 = 33$, while the total number of model variables is $N = n \times 4{,}608 = 1.52064 \times 10^5$. Because the different Eulerian state variables have different units and their values vary in different ranges, they should be standardized before forming the state vector $\mathbf{v}^m(t)$. We subtract the climatological mean and divide by the standard deviation of each variable in the local region for the standardization.

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 $D_v = 4 \times 33 = 132$ 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 composite function has only three layers: the *input layer, the reservoir,* and the *output*

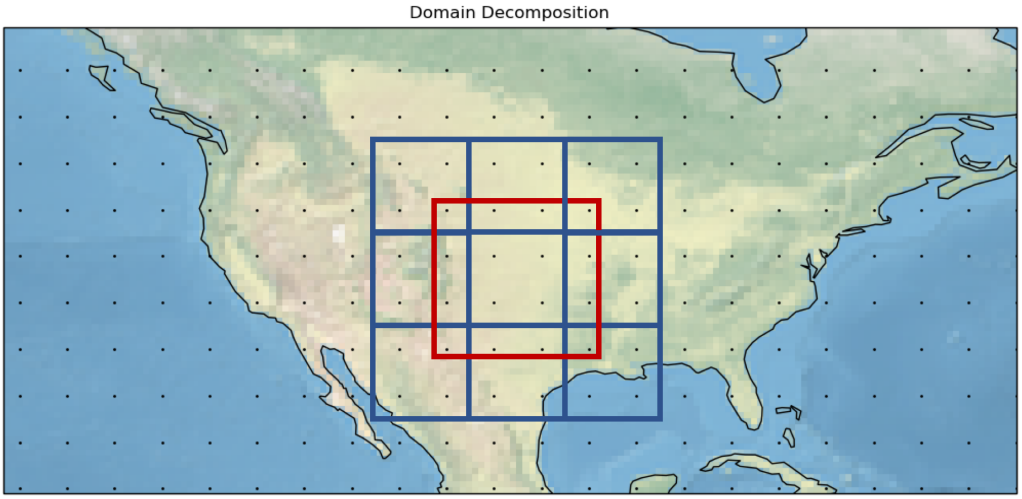

Figure 1. Illustration of the local regions. The local regions are defined on a Mercator map projection, where the black dots indicate the horizontal location of the grid-points of the model. The blue rectangles mark the boundaries of nine adjacent local regions. The red rectangle indicates the boundaries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

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 $\hat{\mathbf{v}}_\ell^m(t)$. (Notice that 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. (We note that the 'local' approach of *Dueben and Bauer* [2018] that was introduced independently of the parallel technique of *Pathak et al.* [2018a] employs a conceptually identical localization strategy.) The dimension of the extended local state vectors is $D_{\hat{v}}$=16×33=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 $D_{\hat{v}} = 12 \times 33 = 396$.

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 $D_r \times D_{\hat{v}}$ 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, determining the average number of components of $\mathbf{r}_\ell(t)$ that can affect a component of $\mathbf{r}_\ell(t+\Delta t)$. 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 is called the *spectral radius* and it controls the length of the memory of the ML model dynamics.

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 *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 training is done by finding the values of the entries of $\mathbf{P}_\ell$ that minimize 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 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 $\mathbf{v}_\ell^a(k\Delta t)$ $(k = -K_t+1, -K_t+2, \ldots, 0)$.

2.3 Implementation on ERA5 Reanalysis Data

2.3.1 Training

The global analyses $\mathbf{v}^a(k\Delta t)$ $(k = -K, -K+1, \ldots)$ are hourly ERA5 reanalyses. We interpolate the reanalysis fields onto the computational grid of the model by first

interpolating the horizontal fields at each pressure level by a 2-dimensional quadratic B-spline interpolation. As the local height of the topography has a major effect on the surface pressure, we adjust the reanalysis surface pressure fields to the SPEEDY topography after completing the horizontal interpolation. The adjustment is done by making the assumption of hydrostatic balance and taking into account the local height difference between the SPEEDY and ERA5 topography [for details, see *Baek et al.*, 2009; *Herrera et al.*, 2018]. Because the vertical coordinate of SPEEDY is σ, the vertical interpolation of the fields requires first computing sigma at each horizontal grid point locations for the reanalysis fields. Then, we interpolate the reanalysis fields to the sigma levels of SPEEDY by a 1-dimensional cubic B-spline interpolation.

This training starts at 0000 UTC[1] on 1 January, 1981 and ends at 2000 UTC on January 24, 2000 ($K \approx 1.66 \times 10^5$). We add a small-magnitude random noise $\boldsymbol{\varepsilon}(t)$ to $\hat{\mathbf{v}}_\ell^a(k\Delta t)$ ($k = -K, -K+1, \ldots, -1$) 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 $K - K_t$ reservoir states that we discard corresponds to the first 43 days of the training data set.

2.3.2 Code Implementation and Performance

Following the convention of numerical weather prediction (NWP), the computer code of the model is written in Fortran, using both MPI and OpenMP for parallelization. The linear problem of Eq. (6) is solved by the LAPACK routine DGESV. All computations are carried out on 1,152 Intel Xeon E5-2670 v2 processors in an Ivy Bridge configuration. Training the ML model takes 67 minutes wall-clock time and requires 2.5 Terabytes of memory.

2.4 The Forecast Cycle

Beyond the training period, the analyses are used only to maintain the synchronization between the reservoirs and the atmosphere. While we use the hourly reanalyses for synchronization, we start a new 5-day forecast only once every 48 hours. (Preparing a 5-day forecast takes about 15 seconds 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 parameters l, D_r, κ, ρ, $\boldsymbol{\varepsilon}$, and β are the *hyperparameters* of the model. Our choice of $l = 1$ is primarily dictated by the limited amount of available computer memory. We found suitable combinations of the other hyperparameters 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, $\beta = 10^{-4}$, 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 0.2 in the tropics (between 30°S and 30°N) and 0.25 elsewhere. For this combination of the hyperparameters, the ML model typically produces realistic forecasts for about five days. Beyond that time, the model remains stable (the errors are bounded), but the fields develop unrealistic atmospheric features.

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

3 Forecast Verification Results

3.1 Benchmark Forecasts

We use two sets of benchmark forecasts for the evaluation of the ML model forecasts. The first set is based on the assumption that the initial state of the atmosphere will persist for the entire time of the forecast. Persistence has a long history of use as a benchmark to decide whether a forecast technique (model) has forecast value. Beating persistence by a model forecast has been a nontrivial task in the history of atmospheric modeling. For instance, of the four celebrated ENIAC forecasts [*Charney et al.*, 1950], which were the first successful experimental numerical weather forecasts, all had a larger mean error, and three had a larger root-mean-square error than persistence [*Lynch*, 2008].

The second set of benchmark forecasts are numerical forecasts 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 state-of-the-art NWP models. Most importantly, similar to all operational models, it solves the system of atmospheric primitive equations and has a realistic climate. It provides an ideal 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 on the SPEEDY grid, using properly interpolated ERA5 reanalyses as the proxy for the true atmospheric state. We measure the magnitude of the forecast errors by the area-weighted root-mean-square difference between the forecasts and the verification data in the NH midlatitudes (Fig. 2) and the tropics (Fig. 3). The root-mean square error is calculated and showed separately for the air temperature, meridional (south-north) coordinate of the wind and specific humidity at the different vertical (pressure) levels. (The small number of forecasts that developed unrealistic values of the specific forecast variable are not included in the computation of the root-mean-square error, but their number is indicated in each panel.) Each row of panels is for a particular state variable at forecast times 24-h, 48-h, and 72-h.

The results suggest that similar to SPEEDY, the ML model outperforms persistence by a significant margin in the extratropics, but not in the tropics. While the overall relative performance of the ML model compared to SPEEDY is mixed, the ML forecasts are more accurate for the specific humidity near the surface in both the extratropics and tropics, especially at the shorter forecast times. The ML forecasts are also more accurate for the temperature in the tropics. The difficulties of SPEEDY in predicting humidity near the surface and temperature in the tropics is not surprising considering that these forecast variables are strongly affected by physical processes that are either not included or parameterized by highly simplified schemes in SPEEDY. The fact that the data-driven ML model can greatly improve upon these forecasts is one of the most encouraging results of our study.

3.3 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 absolute 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 verifying reanalyses (Figure 4).

The axes of troughs and ridges associated with the individual waves lie along the lines that separate the regions of positive and negative values: a pattern of negative (positive) values followed by a pattern of positive (negative) values indicate a trough (ridge). Because at the synoptic scales the eastward group velocity is larger than the eastward phase velocity, new troughs and ridges can develop downstream of the original wave. In Figure 4, four such developments can be observed, each marked by a straight black line and labeled by a capitol letter. The ML model captures the dispersive wave dynamics qualitatively correctly and also provides realistic prediction of the phase and group velocities. There are even some features of the wave packets (center panel) that the ML model (left panel) predicts more accurately than the numerical model (right panel). An example is the intensification of wave packet A from day 3 to day 5.

4 Conclusions

We have demonstrated that a RC-based parallel ML model can predict the global atmospheric state (weather) in the same gridded format as a numerical global weather prediction model. For the tested parameters, the ML model was able to realistically predict the atmospheric state for about five days. Based on our experience with lower dimensional systems, we believe that this time limit could be extended by increasing the dimension D_r of the reservoir states. Overall, the ML model predicts the weather more accurately than persistence and is competitive with a numerical model of the same resolution for the first three forecast days. The forecast variables for which the ML model outperforms the numerical model are those that are influenced the most by parameterized processes in the numerical model. Preliminary results with a model that implements the hybrid scheme of *Wikner et al.* [2020] on our ML model and SPEEDY suggest that the hybrid approach will lead to further significant improvement of the short-term forecast performance.

Our results also suggests that RC-based ML models 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 system for which a high quality training data set is available.

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. Portions of 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. 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–

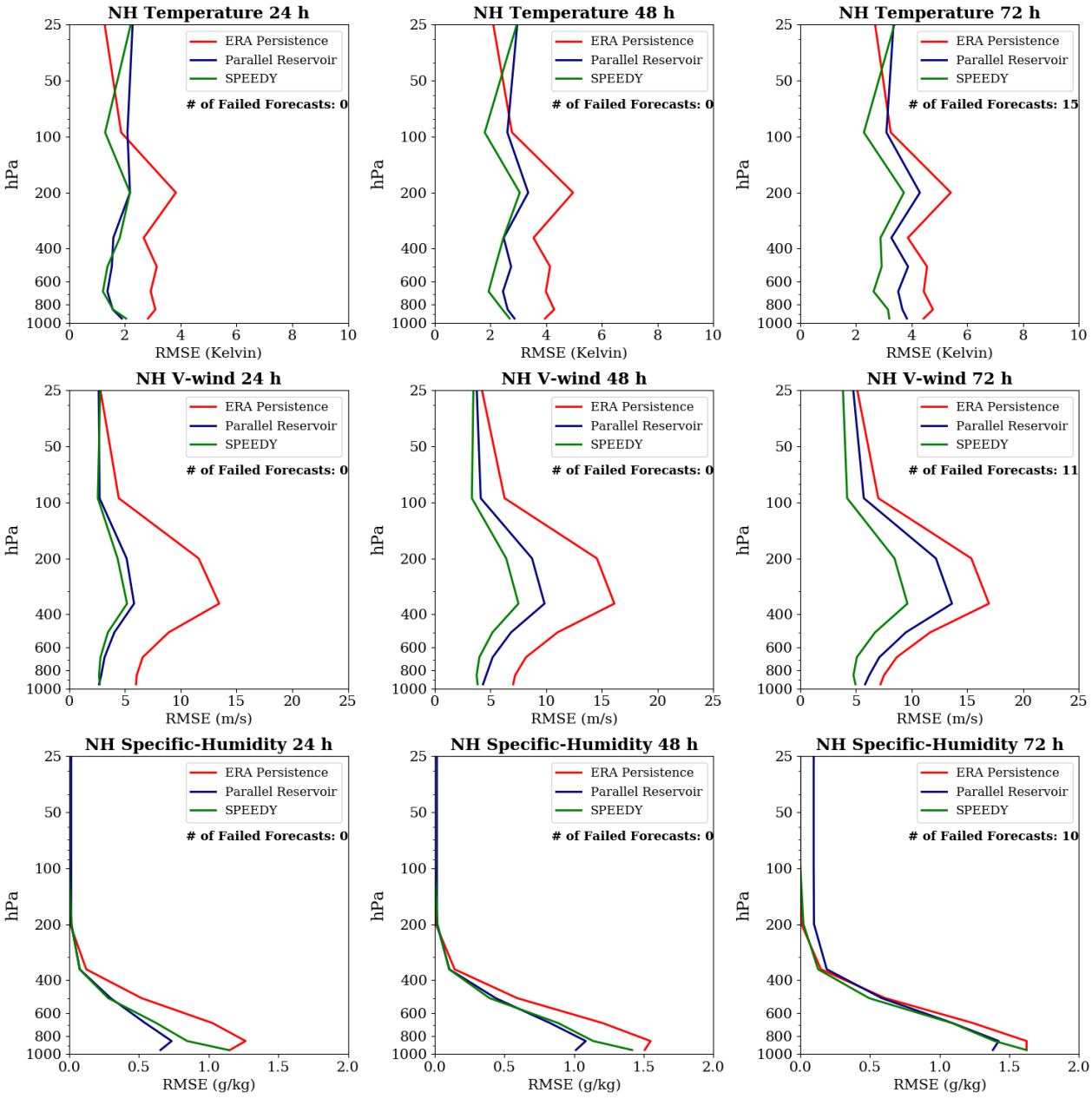

Figure 2. Forecast verification results for the NH midlatitudes (30°N and 70°N). Results are shown for (blue) the ML model, (green) SPEEDY, and (red) persistence. Shown is the area-weighted root-mean-square error at the different atmospheric levels for (top row) the temperature, (middle row) meridional 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. The number of forecasts that developed unrealistic values of the particular forecast variable at the specific forecast time is indicated in each panel.

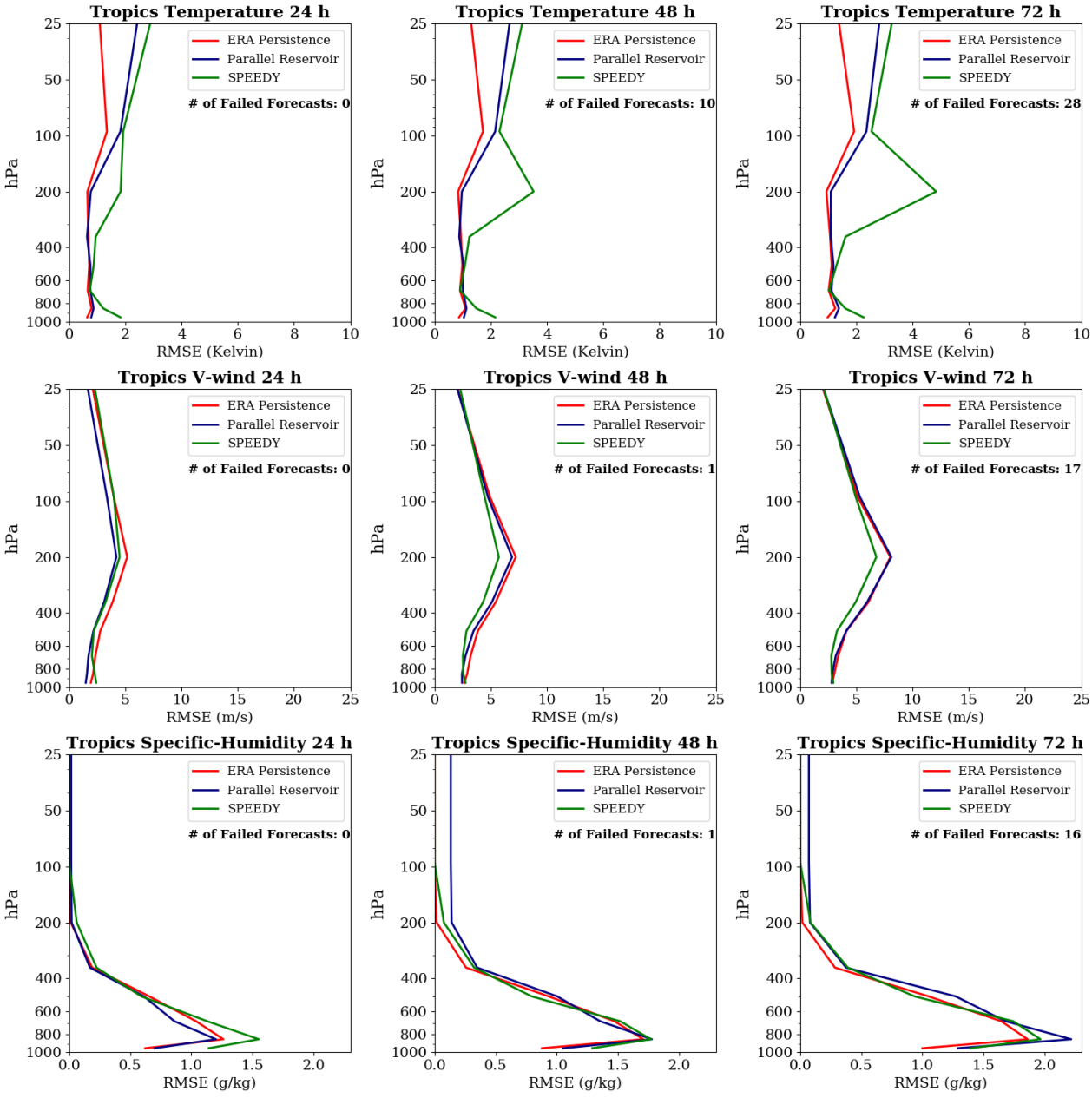

Figure 3. Same as Fig. 2, but for the tropics (30°S and 30°N)

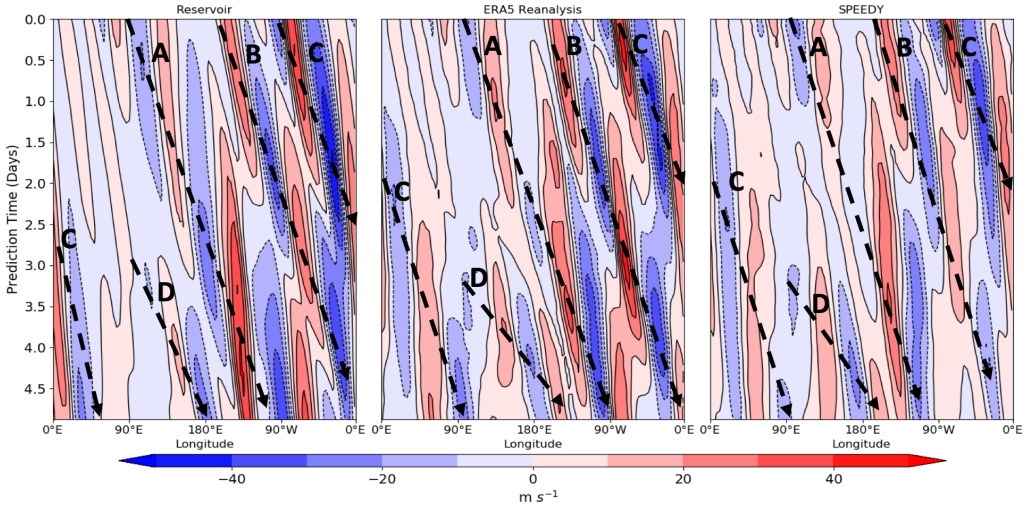

Figure 4. Examples for the dispersive propagation of synoptic-scale Rossby waves in the model forecasts and the reanalyses. The Hovmöller diagrams show the propagation of the same waves packets in (left) the ML model forecast, (middle) reanalyses, and (right) SPEEDY forecast. Shown by color shades is the latitude-weighted meridional mean of the meridional coordinate of the wind for latitude band 30°N-60°N at 200 hPa. The four wave packets present are marked by the straight black lines and distinguished by the different capital letters.

2364.

Charney, J. G., Fjortoft, R., von Neumann, J. (1949). Numerical integration of the barotropic vorticity equation. *Journal of Meteorology 67,* 371–385.

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.

Herrera, M., S., Szunyogh, I., Brainard, A., Kuhl, D. D., Hoppel, K., Bishop, C. H., Holt, T. R., and Zhao, Q. (2018). Regionally enhanced global (REG) 4D-Var. *Monthly Weather Review, 146,* 4015–4038.

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

Lukosevicius, 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.

Lynch, P. (2008). The ENIAC Forecasts: A Re-creation textitBulletin of the American Meteorological Society, 89, 45–56.

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. (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. *arXiv:2002.05514*.