Can a Machine Learning–Enabled Numerical Model Help Extend Effective Forecast Range through Consistently Trained Subgrid-Scale Models?

YONGQUAN QU[a] AND XIAOMING SHI[a,b]

[a] *Division of Environment and Sustainability, Hong Kong University of Science and Technology, Hong Kong, China*
[b] *Center for Ocean Research in Hong Kong and Macau, Hong Kong University of Science and Technology, Hong Kong, China*

(Manuscript received 5 July 2022, in final form 12 November 2022)

ABSTRACT: The development of machine learning (ML) techniques enables data-driven parameterizations, which have been investigated in many recent studies. Some investigations suggest that a priori-trained ML models exhibit satisfying accuracy during training but poor performance when coupled to dynamical cores and tested. Here we use the evolution of the barotropic vorticity equation (BVE) with periodically reinforced shear instability as a prototype problem to develop and evaluate a model-consistent training strategy, which employs a numerical solver supporting automatic differentiation and includes the solver in the loss function for training ML-based subgrid-scale (SGS) turbulence models. This approach enables the interaction between the dynamical core and the ML-based parameterization during the model training phase. The BVE model was run at low, high, and ultrahigh (truth) resolutions. Our training dataset contains only a short period of coarsened high-resolution simulations. However, given initial conditions long after the training dataset time, the trained SGS model can still significantly increase the effective lead time of the BVE model running at the low resolution by up to 50% relative to the BVE simulation without an SGS model. We also tested using a covariance matrix to normalize the loss function and found it can notably boost the performance of the ML parameterization. The SGS model's performance is further improved by conducting transfer learning using a limited number of discontinuous observations, increasing the forecast lead-time improvement to 73%. This study demonstrates a potential pathway to using machine learning to enhance the prediction skills of our climate and weather models.

SIGNIFICANCE STATEMENT: Numerical weather prediction is performed at limited resolution for computational feasibility, and the schemes to estimate unresolved processes are called parameterization. We propose a strategy to develop better deep learning–based parameterization in which an automatic differentiable numerical solver is employed as the dynamic core and interacts with the parameterization scheme during its training. Such a numerical solver enables consistent deep learning, because the parameterization is trained with a direct target of making the numerical model (dynamic core and parameterization together) forecast match observations as much as possible. We demonstrate the feasibility and effectiveness of such a strategy using a surrogate model and advocate that such machine learning–enabled numerical models provide a promising pathway to developing next-generation weather forecast and climate models.

KEYWORDS: Barotropic flows; Numerical weather prediction/forecasting; Deep learning

1. Introduction

Many of the physical processes involved in weather or climate simulations occur at spatial scales smaller than the grid spacings of numerical models. These processes, such as radiative transfer, cumulus convection, and turbulent mixing, can drive heat and momentum budgets and therefore are critical to a numerical model's predicting skill (Bauer et al. 2015). Because explicitly resolving those physical processes are computationally infeasible, we need parameterization schemes that approximate subgrid-scale physical processes using resolvable variables and incorporate the resulting tendencies into the numerical integration. Traditional turbulence parameterization models are developed based on some heuristic assumptions, whose quality and reliability are in question (Chung and Matheou 2014). For example, the eddy viscosity assumption postulates that resolved scalar variances and kinetic energy are always transferred to smaller unresolved scales, but in reality, backscatter can occur at all scales to transfer variance and energy in the opposite direction (Chow et al. 2019). Second, when developing traditional parameterization schemes, the processes to be parameterized are sometimes evaluated independently without allowing a scheme to interact with other processes or the dynamical core (Donahue and Caldwell 2018). Such an artificial separation is also a source of numerical errors and uncertainties (Gross et al. 2018).

In recent years, the development of deep learning (DL) has opened up more opportunities for weather and climate research. Some efforts were made to investigate the feasibility of replacing the entire numerical model with a machine learning (ML) model. For example, MetNet-2, a large-context neural network, outperformed the state-of-the-art, physics-based High-Resolution Ensemble Forecast (HREF) for up to 12 h of lead time in terms of precipitation forecasting (Espeholt et al. 2022). However, for

Qu's current affiliation: Department of Earth and Environmental Engineering, Columbia University, New York City, New York.

Corresponding author: Yongquan Qu, yq2340@columbia.edu

DOI: 10.1175/AIES-D-22-0050.1 e220050

short and medium range and climate change time scales, in which physical principles should be well represented, pure ML methods still fall behind numerical models due to a lack of physical consistency (Weyn et al. 2019). Therefore, using DL methods to augment numerical models is seemingly more attractive and of greater potential for now.

Deep learning–based parameterization is one such application and has gained much attention recently. Deep learning parameterizations can be developed in an a priori setting, that is, learning the map between resolved-scale variables and the tendencies due to subgrid-scale processes directly from training targets and inputs computed from high-resolution benchmark simulations. This approach has been applied to improve the representation of turbulent fluxes in the surface layer (Leufen and Schädler 2019), gravity wave drag (Chantry et al. 2021), moist processes, and a collection of subgrid-scale processes (Gentine et al. 2018; Brenowitz et al. 2020). Those schemes performed well when evaluated offline; however, in online testing, when they are incorporated into numerical models, there are some issues, such as numerical instability and mean state drifting (Brenowitz et al. 2020) and violation of conservation laws (Gentine et al. 2018). Enforcing physical constraints in DL architecture can improve this situation (Beucler et al. 2021); however, those constraints can trigger model instabilities in some cases (Chantry et al. 2021). Overall, a priori training strategy suffers from a physics–dynamics decoupling problem like traditional parameterization. This inconsistent behavior of offline-trained DL models underlines the importance of appropriately incorporating the interaction between DL parameterizations and resolvable physics, dynamical core, and physical principles.

Differentiable programming is one of the potential solutions to make DL parameterizations more tightly coupled to their hosting dynamical core by incorporating the gradient of a numerical solver into back propagation and enabling the gradient-based optimization of the algorithm with DL parameterizations and the dynamical core as a whole (Baydin et al. 2018). Um et al. (2020) developed such a framework using TensorFlow for DL training and Φ_{Flow} for a differentiable partial differentiable equations solver. Kochkov et al. (2021) further integrate the DL models and the numerical method in a "JAX" framework, which supports automatically differentiating native Python and NumPy functions and accelerates them on GPUs or TPUs (Bradbury et al. 2018). The latter feature further accelerates the dynamical core. Kochkov et al. (2021) demonstrated that their learned interpolation method, a DL-based numerical integration approach, takes less time than traditional numerical methods but exhibits the same level of accuracy and numerical stability, suggesting a promising future of applying hybrid DL-differentiable dynamical core in NWP.

However, from Kochkov et al. (2021), it is unclear whether the advantages of DL-based parameterization can be converted to a significant extension of the effective lead time of NWP. Because they used the Kolmogorov flow for their experiments, the flow is highly turbulent and has a limited predictable range. Different from the intensively turbulent Kolmogorov flow, the real atmosphere is more of a mixture of geostrophic turbulence and waves (Vallis 2017) and exhibits quasi-periodic behaviors such as the Madden–Julian oscillation (Zhang 2005) and baroclinic annular mode (Thompson and Barnes 2014). We design a new case with better predictability than the Kolmogorov flow but which also contains an instability mechanism that is periodically activated. We will investigate whether the DL training with a differentiable dynamical core can significantly extend the effective forecast range in this case.

The advantage of using high-resolution simulations for DL training with automatic differentiation is that we can have spatially dense data and make the time interval between reference states relatively short, thus reducing the difficulty of training. To extend the lead time of effective prediction, naïvely, we want to include longer segments of high-resolution simulation data in the training of DL-based parameterization. However, this can become computationally expensive when many consecutive steps from the high-resolution benchmark simulation are included. In addition, high-resolution simulations are different from nature and may become unreliable for long-term prediction. With the development of Earth system observation technologies, we have accumulated a lot of spatiotemporally sparse observation data. Using these observation data for posttraining enhancement of a DL parameterization model through transfer learning appears to be a potentially valuable idea.

This study focuses on studying the strategy of developing subgrid-scale turbulence parameterization with DL models, which are coupled with a differentiable dynamical core during model training. The first issue we want to investigate is the performance of the online training approach and some strategies to improve the DL model training with high-resolution benchmark simulation data. The second question we want to answer is whether and how we can adjust the pretrained DL model using limited observational data to achieve better generalizability or further performance improvement. Below, we first introduce our idealized case and its predictability issue in section 2a. DL training data, models, and strategies are introduced in sections 2b and 2c. Results are presented for pretraining and transfer learning in section 3, followed by a summary and discussion in section 4.

2. Experimental design and methods

a. *An idealized case with BVE*

The barotropic vorticity equation (BVE) describes the conservation of vorticity in a two-dimensional flow. Here, we design a new case with better predictability than Kolmogorov flow by periodically suppressing the forcing to the flow.

1) GOVERNING EQUATIONS

The governing equation for the nondimensionalized BVE model is

$$\frac{\partial \zeta}{\partial t} + J(\psi, \zeta) = -\nu\nabla^4\zeta + f, \tag{1}$$

where $\zeta = \nabla^2\psi$ is the vorticity, ψ is the streamfunction, and $J(\psi, \zeta) = \psi_x\zeta_y - \psi_y\zeta_x = v\zeta_y + u\zeta_x$ is the advection of vorticity. The two terms on the right-hand side are hyperdiffusion and external forcing, where ν is the kinematic viscosity. The forcing

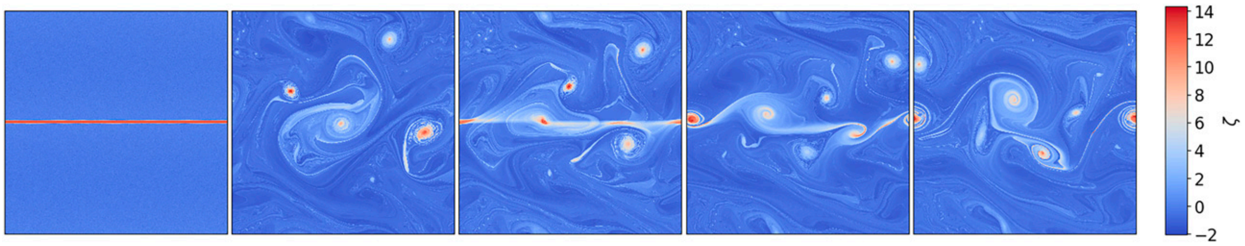

FIG. 1. Snapshots of vorticity in a DNS of the BVE case at t = (left) 0, (left center) 202.5, (center) 205.0, (right center) 207.5, and (right) 210.0 time units, with forcing being strongest at 205.0.

term determines the characteristics of the flow. In our simulations, the forcing f is periodic in time and y direction. When it is active, it re-establishes the strong shear zone in the middle of the domain and provides sufficient conditions for the Kelvin–Helmholtz instability. The forcing term is configured as a damping term to the streamfunction. It can be conceptually written as f_ψ in the following equation,

$$\frac{\partial\psi}{\partial t} = \mathscr{G}(\psi) + f_\psi, \tag{2}$$

where $\mathscr{G}(\psi)$ is the grid-resolved tendency of the streamfunction and

$$f_\psi = -\alpha(\psi - \psi_0). \tag{3}$$

Here α^{-1} is the (e folding) damping time scale for the deviation of the streamfunction from the initial condition ψ_0. The damping coefficient is

$$\alpha = \frac{1}{2}\left[\frac{1-\cos(\pi t/5)}{2}\right]^4 \left\{\frac{24}{25}\left[\frac{1-\cos(y)}{2}\right]^4 + \frac{1}{25}\right\}, \tag{4}$$

which makes the damping term have a period of 10 time units and be 24 times as large in the middle ($y = \pi$) as in the north/south edge ($y = 0, 2\pi$).

The initial condition comprises an isolated shear zone in the middle of the domain, which is a prototype situation of barotropic instability (Vallis 2017). We created this initial condition by prescribing the initial vorticity,

$$\zeta_0(x,y) = \begin{cases} -\dfrac{32}{63}\sin\left(\pi - \dfrac{64}{63}y\right) + \epsilon(x,y) & 0 \le y < \dfrac{63}{64}\pi \\ \dfrac{64}{\pi} + \epsilon(x,y) & \dfrac{63}{64}\pi \le y \le \dfrac{65}{64}\pi, \\ -\dfrac{32}{63}\sin\left(\dfrac{64}{63}y - \dfrac{65}{63}\pi\right) + \epsilon(x,y) & \dfrac{65}{64}\pi < y \le 2\pi \end{cases} \tag{5}$$

where ϵ is uniformly distributed random noise with amplitude between $\pm 0.05 \times \pi/64$.

The natural atmosphere behaves like waves in the absence of instability and therefore is of substantial predictability. However, baroclinic, convective, or other instability may occur at some time and location, exhibiting quasi-periodic behaviors (Zhang 2005; Thompson and Barnes 2014). Figure 1 shows the vorticity field during one cycle of our forcing. When the forcing is weak, the flow is dominated by larger eddies and may be less dependent on subgrid-scale process (SGS) parameterization. When the forcing is active, the SGS process may be primarily unresolved in the middle shear zone; there are also regimes where eddies are partially resolved, so they are more like the gray zones of NWP. This forcing seems to be a better choice than having a laminar flow or intensively turbulent flow as an analogy for the actual weather forecast.

2) PREDICTABILITY LIMIT

To further illustrate the different predictability of the Kolmogorov flow and our BVE setting, we designed a predictability limit test based on different simulation configurations. We used a pseudospectral spatial differentiation and exponential time differencing Runge–Kutta fourth-order (ETDRK4) method for time integration (Kassam and Trefethen 2005). In this test, we performed four simulation groups, TRUTH, ALT, HighRes, and LowRes, which were also used later in the machine learning part. The TRUTH group is a direct numerical simulation (DNS) run on a 1024 × 1024 grid on a domain of $2\pi \times 2\pi$. The first 200 time units of the TRUTH run were discarded, and we collected one snapshot every 2.5 time units for a total of 100 time slices, which are used as initial conditions for other groups of simulations. The ALT, meaning alternative truth, is also run on the 1024 × 1024 grid but carries Fourier-space vorticity each integration step, whereas the TRUTH group carries physical-space vorticity states. The only difference between TRUTH and ALT is a roundoff error, and thus the latter represents an upper limit on the ability to predict TRUTH with a lower-resolution, parameterized simulation. The HighRes group runs on 256 × 256 grids, which stands for high-resolution simulations in NWP, which has a high accuracy for short time periods; however, it has a high computational cost and a low accuracy for long time periods. The LowRes group, representing the operational model in NWP, runs fast on 64 × 64 grids but needs good SGS models for a breakthrough in the lead time of prediction.

Figure 2 compares the predictability of the Kolmogorov flow and the BVE with periodic shear forcing through the root-mean-square error (RMSE), R^2 (with R being the correlation coefficient), and instantaneous vorticity field for simulations with one initial condition. The forcing in the Kolmogorov flow follows the setting in Kochkov et al. (2021), which has a

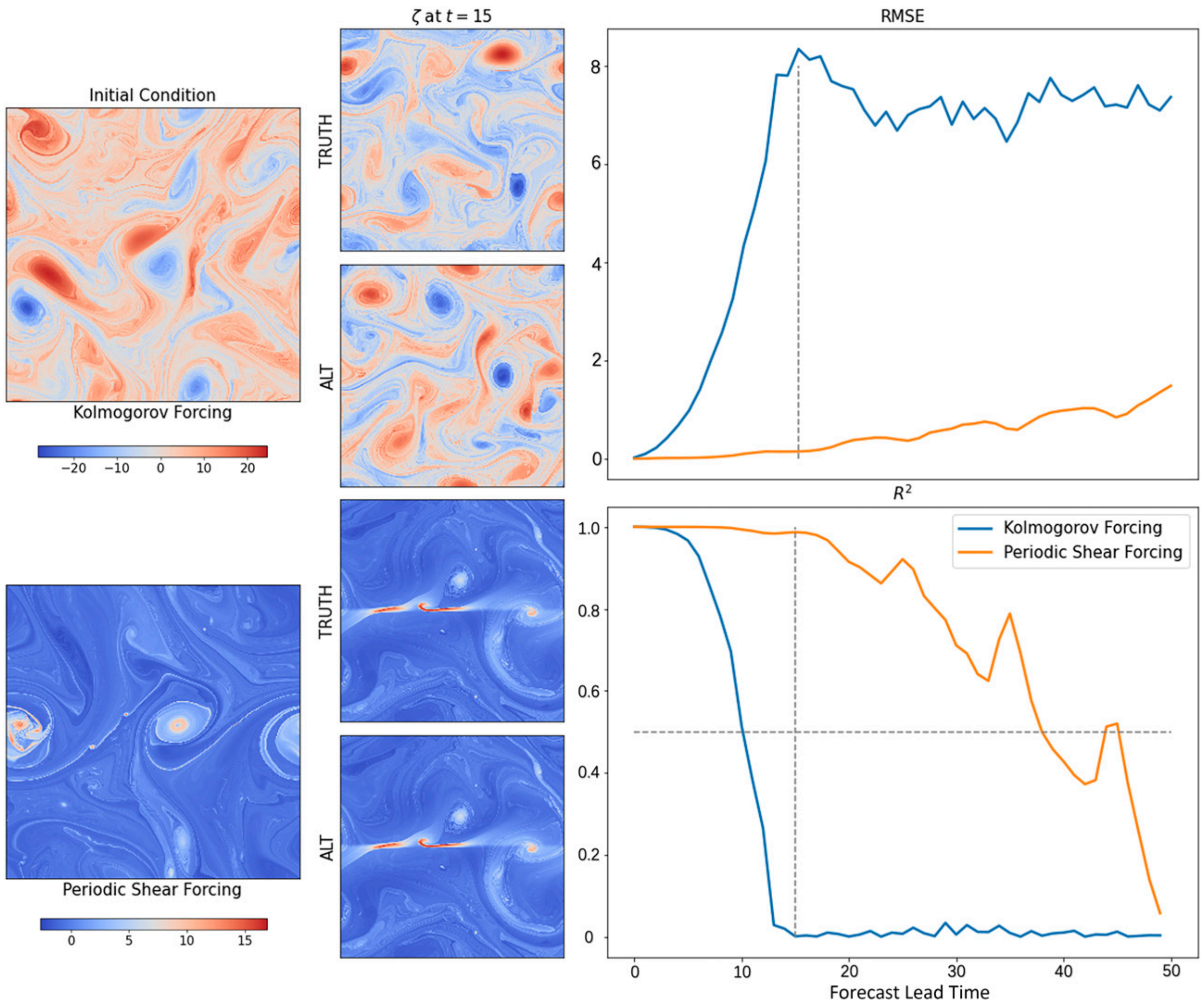

FIG. 2. (left) Initial conditions for simulations, (center) vorticity fields at $t = 15$ time units for the TRUTH and ALT simulations, and (right) RMSE and R^2, with R being the correlation coefficient, over time for both forcings when using ALT to predict TRUTH.

wavenumber-4 structure in the y direction and creates constant (in time) forcing for zonal wind shears. It is so turbulent that within about 10 time units, even at the highest accuracy in our study, rounding errors can develop into significant differences, making it unsuitable as an analogy for numerical weather forecasting. In contrast, the BVE with periodic forcing has better predictability with slower error growth and correlation decreasing. Unresolved or poorly resolved scales change the flow after a relatively longer period than the Kolmogorov flow.

Figure 3 shows the average error growth and R^2 curves as well as their standard deviation of 100 Alt, HighRes, and LowRes simulations, respectively, for the periodic shear forcing setup. We arbitrarily set the threshold for a forecast to remain valid as having R^2 greater than 0.5. The effective forecast leading time for the LowRes models is around 11 time units (~1 forcing cycle), which is to be improved through DL methods using HighRes simulations, which have an effective forecast leading time around 21 time units (~2 forcing cycles). The upper limit of the predictability for our BVE problem is about 37 time units based on the ALT simulations. Having these significantly different effective lead times makes it easy to quantify the benefits of DL-based parameterizations with regard to improving forecasting performance.

b. SGS processes

SGS processes refer to the dynamics and physics not resolved by a numerical model. The governing equation [Eq. (1)] on the coarse-grid mesh can be written as

$$\frac{\partial \overline{\zeta}}{\partial t} + J(\overline{\psi}, \overline{\zeta}) = -\nu \nabla^4 \overline{\zeta} + f(\overline{\psi} - \overline{\psi_0}) + \tau, \tag{6}$$

where the overline denotes filtering of the DNS data onto the LowRes grid mesh. The term τ represents the tendency of the SGS processes,

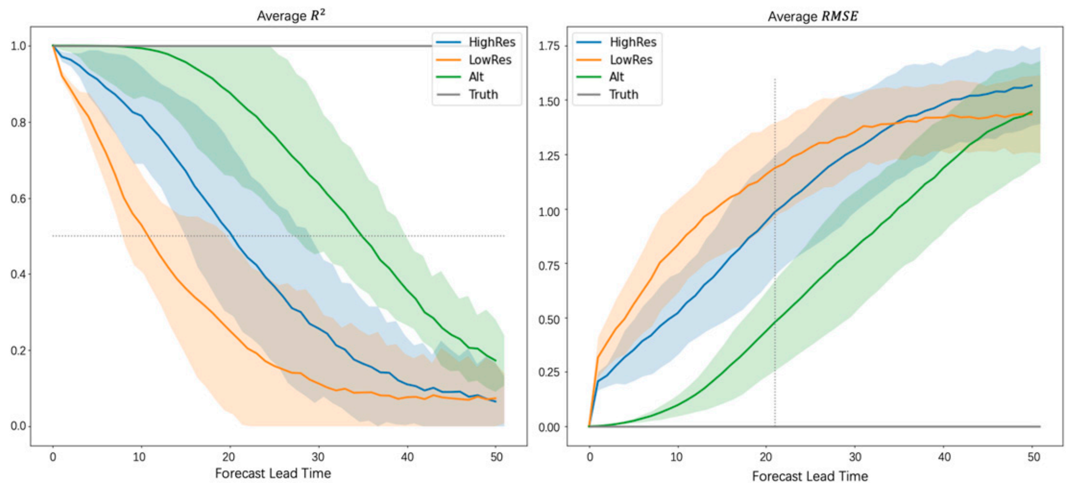

FIG. 3. The averaged (right) RMSE and (left) R^2 curves over time for ALT, HighRes, and LowRes simulations of BVE with periodic shear forcing to predict the TRUTH.

$$\tau = -[\overline{J(\psi,\zeta)} - J(\overline{\psi},\overline{\zeta})] + [\overline{f(\psi-\psi_0)} - f(\overline{\psi}-\overline{\psi_0})]. \tag{7}$$

The effect of the hyperdiffusion is replaced by implicit high-wavenumber filtering, in which the energy of wavenumbers greater than $2k_{\max}/3$, where $k_{\max}$ is the highest-resolved wavenumber, is tapered down with a minimum tapering factor of 10^{-15} at $k_{\max}$ to avoid energy accumulation at small scales. Thus, the hyperdiffusion is not included in the expression. Equation (7) includes two terms, the first is the usual definition of SGS turbulence in a numerical model, and the second is related to the forcing, which is more analogous to radiation in the atmosphere. It appears here because the forcing in the coarse-grid model is nudging the flow toward a smoothed initial field. The misrepresented shear can affect the growth of small-scale eddies, which later interact with larger eddies. Thus, when the forcing term is active, the parameterization of τ becomes challenging because the actual shear zone is poorly resolved.

In this paper, the DL-based parameterization does not represent the SGS term as an explicit tendency but rather as a correction term composed of the collection of SGS processes; that is,

$$T(\overline{\zeta}'_{t_0+\Delta t}) = \int_{t_0}^{t_0+\Delta t} \tau\, dt, \tag{8}$$

where

$$\overline{\zeta}'_{t_0+\Delta t} = \overline{\zeta}_{t_0} + \int_{t_0}^{t_0+\Delta t} \mathscr{G}(\overline{\psi},\overline{\zeta},t)\, dt \tag{9}$$

is the partial prediction of vorticity at time $t_0 + \Delta t$ by integrating the grid-resolvable dynamics $\mathscr{G}(\overline{\psi},\overline{\zeta},t)$ only. The SGS correction term T is a function of $\overline{\zeta}'_{t_0+\Delta t}$, and the full prediction of vorticity at time $t_0 + \Delta t$ is

$$\overline{\zeta}_{t_0+\Delta t} = \overline{\zeta}'_{t_0+\Delta t} + T. \tag{10}$$

This corrector form is suggested to provide better numerical stability (Um et al. 2020; Kochkov et al. 2021), and this separation of resolved and SGS processes integration is analogous to the idea of partial splitting methods in numerical solvers (Durran 2010).

In addition, we evaluate the performance of the traditional Smagorinsky turbulence parameterization scheme (Smagorinsky 1963). Note that this can only model the first term on the right-hand side of Eq. (7), that is,

$$[\overline{J(\psi,\zeta)} - J(\overline{\psi},\overline{\zeta})] = \nu_e \nabla^2 \overline{\zeta}, \tag{11}$$

where

$$\nu_e = (C_s\Delta)^2|\overline{S}|,$$

$$\overline{S} = \frac{1}{2}\left(\frac{\partial \overline{u}}{\partial y} + \frac{\partial \overline{v}}{\partial x}\right),$$

C_s is a coefficient to be determined by linear regression, and Δ is a measure of grid size.

c. *Model-consistent learning*

We denote the BVE model state at time-step t_k by $\mathbf{x}_{t_k}$ and continue to use the overline to denote the filtering of the LowRes grid mesh. The targeting coarse-grained model state from HighRes simulation in DL training or TRUTH simulation in transfer learning is denoted by $\hat{\mathbf{x}}_{t_k}$. For the training of a DL-based SGS model, we seek a neural network $\mathscr{N}$ to represent T at each time-step t_k such that the corrected physical state,

$$\overline{\mathbf{x}}_{t_k} := \mathscr{N}(\overline{\mathbf{x}}'_{t_k}) + \overline{\mathbf{x}}'_{t_k}, \tag{12}$$

can be used to approximate $\hat{\mathbf{x}}_{t_k}$. Further, still using $\mathscr{G}$ to denote the dynamical core without the SGS process and denoting the full BVE model as $\mathscr{M}$, we have

$$\overline{\mathbf{x}}_{t_{k+1}} = \mathscr{M}\overline{\mathbf{x}}_{t_k} = (\mathscr{N} + \mathscr{T})\circ \mathscr{G}\overline{\mathbf{x}}_{t_k}, \tag{13}$$

where $\mathscr{T}$ is the identity map.

1) Pretraining

The first objective of our study is to investigate whether high-resolution simulations can be used to improve the lead time of operational forecasts. The training data are therefore selected from the HighRes dataset and coarse grained to 64×64 resolution. A sample series in the training set contains an initial condition and n consecutive look-ahead steps of physical states $\hat{\mathbf{x}}_{t_0}, \hat{\mathbf{x}}_{t_0+\Delta t}, \ldots, \hat{\mathbf{x}}_{t_0+n\Delta t}$, where $\Delta t = 0.05$ is the time step for LowRes simulations. For conciseness, we denote the time-step $t_0 + k\Delta t$ as t_k. The neural network $\mathscr{N}$ to be trained is a convolutional neural network (CNN) with 17 layers. The training is carried out by minimizing an accumulative error, which is

$$\mathscr{L} = \sum_{k=1}^{n} L(\mathscr{M}^k \hat{\mathbf{x}}_{t_0}, \hat{\mathbf{x}}_{t_k}), \qquad (14)$$

where L captures the difference in the physical states of the HighRes simulation and the prediction of the ML-enabled model $\mathscr{M}$ at the same time step. When $n = 1$, the training of the neural network falls back to the so-called a priori training because $\mathscr{N}$ is only passed through once and the optimization of $\mathscr{N}$ does not involve the dynamical core $\mathscr{G}$ [cf. Eqs. (12) and (13)].

A natural choice of L is the mean square error (MSE), which is

$$L_{\text{mse}}(\mathscr{M}^k \hat{\mathbf{x}}_{t_0}, \hat{\mathbf{x}}_{t_k}) = (\mathscr{M}^k \hat{\mathbf{x}}_{t_0} - \hat{\mathbf{x}}_{t_k})^{\mathrm{T}} (\mathscr{M}^k \hat{\mathbf{x}}_{t_0} - \hat{\mathbf{x}}_{t_k}). \qquad (15)$$

We also test the use a covariance matrix to normalize the MSE, that is,

$$L_{\text{cov}}(\mathscr{M}^k \hat{\mathbf{x}}_{t_0}, \hat{\mathbf{x}}_{t_k}) = (\mathscr{M}^k \hat{\mathbf{x}}_{t_0} - \hat{\mathbf{x}}_{t_k})^{\mathrm{T}} \mathbf{R}_k^{-1} (\mathscr{M}^k \hat{\mathbf{x}}_{t_0} - \hat{\mathbf{x}}_{t_k}), \qquad (16)$$

where $\mathbf{R}_k$ is the covariance matrix of $\hat{\mathbf{x}}_{t_k}$. Matrix $\mathbf{R}_k$ can be ill conditioned, so we use the ridge regression reconditioning (Tabeart et al. 2020) to improve the conditioning of the covariance matrix before calculating its inverse. The idea of using covariance matrices to normalize cost function originates from the weak constraint 4D-Var data assimilation, which relaxes the perfect model assumption and is intended to update initial condition and model bias errors simultaneously (Laloyaux et al. 2020).

We represent $\mathscr{N}$ with a CNN, which is often used in computer vision research. The input of the CNN is a batch of tensors of $\overline{\zeta}, \overline{\psi}, f(\overline{\psi} - \overline{\psi_0})$ with shape (64, 64, 3). The output of the CNN is the corresponding SGS correction term T with shape (64, 64, 1). The physical variables to be included for evaluating the loss are vorticity and streamfunction. Details of our CNN architecture and hyperparameters can be found in appendix A.

We generate a TRUTH run with 20 000 time units. The training set contains data from 200 to 1000 time units for the pretraining, which equals 80 cycles of the forcing. We pick one snapshot every 2.5 time units and apply a local average filter to get an initial condition for the HighRes simulation at the resolution of 256×256. We run the HighRes simulation for each initial condition for only 5 time units. This ensures the HighRes simulations accurately approximate TRUTH with an average R^2 larger than 0.95 according to the predictability limit test. Then the training set is constructed as the collection of n consecutive snapshots of the HighRes vorticity field that are coarse grained to 64×64. In this experiment, we perform the training for $n = 8, 16, 24, 32$ to see if including more time steps in a training sample series can improve the performance. The case $n = 1$ is also tested to see the performance of the CNN trained without interacting with the dynamical core.

2) Transfer learning

We further conduct experiments to investigate the feasibility of using discontinuous observation data to adjust and improve the pretrained model. The transfer learning is carried out by fine-tuning the whole pretrained CNN with an accumulative loss function similar to Eq. (14). However, here the training reference $\hat{\mathbf{x}}_{t_k}$ is coarse grained from the TRUTH data instead of HighRes. Moreover, the observation data are assumed to be temporally sparse, like observation data in reality. Therefore, t_k should instead be defined as $t_0 + km\Delta t$, where $m\Delta t$ is the time interval between two consecutive observations, and the loss function is

$$\mathscr{L} = \sum_{k=1}^{n} L(\mathscr{M}^{km} \hat{\mathbf{x}}_{t_0}, \hat{\mathbf{x}}_{t_k}). \qquad (17)$$

The input and output variables for transfer learning are the same as that for pretraining. The training set contains data from the TRUTH simulation between $t = 1000$ and $t = 15\,000$, which equals 1400 cycles of forcing. We pick one snapshot every $m\Delta t = 2$ time units from the TRUTH simulation, that is, five observations per cycle, and apply a local average filter to get 64×64 resolution observations. Each training sample series consists of n observations; here we conduct transfer learning for $n = 3$.

3. Results

We select a TRUTH snapshot every 2.5 time units for the period from 15 000 to 19 500 time units, which is 1400 forcing cycles later than the training dataset time, and the resulting collection of snapshots are used as the initial conditions to run a series of LowRes, HighRes, and LowRes with Smagorinsky SGS (LowRes_Smagorinsky) simulations. To evaluate the performance of the DL-based SGS models we obtained, we conducted the online test, which couples the trained neural works with the dynamic core, and the accuracy of predicted vorticity is evaluated.

a. Pretraining

Here, we first report the model's performance during 15 000–19 500 time units, the whole test period. Figure 4 shows the averaged coefficient of determination R^2 and RMSE curves for forecasts using the BVE model with dynamic core coupled with the pretrained SGS models. The Smagorinsky SGS model performs worse in our BVE configuration than LowRes, which does not use any SGS model. CNN1, which includes only one look-ahead step in its training, outperforms

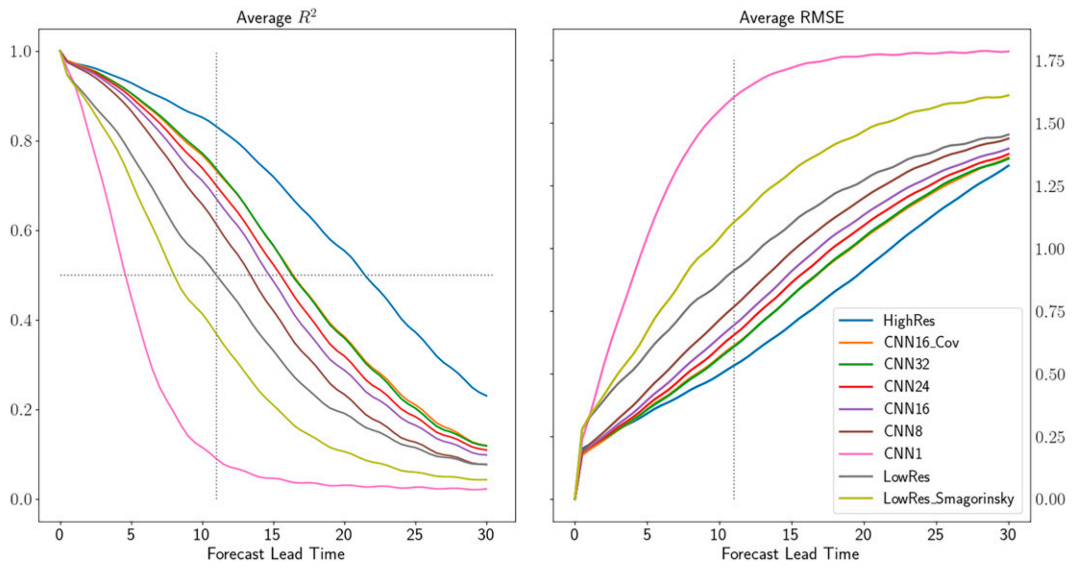

FIG. 4. Averaged (left) R^2 and (right) RMSE curves for online test of pretrained models. LowRes and HighRes are simulations not using any SGS models. LowRes_Smagorinsky uses the Smagorinsky scheme. CNN indicates that the pretrained SGS model is a CNN, the number following CNN is the number of look-ahead time steps during the training, and Cov indicates that the model is trained on the basis of the accumulative covariance-matrix-normalized MSE. Note that curves CNN16_Cov and CNN32 almost overlap. The horizontal dotted line indicates $R^2 = 0.5$, and the vertical dotted lines indicate the forecast time $t = 11$ when the LowRes simulation becomes invalid.

LowRes in the first time unit, and then the error grows and R^2 drops rapidly, giving forecasts significantly worse than LowRes. Somewhat surprisingly, the performance of CNN1 is even significantly worse than the simulation using the Smagorinsky scheme (LowRes_Smagorinsky) after 1 time unit of lead time. This result suggests that the a priori learning approach for our problem is of poor generalizability.

Every other DL model with more than one look-ahead step provides notable improvements in error growth and R^2, which are progressively better as the number of look-ahead steps increases. The different performance associated with the number of look-ahead steps is most discernable before approximately 20 time units of lead time, after which all of those models lose the validity in their predictions. The improvements are further quantitatively reported in Table 1. The effective forecast lead time refers to a forecast period during which $R^2 \geq 0.5$ with respect to TRUTH. The percentage of forecast lead time extended and RMSE is evaluated based on the LowRes simulation's effective forecast lead time, which is 11 time units. The percentage of reducible RMSE that is in fact reduced is the ratio of the RMSE reduced to the difference in the RMSE of LowRes and HighRes simulations when LowRes becomes invalid. It is referred to as "reducible" RMSE because the pretraining dataset comes from HighRes; thus, it is reasonable to assume that the RMSE of HighRes is the lower bound of the RMSE of the simulations using our pretrained SGS model.

We first compare the models trained with different numbers of look-ahead steps of L_{mse}. In Table 1, including 8 look-ahead steps to train a CNN results in a 22.73% extension of effective

TABLE 1. Quantitative evaluation of online tests of models. The RMSE is evaluated when LowRes becomes invalid ($R^2 < 0.5$; approximately at the lead time of 11 time units), and the reducible RMSE is the difference between HighRes RMSE and LowRes RMSE, evaluated at the lead time of 11 time units, i.e., reducible RMSE = $\mathrm{RMSE}_{\mathrm{HighRes}|_{t=11}} - \mathrm{RMSE}_{\mathrm{LowRes}|_{t=11}}$. CNN16_TL is a fine-tuned CNN16. The boldface font in the table indicates the best result among pretrained models.

Name	Effective forecast lead time	Percent of forecast lead-time extended	RMSE	Percent of reducible RMSE reduced
Smagorinsky	8.5	—	1.104 ± 0.047	—
LowRes	11.0	—	0.911 ± 0.034	—
CNN8	13.5	22.73%	0.766 ± 0.032	38.21%
CNN16	15.0	36.37%	0.693 ± 0.038	57.51%
CNN24	16.0	45.45%	0.653 ± 0.034	67.93%
CNN32	**16.5**	**50%**	**0.607 ± 0.036**	**80.15%**
CNN16_Cov	**16.5**	**50%**	0.611 ± 0.036	79.08%
CNN16_TL	**19**	**72.73%**	**0.515 ± 0.028**	**104.49%**
HighRes	21.5	—	0.532 ± 0.020	—

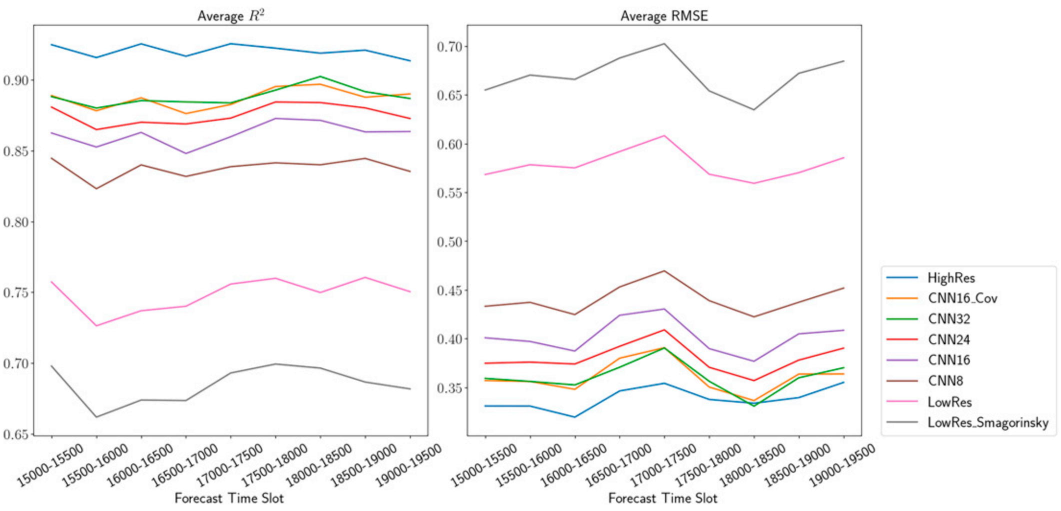

FIG. 5. The (left) R^2 and (right) RMSE time series for online tests of pretrained models for the averaged results from each of the nine successive time segments.

forecast lead time and 38.21% reduction of RMSE relative to LowRes. These improvements can be further increased to 50% and 80.15%, respectively, if the look-ahead step is 32. Considering the test set is 1400 forcing cycles later in time than the training set with only 80 cycles, the CNN models trained with the auto-differentiable dynamic core exhibit substantial generalizability in our BVE case. Also, increasing the number of look-ahead time steps in the training phase can significantly enhance the model's forecasting ability.

CNN16_Cov, which has the same CNN architecture as others but introduces the covariance matrix to normalize the loss function, remarkably outperforms CNN16. Both are trained with 16 look-ahead steps. CNN16_Cov achieves almost the same performance as CNN32 for effective forecast lead time and RMSE. Changing the R^2 threshold will not change the conclusions above; see appendix B for the sensitivity test. It is also worth mentioning that the training of CNN16_Cov converges after 20 epochs while CNN32 converges after around 100 epochs. These improvements are statistically significant, and detailed hypothesis tests are discussed in appendix C.

It is natural to ask whether the improvements brought about by DL models can be preserved when the test set is farther away from the training set in time. For a good parameterization, we hope the improvements can be time invariant. Therefore, we divide the test period into nine successive segments and report the time-averaged R^2 and RMSE series in Fig. 5. The curves do not show an increasing trend in RMSE or a decreasing trend in R^2. The online test performance is not influenced by how far in time it is conducted relative to the training set. This insensitivity indicates that the DL parameterization trained with the auto-differentiable dynamic core is time invariant in our BVE case.

The time-averaged enstrophy spectrum can be used to evaluate the statistical performance of the models and is shown in Fig. 6. The LowRes simulation cannot sufficiently resolve high-wavenumber vorticity and underpredicts enstrophy at medium and large wavenumbers. The enstrophy spectrum of LowRes_Smagorinsky is significantly different from that of other models for both large and small wavenumbers. All DL models trained with the auto-differentiable dynamic core provide a better simulation of the smallest scales of the vorticity field relative to LowRes. However, including more look-ahead steps does not help increase the performance here. CNN32 gives the least-satisfactory simulation among all the DL models for the smallest scales, contrasting with its highest effective forecast lead-time extension and RMSE reduction. CNN8 gives the best small-scale enstrophy distribution. However, it cannot sufficiently represent large-scale structures. CNN16_Cov slightly improves the smallest-scale simulation when compared with CNN16 and CNN24. One possible reason is that the neural networks were optimized only based on accumulated mean square error, without any penalty on the spectral distribution of enstrophy. Another possible reason is the operator used for coarse graining. According to Ross et al. (2023), the online spectral results are related to filtering and the coarse graining choice. So the designing of dataset is also critical.

Overall, the CNN turbulence models trained with more look-ahead steps of L_{mse} exhibit longer extensions of effective forecast lead time and forecast error reduction. These improvements are time invariant, suggesting the generalizability of our SGS model training approach. However, including more look-ahead steps does not help simulate the smallest-scale dynamics better, and the reason behind this is unclear. Normalizing L_{mse} by covariance matrices elevates model performance and shortens model training time. There are also other reasons why CNN16_Cov is preferable to CNN32, which we will discuss in section 3c.

b. Transfer learning

We first tried training a randomly initialized CNN using temporally sparse data, but the training hardly converged. In

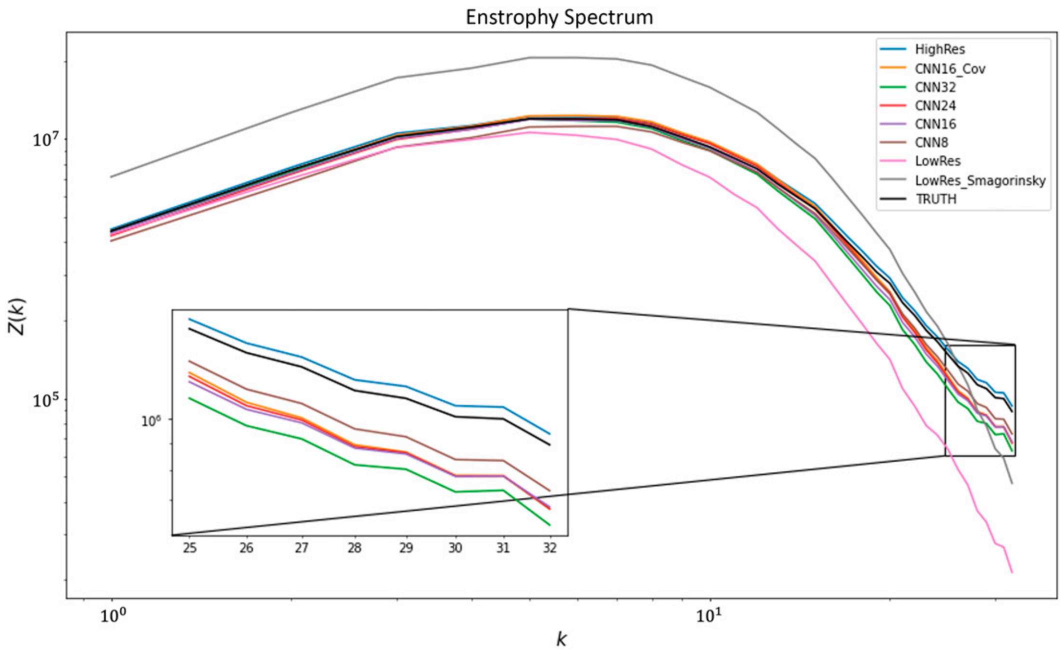

FIG. 6. Enstrophy spectrum averaged over the testing period.

the forward pass of the first training step, the randomly initialized CNN adds noise after every integration step of the BVE model; the error can grow considerably large before the first MSE calculation, which is conducted after 40 steps of dynamical core integration and SGS model "corrections." This makes training from scratch using temporally sparse data hard and underscores the importance of the pretraining using "continuous" high-resolution data.

Figure 7 shows the transfer-learning results using the pretrained CNN models. Using the same neural network structure and taking weights of the pretrained CNN16 as initialization, the fine-tuned CNN16 achieves substantial improvements. For

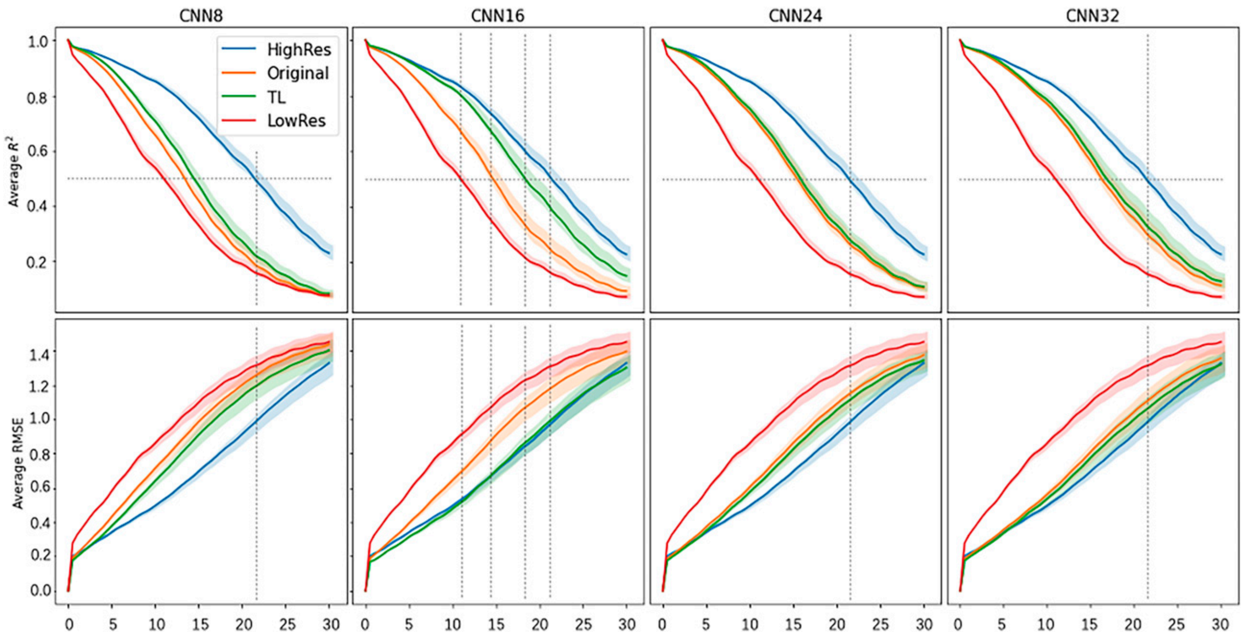

FIG. 7. Averaged (top) R^2 and (bottom) RMSE curves for online tests of model performance after the transfer learnig of the SGS models. Curves for pretrained SGS models are labeled as Original, and TL indicates the results for an SGS model improved through transfer learning.

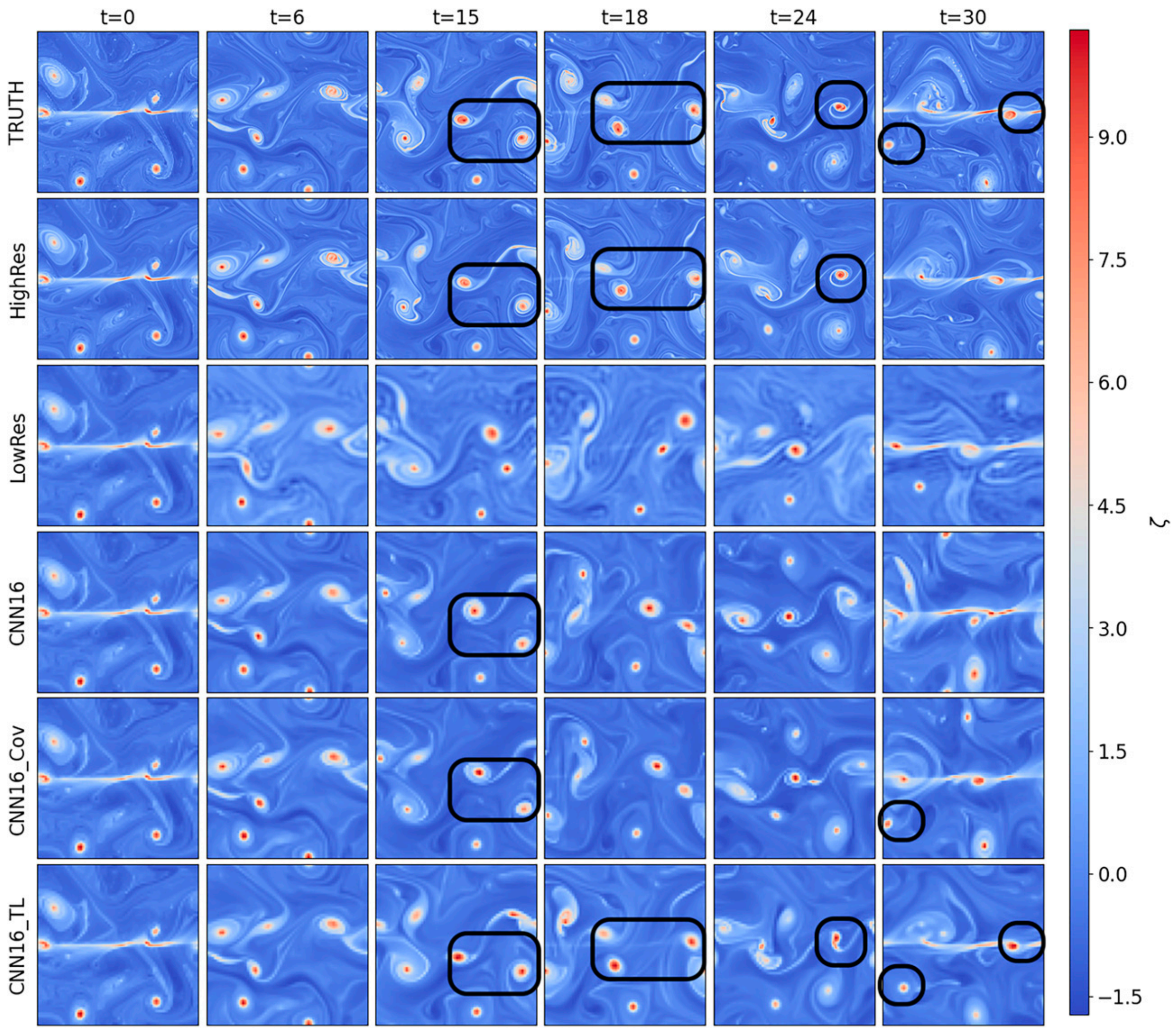

FIG. 8. Vorticity fields of 30-time-unit predictions starting from a TRUTH initial condition at t = 16 525 time units.

forecast RMSE, the fine-tuned CNN16 outperforms HighRes for up to 11 time units with very small standard deviance. The effective forecast lead time is 19 time units for the fine-tuned CNN16, a 72.73% extension relative to LowRes. In comparison with the best-performed pretrained CNN16_Cov, CNN16_TL extends 22.73% of forecast lead time and reduces 25.41% of reducible RMSE, although R^2 is still not as good as HighRes.

Note that the same training strategy and network structure are used; we observe that choosing a pretrained model to fine-tune is critical. However, it still needs to be determined how to make a good choice. A better-pretraining R^2 and RMSE does not guarantee a better fine-tuned network. The performances of CNN8, CNN24, and CNN32 after transfer learning are slightly improved relative to CNN16. The geometry of the loss function used in our transfer learning, which is an accumulated MSE over tens of integration steps, may be very complicated. Therefore, it may not be surprising that they end up staying at different local minimums. As the prediction is sensitive to tiny differences, the online-test results of the fine-tuned networks can be sensitive to the choice of the pretrained initial weight for transfer learning.

Figure 8 illustrates one instance of 30-time-unit predictions of different simulations starting from a TRUTH initial condition at t = 16 525 time units. After 6 time units, each model can forecast the position of vortices well. There are some numerical oscillations at small scales in the LowRes simulation, which can be removed by our DL-based SGS models. After 15 time units, LowRes cannot predict the correct position of the two vortices that should be near the left edge, but the models using the DL-based turbulence models can predict with good accuracy. After 18 time units, CNN16 and CNN16_Cov lost their ability to simulate most of the vortices, which, however, can still be predicted by CNN16_TL. After 30 time units, the CNN16_TL simulation well simulates vortices that cannot be successfully simulated by high-resolution models, as highlighted in Fig. 8. This instance

TABLE 2. The wall time (s) required to train a batch of 8 samples.

Name:	CNN8	CNN16	CNN16_Cov	CNN24	CNN32
Time	0.0815	0.1759	0.1806	0.2678	0.3618

indicates the fine-tuned model has the potential to outperform HighRes simulation in some cases, although, on average, after 30 time units, the forecast produced by CNN16_TL has lower R^2 relative to HighRes.

c. *Computational cost*

Recall that n is defined as the number of look-ahead steps. For pretraining, there are n times of iteration through the neural networks and dynamic core for calculating the loss and its gradient. Therefore, the training can be time-consuming. Table 2 reports the wall time required to train a batch of 8 samples for the same CNN with different look-ahead steps. The test is conducted on a single NVIDIA RTX3090 with 80% memory preallocation. The training time per batch is linear with respect to n. Using a covariance-matrix-normalized loss function increases the training time slightly because of additional matrix multiplication. CNN16_Cov takes one-half of the training time of CNN32 per batch, yet they perform almost identically, and CNN16_Cov gives a better enstrophy spectrum.

Another issue to consider is the additional computation time due to using the DL-based SGS model coupled with the dynamic core when the model is used to forecast. Table 3 shows the time required for different calculation processes in forecasting starting from one initial condition. The inference of the DL model takes 36.38% of the time of the whole simulation. Therefore, the additional cost of using a DL-based SGS model is not negligible, but it is affordable and within a reasonable range considering the complexity of turbulence. As a reference, radiation is usually the most computationally expensive part of an atmospheric model; our experience with the latest version of Cloud Model 1, release 21.0 (CM1; Bryan and Fritsch 2002), is that the one-step computation time of radiation, using the RRTMG scheme (Iacono et al. 2008), is approximately 11 times of the computation time of CM1's dynamical core (mainly advection). Most modelers chose to calculate the radiation only once in multiple model steps to save the overall computational cost. It is also a possible strategy for DL-based SGS models to save computation time in applications.

4. Summary and discussion

In this study, we demonstrated a potential strategy to enhance the prediction skills of NWP models using an idealized BVE case. The strategy starts with pretraining a DL-based SGS process model using high-resolution simulation data, and then we fine-tune the pretrained DL model with transfer learning using observation data. The pretraining is accomplished by incorporating an auto-differentiable dynamic core and exhibits considerable extension of forecast lead time and reduction in RMSE. The DL-based SGS model has significant generalizability as the improvement is time invariant. Using a covariance matrix to normalize the loss function can also improve the performance at a lower computational time cost.

The pretrained SGS model is further fine tuned with the autodifferentiable dynamic core and temporally sparse data for transfer learning. The forecast ability of the models can be further enhanced. When this strategy is applied to an operational forecasting model, we can train a DL model using a relatively large high-resolution dataset once and keep updating it as we accumulate more observation data. Our study shows that this strategy is promising in substantially improving NWP forecast skills. As the observational data are always sparse and noisy, performing transfer learning from these data would be an interesting future topic.

However, there are still some remaining challenges. The first challenge is the representativeness of our idealized study case, which is a two-dimensional flow. Previous studies (Kochkov et al. 2021; Frezat et al. 2022) also apply differentiable physics-based training on two-dimensional flows. A necessary follow-up research direction is to investigate this strategy in three-dimensional flows. In a three-dimensional problem, the computational and dynamical complexity is significantly higher; therefore, the look-ahead steps can be limited, and the choice of DL models also needs to be adjusted accordingly.

Another potential issue is related to the model generalizability implied by our study. The statement that our model is "time invariant" is based on our case with only one temporally periodic forcing. The real atmosphere and climate systems possess forcing variability at different time scales, from diurnal cycles to decadal oscillations, thus requiring more data to encompass all those variabilities in pretraining. However, most likely, we cannot have sufficient samples to represent low-frequency climate variability or global warming–induced changes; therefore, the DL-based SGS models may encounter "out of sample" features that hinder their performance. This issue highlights the importance of updating DL models using transfer learning investigated in our study. A next-generation NWP probably should keep learning from a continuous stream of observation data to adjust its behavior, which involves a relatively new field called lifelong learning or continual learning (Parisi et al. 2019).

We observed some inconsistencies in our study. For pretraining, we saw that RMSE and R^2 improved, but the enstrophy

TABLE 3. The wall time (ms) required for a 10-time-unit (1 cycle) forecast with HighRes dynamical core, LowRes dynamical core, and DL parameterization.

Process	HighRes integration	LowRes integration	Calculate DL input	DL inference	Correction
Time	1138.56	128.02	72.72	120.19	9.48

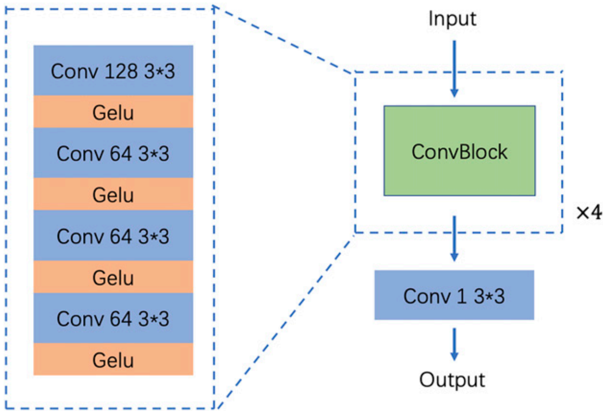

FIG. A1. CNN architecture used in this study, consisting of four convolution blocks, each containing four 2D convolution layers and GELU activations. The number of channels for the four convolution layers is 128, 64, 64, and 64, respectively, and the kernel size is 3×3.

spectrum did not. Also, a pretrained weight with better online-test RMSE and R^2 may not be better performed when working as the initial weight for transfer learning. We believe that one of the causes for these inconsistencies comes from how we optimize and evaluate our model, and the conventional loss functions and metrics may not be sufficient. It would be meaningful to further explore possibilities to design loss functions and metrics to train and identify a deep learning model with better consistency.

The approach to developing the DL-based SGS turbulence model proposed in this study requires the dynamic core of interest to be automatically differentiable; thus, the availability of an auto-differentiable NWP dynamical core is essential. There are now projects that noted the potential of differentiable programming for Earth system modeling, such as the differentiable programming in Julia for Earth system modeling (DJ4Earth) program (Edelman et al. 2021). We believe that the deep learning–based turbulence models combined with an automatically differentiable dynamic core will bring us a significant step forward in advancing modern NWP and climate models, which will become capable of exploiting high-resolution simulations, observations, and deep learning (Schneider et al. 2017).

Acknowledgments. The authors acknowledge the fund support by the Research Grants Council of Hong Kong SAR, China (HKUST-16301721) and the Center for Ocean Research in Hong Kong and Macau (CORE), a joint research center between the Qingdao Pilot National Laboratory for Marine Science and Technology (QNLM) and the Hong Kong University of Science and Technology (HKUST).

Data availability statement. The code for the dynamical core and deep learning supporting the findings of this study is available online (https://github.com/YONGQUAN-QU/BVEX).

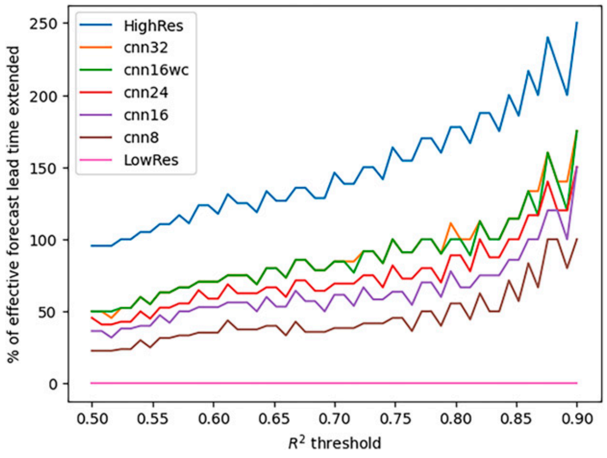

FIG. B1. Changes in percentage of effective forecast lead time for different models with respect to rising R^2 threshold.

APPENDIX A

Neural Network Architectures and Hyperparameters

All of the layers and operations that compose the CNN are shown in Fig. A1. Following the basic ConvNet structure used by Kochkov et al. (2021), we set the kernel size as 3×3. For detailed configuration of each convolutional layer, we set the interwindow stride, input dilation, and kernel dilation as 1. This makes the width and height of the output feature map from each convolutional layer the same as that of the input feature map. Further, we did not include any pooling layer; therefore, the spatial dimensions of the input and output of the CNN are identical. The input of the CNN is a batch of tensors of $\overline{\zeta}, \overline{\psi}, f(\overline{\psi} - \overline{\psi_0})$ with shape (64, 64, 3), and the output of the CNN is the corresponding SGS correction term T with shape (64, 64, 1).

We did not include batch normalization and dropout. The activation function for each convolution layer except for the last one is Gaussian error linear unit (GELU). As compared with the rectified linear unit (ReLU), GELU introduces increased curvature and nonmonotonicity, which potentially makes it easier to approximate complicated functions (Hendrycks and Gimpel 2016).

APPENDIX B

Sensitivity Test

We conducted a sensitivity test to see how the percentage of forecast lead time extended changes with respect to the rising R^2 threshold. The range of the R^2 threshold that we tested is 0.5–0.9. As shown in Fig. B1, changing the threshold does not change the relative order of how well individual models perform.

APPENDIX C

Test of Significance

The distributions of R^2 are not normal; we applied a nonparametric Mann–Whitney U rank test for the distributions

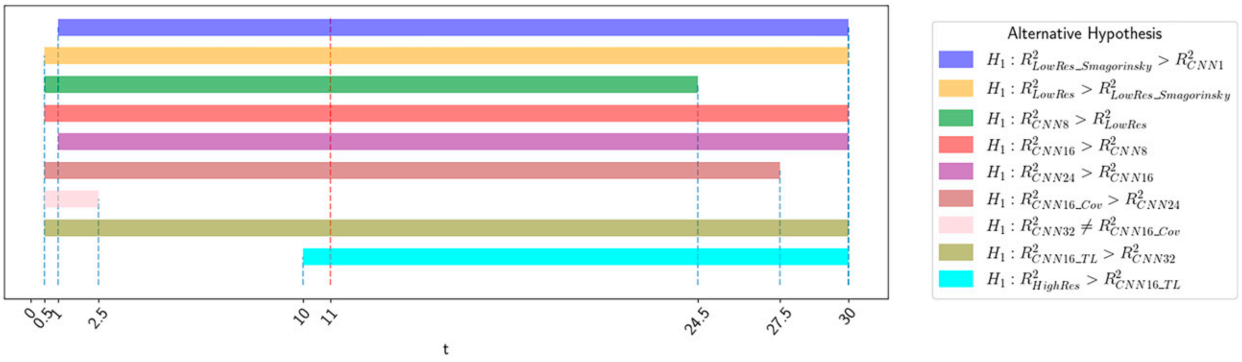

FIG. C1. Forecast lead-time intervals in which the tested alternative hypotheses have p value < 0.05.

of R^2 of different models at given forecast lead time t. We have 200 samples of R^2 for a given model at given forecast lead time t. The null hypothesis to be tested is that two distributions of R^2 are identical. The test results are shown in Fig. C1. The effective lead times for different models that are compared in Table 1 lie in the intersection of the significant time intervals of all the single-sided Mann–Whitney U rank test. So the improvements of R^2, and therefore the effective forecast lead time extensions, brought by online-trained models and the fine-tuned model mentioned in Table 1 are statistically significant with p value < 0.05. There is also a two-sided Mann–Whitney U rank test, which is colored in pink in Fig. C1, that illustrates that most of the time we cannot reject the corresponding null hypothesis that the distributions of R^2 of CNN16_TL and CNN32 are identical. The statistical significance makes us more optimistic to give a positive answer to the question posed by the paper's title.

REFERENCES

Bauer, P., A. Thorpe, and G. Brunet, 2015: The quiet revolution of numerical weather prediction. *Nature*, **525**, 47–55, https://doi.org/10.1038/nature14956.

Baydin, A. G., B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, 2018: Automatic differentiation in machine learning: A survey. *J. Mach. Learn. Res.*, **18**, 1–43.

Beucler, T., M. Pritchard, S. Rasp, J. Ott, P. Baldi, and P. Gentine, 2021: Enforcing analytic constraints in neural networks emulating physical systems. *Phys. Rev. Lett.*, **126**, 098302, https://doi.org/10.1103/PhysRevLett.126.098302.

Bradbury, J., and Coauthors, 2018: JAX: Composable transformations of Python+ NumPy programs, version 0.2.5. Accessed 4 July 2022, http://github.com/google/jax.

Brenowitz, N. D., B. Henn, J. McGibbon, S. K. Clark, A. Kwa, W. A. Perkins, O. Watt-Meyer, and C. S. Bretherton, 2020: Machine learning climate model dynamics: Offline versus online performance. arXiv, 2011.03081v1, https://doi.org/10.48550/arXiv.2011.03081.

Bryan, G. H., and J. M. Fritsch, 2002: A benchmark simulation for moist nonhydrostatic numerical models. *Mon. Wea. Rev.*, **130**, 2917–2928, https://doi.org/10.1175/1520-0493(2002)130<2917:ABSFMN>2.0.CO;2.

Chantry, M., S. Hatfield, P. Dueben, I. Polichtchouk, and T. Palmer, 2021: Machine learning emulation of gravity wave drag in numerical weather forecasting. *J. Adv. Model. Earth Syst.*, **13**, e2021MS002477, https://doi.org/10.1029/2021MS002477.

Chow, F. K., C. Schär, N. Ban, K. A. Lundquist, L. Schlemmer, and X. Shi, 2019: Crossing multiple gray zones in the transition from mesoscale to microscale simulation over complex terrain. *Atmosphere*, **10**, 274, https://doi.org/10.3390/atmos10050274.

Chung, D., and G. Matheou, 2014: Large-eddy simulation of stratified turbulence. Part I: A vortex-based subgrid-scale model. *J. Atmos. Sci.*, **71**, 1863–1879, https://doi.org/10.1175/JAS-D-13-0126.1.

Donahue, A. S., and P. M. Caldwell, 2018: Impact of physics parameterization ordering in a global atmosphere model. *J. Adv. Model. Earth Syst.*, **10**, 481–499, https://doi.org/10.1002/2017MS001067.

Durran, D. R., 2010: *Numerical Methods for Fluid Dynamics: With Applications to Geophysics*. Texts in Applied Mathematics, Vol. 32, Springer Science & Business Media, 516 pp.

Edelman, A., C. Rackauckas, and C. N. Hill, 2021: Earth system model applications. Dj4earth, https://dj4earth.github.io/research/applications/.

Espeholt, L., and Coauthors, 2022: Deep learning for twelve hour precipitation forecasts. *Nat. Commun.*, **13**, 5145, https://doi.org/10.1038/s41467-022-32483-x.

Frezat, H., J. L. Sommer, R. Fablet, G. Balarac, and R. Lguensat, 2022: A posteriori learning for quasi-geostrophic turbulence parametrization. *J. Adv. Model. Earth Syst.*, **14**, e2022MS003124, https://doi.org/10.1029/2022MS003124.

Gentine, P., M. Pritchard, S. Rasp, G. Reinaudi, and G. Yacalis, 2018: Could machine learning break the convection parameterization deadlock? *Geophys. Res. Lett.*, **45**, 5742–5751, https://doi.org/10.1029/2018GL078202.

Gross, M., and Coauthors, 2018: Physics–dynamics coupling in weather, climate, and Earth system models: Challenges and recent progress. *Mon. Wea. Rev.*, **146**, 3505–3544, https://doi.org/10.1175/MWR-D-17-0345.1.

Hendrycks, D., and K. Gimpel, 2016: Gaussian error linear units (GELUs). arXiv, 1606.08415v4, https://doi.org/10.48550/arXiv.1606.08415.

Iacono, M. J., J. S. Delamere, E. J. Mlawer, M. W. Shephard, S. A. Clough, and W. D. Collins, 2008: Radiative forcing by long-lived greenhouse gases: Calculations with the AER radiative transfer models. *J. Geophys. Res.*, **113**, D13103, https://doi.org/10.1029/2008JD009944.

Kassam, A.-K., and L. N. Trefethen, 2005: Fourth-order time-stepping for stiff PDEs. *SIAM J. Sci. Comput.*, **26**, 1214–1233, https://doi.org/10.1137/S1064827502410633.

Kochkov, D., J. A. Smith, A. Alieva, Q. Wang, M. P. Brenner, and S. Hoyer, 2021: Machine learning–accelerated computational fluid dynamics. *Proc. Natl. Acad. Sci. USA*, **118**, e2101784118, https://doi.org/10.1073/pnas.2101784118.

Laloyaux, P., M. Bonavita, M. Chrust, and S. Gürol, 2020: Exploring the potential and limitations of weak-constraint 4D-Var. *Quart. J. Roy. Meteor. Soc.*, **146**, 4067–4082, https://doi.org/10.1002/qj.3891.

Leufen, L. H., and G. Schädler, 2019: Calculating the turbulent fluxes in the atmospheric surface layer with neural networks. *Geosci. Model Dev.*, **12**, 2033–2047, https://doi.org/10.5194/gmd-12-2033-2019.

Parisi, G. I., R. Kemker, J. L. Part, C. Kanan, and S. Wermter, 2019: Continual lifelong learning with neural networks: A review. *Neural Networks*, **113**, 54–71, https://doi.org/10.1016/j.neunet.2019.01.012.

Ross, A. S., Z. Li, P. Perezhogin, C. Fernandez-Granda, and L. Zanna, 2023: Benchmarking of machine learning ocean subgrid parameterizations in an idealized model. *J. Adv. Model. Earth Syst.*, **15**, e2022MS003258, https://doi.org/10.1029/2022MS003258.

Schneider, T., S. Lan, A. Stuart, and J. Teixeira, 2017: Earth system modeling 2.0: A blueprint for models that learn from observations and targeted high-resolution simulations. *Geophys. Res. Lett.*, **44**, 12 396–12 417, https://doi.org/10.1002/2017GL076101.

Smagorinsky, J., 1963: General circulation experiments with the primitive equations: I. The basic experiment. *Mon. Wea. Rev.*, **91**, 99–164, https://doi.org/10.1175/1520-0493(1963)091<0099:GCEWTP>2.3.CO;2.

Tabeart, J. M., S. L. Dance, A. S. Lawless, S. Migliorini, N. K. Nichols, F. Smith, and J. A. Waller, 2020: The impact of using reconditioned correlated observation-error covariance matrices in the Met Office 1D-Var system. *Quart. J. Roy. Meteor. Soc.*, **146**, 1372–1390, https://doi.org/10.1002/qj.3741.

Thompson, D. W. J., and E. A. Barnes, 2014: Periodic variability in the large-scale Southern Hemisphere atmospheric circulation. *Science*, **343**, 641–645, https://doi.org/10.1126/science.1247660.

Um, K., R. Brand, Y. R. Fei, P. Holl, and N. Thuerey, 2020: Solver-in-the-loop: Learning from differentiable physics to interact with iterative PDE-solvers. *Proc. 34th Int. Conf. on Neural Information Proc. Systems*, Vancouver, BC, Canada, Curran Associates Inc., 6111–6122.

Vallis, G. K., 2017: *Atmospheric and Oceanic Fluid Dynamics: Fundamentals and Large-Scale Circulation*. 2nd ed. Cambridge University Press, 946 pp.

Weyn, J. A., D. R. Durran, and R. Caruana, 2019: Can machines learn to predict weather? Using deep learning to predict gridded 500-hPa geopotential height from historical weather data. *J. Adv. Model. Earth Syst.*, **11**, 2680–2693, https://doi.org/10.1029/2019MS001705.

Zhang, C., 2005: Madden–Julian Oscillation. *Rev. Geophys.*, **43**, RG2003, https://doi.org/10.1029/2004RG000158.