Data-driven subgrid-scale turbulence parameterization: methods for improving model efficiency

MACHINE LEARNING
Earth

OPEN ACCESS

RECEIVED
21 November 2025

REVISED
4 April 2026

ACCEPTED FOR PUBLICATION
14 May 2026

PUBLISHED
16 June 2026

PAPER

Data-driven subgrid-scale turbulence parameterization: methods for improving model efficiency

Xingyu Zhu[1], **Yongquan Qu**[2,3], **Naigeng Wu**[4], **Hualong Zhang**[4] and **Xiaoming Shi**[1,5,*]

[1] Division of Environment and Sustainability, Hong Kong University of Science and Technology, Hong Kong Special Administrative Region of China, People's Republic of China
[2] NSF Center for Learning the Earth with Artificial Intelligence and Physics (LEAP), Columbia University, New York, NY, United States of America
[3] Department of Earth and Environmental Engineering, Columbia University, New York, NY, United States of America
[4] Guangdong Meteorological Observatory, Marine Weather Forecast Center of South China Sea, Guangzhou, People's Republic of China
[5] Current address: IAS Center for AI for Scientific Discoveries, Hong Kong University of Science and Technology, Hong Kong Special Administrative Region of China, People's Republic of China.
[*] Author to whom any correspondence should be addressed.

E-mail: shixm@ust.hk

Keywords: hybrid physics-DL model, model efficiency improvement, data-driven SGS models

Supplementary material for this article is available online

Abstract

Data-driven parameterization schemes have gained much attention in atmospheric science in recent years. We designed an idealized testing case with the barotropic vorticity equation model, which employed a subgrid-scale turbulence parameterization based on a deep learning (DL) model. Its effective forecast lead time is substantially improved with the trained DL model parameterization. However, a challenge is the significant amount of computational time required to run the DL model, posing a hurdle for practical applications. Thus, this research aims at improving the DL parameterization's efficiency while minimizing any negative impact on its efficacy. Here we propose three methods, reducing the amount of model parameters, focusing on critical subdomain information only as the input, or combining both. In addition, we introduce new designs to the loss function and improve the baseline model's effective forecast lead time. Through these methods and utilizing the solver-in-the-loop training approach, we reduced the model's computing time while preserving its prediction skills. These acceleration approaches can contribute to the development of more efficient data-driven physical parameterizations and inspire further explorations.

1. Introduction

Appropriately handling sub-grid scale (SGS) processes is essential to numerical weather forecast models. There are various complex physical processes involving significantly different scales that need to be included in a weather forecast model. Those fine-scale processes that cannot be resolved by the numerical models' resolution are called SGS processes. Despite their relatively small sizes and rapid evolution, SGS eddies exert substantial influence on weather forecast accuracy, weather patterns, and even climate dynamics [1, 23, 29]. Traditional parameterization schemes are employed to represent SGS turbulence in numerical weather predictions. However, since SGS turbulence is nonlinear and difficult to describe by simple formulas, those traditional schemes based on various assumptions can add great uncertainties to numerical models and cause unpredictable biases [20, 21].

Data-driven models can serve as a potent approach in such circumstances. One prominent class, which is called 'pure deep learning (DL)' models, learns the full system dynamics directly from data. Employing diverse neural networks (NN) and vast training datasets, DL models can directly extract data features to make predictions, and the forecast results of up-to-date data-driven models, especially for

weather forecasts, can be comparable to those of conventional numerical models, with faster computing speed at the same time. PanGu [3], GraphCast [18], and FengWu [5] are outstanding representatives of data-driven models for medium-range global weather forecasting, showing promising prospects for the utilization of DL.

However, it is difficult to keep pure DL models stable in long-time simulations [41], and instabilities often arise in posteriori simulations [9]. To address these problems, recent work has explored incorporating physical constraints into the NN to improve stability and accuracy [8, 10, 27]. But here we introduce another prominent class, which is called hybrid modeling. Hybrid modelling couples a learnable machine learning component into a physics-based numerical model [28] and is used to extend effective model forecast lead time and enhance the ability to generalize and extrapolate to unseen conditions which are not represented in the training data [34]. This is critical for predicting extreme climate and weather events, which are notoriously difficult to forecast accurately yet have the potential to cause substantial damage. An excellent example of hybrid models is NeuralGCM [16], which demonstrates enhanced stability over lead time, and avoids a clear trend of increasing error when initialized with unseen data, showing the promising values of data-driven parameterizations with differentiable models for SGS processes.

Specifically in terms of SGS turbulence parameterizations, DL models can be applied as a learnable correction term for SGS turbulence to a physics-based dynamical core. For instance, the work of [15] got relatively long-time but still stable simulation results of the Kolmogorov flow by employing such a coupled dynamical core. Their work further employed a solver-in-the-loop (SOL) method [35]. The SOL method allows multiple steps of integration of the dynamical core within each training iteration, which enables the DL model to learn more about the evolving trajectory of physical patterns and thus output more correct SGS corrections for the simulation. Each integration step of the physics dynamical core included in the training iteration is named a look-ahead step, and multiple such steps can be used per iteration with the SOL method. Therefore, by incorporating the fundamental principles of physics into the training process, the trained learnable correction term can serve as a more accurate solution to parameterize SGS turbulence and can further be integrated into the physics model. Following the former work of [15], the work of [26] (QS23 hereafter) obtains similar results by using a convolutional NN (CNN) to train the SGS model for a barotropic vorticity equation (BVE) model with temporally periodic shear forcing, which is more comparable to real atmosphere states compared to the Kolmogorov flow.

However, though existing work using data-driven models has made great efforts to obtain improvements in extending effective forecast lead time, few studies have focused on the efficiency issue, which is important for practical applications. As for parameterization schemes, they should be an integral part of the overall numerical weather forecast model. When such schemes are embedded into these models, they should not substantially increase the computational time. As a result, how to significantly reduce the data-driven SGS model's inference time, which is the time for the model to process data and make predictions, is a meaningful problem to be investigated.

Our work focuses on the solutions of improving the hybrid data-driven model's efficiency while minimizing any negative impact on its accuracy. We use the trained model with eight look-ahead steps from QS23 as a baseline and modify its architecture to lower computational costs. However, improving the DL model's speed without sacrificing its model performance is often challenging. To maintain the hybrid SGS model's effective forecast lead time, we incorporate more physical information into the training loop of the newly refined models by increasing the number of look-ahead steps with the SOL method. Moreover, we also enhance the design of the original loss function and extend the extra effective forecast lead time of the baseline model.

This paper is organized as follows. Section 2 presents the technical details of this paper. In section 2.1, we describe the idealized case setup, including the governing equations and the predictions for the SGS part. Section 2.2 details the data preparation. In section 2.3, the improved design of the loss function is described. Section 2.4 introduces how the model capability is preserved. The newly designed models are introduced in section 2.5, where three different ideas are proposed to achieve our goal of improving SGS model computation efficiency: designing a more compact model, focusing on informative sub-domain data, or both. Section 3 presents the results. Section 3.1 evaluates the impact of the improved loss functions on the model's forecast accuracy, and section 3.2 assesses the computational efficiency gains achieved by the newly designed model architectures. Finally, section 4 presents the summary and discussion for this work.

2. Experiment design and methods

2.1. Governing equations and SGS correction

Same as the case designed by QS23, we use the BVE to describe the evolution of a two-dimensional flow, and activate barotropic instability by adding a shear forcing to the middle of the domain repeatedly at a regular time interval of ten time units.

The equation can be written as:

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

where ψ is the stream function, and ζ is the vorticity, which is defined as: $\zeta = \nabla^2 \psi$. The Jacobian operator $\mathcal{J}(\psi, \zeta)$ is the advection of vorticity, which equals $\psi_x \zeta_y - \psi_y \zeta_x = v\zeta_y + u\zeta_x$. $-\nu\nabla^4\zeta$ is the hyper-diffusion, where ν denotes the hyperviscosity coefficient, and ∇^4 is introduced to dissipate enstrophy at small scales while preserving large-scale dynamics. f is the external forcing term that is used to continuously generate conditions favorable for barotropic instability. It is implemented as a damping term acting on the stream function, which can be written as:

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

where $\mathcal{G}$ represents the grid-scale tendency of the stream function computed from all other physical processes, and f_ψ is the damping term defined as:

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

ψ_0 is the initial stream function field, and α is the damping coefficient prescribed as a function of time t and the meridional coordinate y:

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

The numerical simulation is initialized with an isolated shear layer centered in the middle of the domain [36]. This initial state is constructed by:

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

where ϵ represents a small-amplitude random perturbation uniformly distributed within $\pm 0.05 \times \frac{\pi}{64}$.

Limited by the grid size, some small-scale processes cannot be explicitly resolved, which is referred to as the SGS process. With a coarse grid, the governing equation is written as:

$$\frac{\partial \bar{\zeta}}{\partial t} + \mathcal{J}(\bar{\psi}, \bar{\zeta}) = -\nu \nabla^4 \bar{\zeta} + f_{\bar{\psi}} + \tau, \tag{6}$$

where $\bar{\cdot}$ denotes variables filtered at the coarse grid resolution, and τ represents the SGS tendency which needs to be parameterized by the DL model.

Same with QS23, to maintain the numerical stability, the hybrid SGS model is applied as a correction term for the parameterization, rather than directly predicting the tendency itself. The SGS correction is defined as one-step integration of the SGS tendency:

$$T_\zeta = \int_{t_0}^{t_0+\Delta t} \tau \mathrm{d}t, \tag{7}$$

where T is the SGS correction and can be generated by the DL model:

$$T_\zeta = \mathcal{N}\left(\hat{\bar{\zeta}}_{t_0+\Delta t}, \hat{\bar{\psi}}_{t_0+\Delta t}\right). \tag{8}$$

$\mathcal{N}$ denotes the DL model, and $\hat{(\cdot)}$ represents the coarse-grid solution obtained from the BVE model via direct numerical simulation. The unresolved SGS part is corrected by the DL model through its output T_ζ. The corrected vorticity at time $t_0 + \Delta t$ is then given by:

$$\zeta_{t_0+\triangle t} = \hat{\zeta}_{t_0+\Delta t} + T_\zeta. \tag{9}$$

To sum up, following QS23, we first generate a high-resolution reference dataset by performing direct numerical simulations with the BVE model, which serves as the ground truth ('HighRes Truth'). This high-resolution data is then coarse-grained to a lower resolution, yielding the 'LowRes Truth', which is the true state on the coarse grid and is used to form the training dataset for the DL model. During training, the LowRes Truth is advanced by one time step using the coarse-grid solver, producing a provisional 'LowRes' state that requires SGS correction. This state is fed into the DL model, which outputs an SGS correction term. Adding this correction back to the LowRes state yields the final predicted state, against which the loss is computed to update the model parameters.

Figure 1(a) visualizes the vorticity evolution patterns and demonstrates how the forecast errors grow due to varying SGS processes in our case. The domain is characterized by large eddies for the majority of the time, but at the time when middle shear forcing is enforced (e.g. $t = 845$ and $t = 855$ in figure 1(a)), small-scale eddies are generated and cannot be sufficiently resolved by the relatively coarse grid mesh. Thus the LowRes simulation without a parameterization scheme to represent the SGS processes exhibits obvious forecast biases within a short integration time. Also, the classic Smagorinsky scheme performs even worse in this case (figure 1(b)). However, the CNN-QS8 simulation, which employs the hybrid model trained in QS23, can still get a satisfactory prediction result after the same integration time, showing the effectiveness of hybrid models for SGS parameterizations. More details about the previous work, such as the original design of the CNN model, can be referred to from QS23.

2.2. Data preparation

For the dataset, firstly a 'HighRes Truth' simulation group needs to be generated. A long-duration benchmark simulation of 3200 time units with 256×256 points in a 2π by 2π domain was run with a time step $\Delta t_1 = 0.01$, and the simulated vorticity and stream function fields were saved to serve as the 'HighRes Truth' data group. Then it was coarse-grained into a domain of 64×64 points and archived for every 5 time steps, i.e. $\Delta t_2 = 5 \times \Delta t_1 = 0.05$, which was also called the 'LowRes Truth' data group. This 'LowRes Truth' will serve as the benchmark dataset, and it will also be used as initial conditions for the LowRes simulations on the coarse grid for later experiments, including training and online testing.

The first 50 000 time steps of the overall dataset, equal to 2500 time units or 250 forcing cycles, are chosen for model training (80%) and validation (20%). Then the 54 000th to 64 000th time steps of the benchmark dataset are used for online testing. During online testing, 250 different initial vorticity patterns are selected as the initial conditions for predictions of 50 time units, and the final evaluation is based on the averaged results of all the predictions.

2.3. Loss function design

Apart from the design of the simulation and dataset, the loss function in the optimization problem also plays a key role in DL, especially when it is employed for physics-related tasks. Thus firstly some useful modifications are made to the original loss function. The original loss function from QS23 is written as:

$$\mathcal{L}_{\text{original}} = \frac{1}{N}\sum_{k=1}^{N} \bar{L}\left(\mathcal{M}^k \boldsymbol{x}_{t_0}, \boldsymbol{x}_{t_k}\right), \tag{10}$$

where N is the number of look-ahead steps used for training, and $\boldsymbol{x}$ is the tensor of the general physical state consisting of ζ (vorticity) and ψ (stream function). $\mathcal{M}$ represents the hybrid model (dynamical core and the SGS model, if any), L is the element-wise squared error calculated in each look-ahead step for both the vorticity and stream function loss terms, and $\bar{L}$ represents its average over the entire domain and batch dimension. $\mathcal{L}$ is the final loss function which is the accumulation of $\bar{L}$.

However, significant disparities in magnitude are found for the two different loss terms of the vorticity ζ and the stream function ψ, and no normalization was applied for them in QS23. This imbalance between the two loss terms will then cause the DL model to be dominated by a single physical quantity during training.

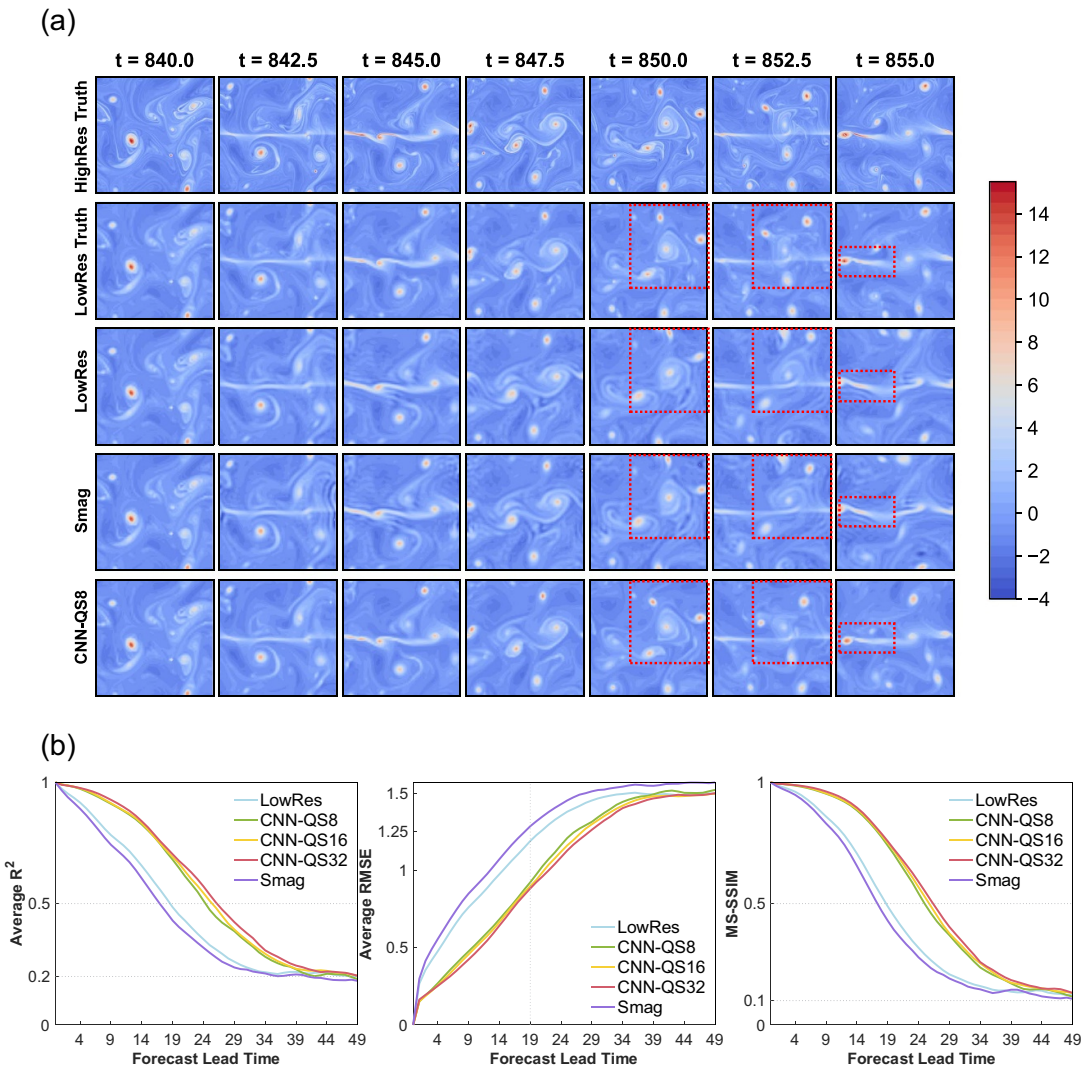

Figure 1. Snapshots of vorticity patterns every 2.5 s from $t = 840$ to $t = 855$ for the truth and the predictions running with a coarser resolution (a), and the averaged R, RMSE and MS-SSIM curves for CNN-QS with different look-ahead time steps, where R is the correlation coefficient, MS-SSIM is the multi-scale structural similarity, and the number following the name of the model in the legend represents the number of the applied look-ahead-steps (b). For (a), the first two rows are the HighRes Truth and the LowRes Truth, respectively, and the last three rows are the predictions running on the low resolution of 64×64 with the same initial condition from the LowRes Truth. The third row is the direct numerical simulation without any parameterization scheme (LowRes), the forth row is the LowRes simulation with the conventional Smagorinsky scheme, and the fifth row is the LowRes simulation with a data-driven parameterization scheme using CNN-QS with 8 look-ahead steps (CNN-QS8). In (b), the horizontal dotted lines mark the R^2 and MS-SSIM thresholds of 0.5, as well as their respective minima near 0.2. The vertical dotted line indicates the maximum effective forecast lead time for the LowRes simulation.

To deal with this, a scaling parameter β_k is applied to the original loss function,

$$\begin{aligned}\beta_k &= \log_{10}\bar{L}\left(\mathcal{M}^k\zeta_{t_0},\zeta_{t_k}\right)-\log_{10}\bar{L}\left(\mathcal{M}^k\psi_{t_0},\psi_{t_k}\right), k=1,2,\ldots,N\\ \mathcal{L}_{\text{scaled}} &= \frac{1}{N}\sum_{k=1}^{N}\left(\bar{L}_{\zeta_k}+10^{\beta_k}\bar{L}_{\psi_k}\right).\end{aligned} \tag{11}$$

The design for the scaled loss function originates from a thought similar to that of the physics-informed NNs (PINNs) [27]. For PINNs, different residuals of the governing equations are assigned weights in the loss function to balance their contributions and enforce physical consistency during training. Here a scaling parameter β_k is applied to split the original L term, which generally represents the physical state, and strengthen the impact of the stream function for the training trajectory because the significant difference in the order of magnitude for $\bar{L}_\zeta$ and $\bar{L}_\psi$ is observed, which can be around 10^4–10^6 in this case. β_k can be calculated and updated for each training epoch, and it can also be a heuristic constant that is pre-tested.

Another interesting thing we find is that though periodic boundary conditions are applied for paddings, sometimes the trained model still exhibits deficiencies in giving precise predictions at boundaries of the domain in the y direction, both in terms of locations and intensity of vortices. Thus another newly designed loss function is further tested based on the scaled one,

$$\begin{aligned} f(y) &= \cos\left(\frac{4\pi}{63}y\right) + 2, 0 \leqslant y \leqslant 63 \\ L_{\cos-\zeta_k} &= f(y) \times L\left(\mathcal{M}^k\zeta_{t_0}, \zeta_{t_k}\right) \\ L_{\cos-\psi_k} &= f(y) \times L\left(\mathcal{M}^k\psi_{t_0}, \psi_{t_k}\right) \\ \beta_k &= \log_{10}\left(\bar{L}_{\cos-\zeta_k}\right) - \log_{10}\left(\bar{L}_{\cos-\psi_k}\right), k = 1, 2, \ldots, N \\ \mathcal{L}_{\cos} &= \frac{1}{N}\sum_{k=1}^{N}\left(\bar{L}_{\cos-\zeta_k} + 10^{\beta_k}\bar{L}_{\cos-\psi_k}\right), \end{aligned} \tag{12}$$

where y is the count of rows in y direction, and the cosine loss function is calculated by adding a cosine-shaped amplification function to the scaled loss in y direction, which aims at driving DL models to give more attention to the evolution of physical states at boundaries and the center.

2.4. Preserve model capability

The primary focus of this study is to reduce the computational complexity of the hybrid model. However, this inevitably comes at the cost of reduced effective forecast lead time. To mitigate this trade-off, we integrated more physics information into the training loop via the SOL method.

We continuously implement the SOL method proposed by [35] with a JAX framework [4] to integrate the PDE solver with DL NNs and do automatic differentiation over them. As introduced earlier, the SOL method incorporates the physics dynamical core directly into the training loop. By using multiple look-ahead steps in training loops, the DL model can learn a longer evolving trajectory of the physical patterns. This enables extending effective forecast lead time for the same model, which provides a pragmatic solution to compensate for the seemingly unavoidable accuracy loss when reducing the amount of model parameters or input data for reducing computation time. As illustrated in figure 1(b), models trained with more look-ahead steps (indicated by the number following the model name, e.g. CNN-QS8, CNN-QS16) achieve longer effective forecast lead times.

Here, the original CNN model proposed in QS23 (model architecture shown in figure 2(a)), which is named CNN-QS in this paper, will be the foundation architecture for future modifications. Eight look-ahead steps are used for the baseline, which is denoted by CNN-QS8. All the refined models will be trained with longer look-ahead steps, for which 32 is manually chosen in this work, and compared to the baseline with respect to the effective forecast lead time and inference time.

2.5. Methods for model efficiency improvement

To reduce the inference time of the original NN, the basic ideas we considered are either to compress the DL model or to compress the physical state data for computing. Based on these principles and inspired by the original model structure of CNN-QS, which has been shown in figure 2(a), three types of methods are proposed and can be classified as (1) using a more compact DL model, (2) applying the DL model to sub-domains of the simulation, and (3) adopting both (1) and (2).

The newly designed model which is more compact than the previous one is named CH**x**Net (figure 2(b)). Separable depthwise convolution with residual blocks is used in this model. Such a model structure is first proposed in Xception [7], and has been employed for many models, such as a refined ResNet-26 [11] and R**x**Net [30]. It has been shown in former experiments that such model structures can effectively make a compact model without obvious losses in the model capability. Thus by using separable depthwise convolution with residual blocks and fewer convolutional layers, the number of model parameters in CH**x**Net is compressed and it is expected that the computational time can be reduced without much loss in the model forecast ability.

The second method is named a split-grid method (SGM, figure 2(c)), which employs the concepts of nested grids common in atmospheric models, such as the Weather Research and Forecast (WRF) model [32]. The WRF model uses multiple nested domains with different resolutions and sub-domain bounds. Similarly, in SGM, the original BVE model domain is split into two parts. One only contains the central half of the original domain where the shear forcing is activated and the initial resolution is retained, and the other covers the whole domain but uses a coarser resolution. This structure allows for less data to be used for computing SGS tendencies. Meanwhile, keeping the resolution for the key area where most fine-scale eddies are generated can help the model retain the most important information, thereby ensuring

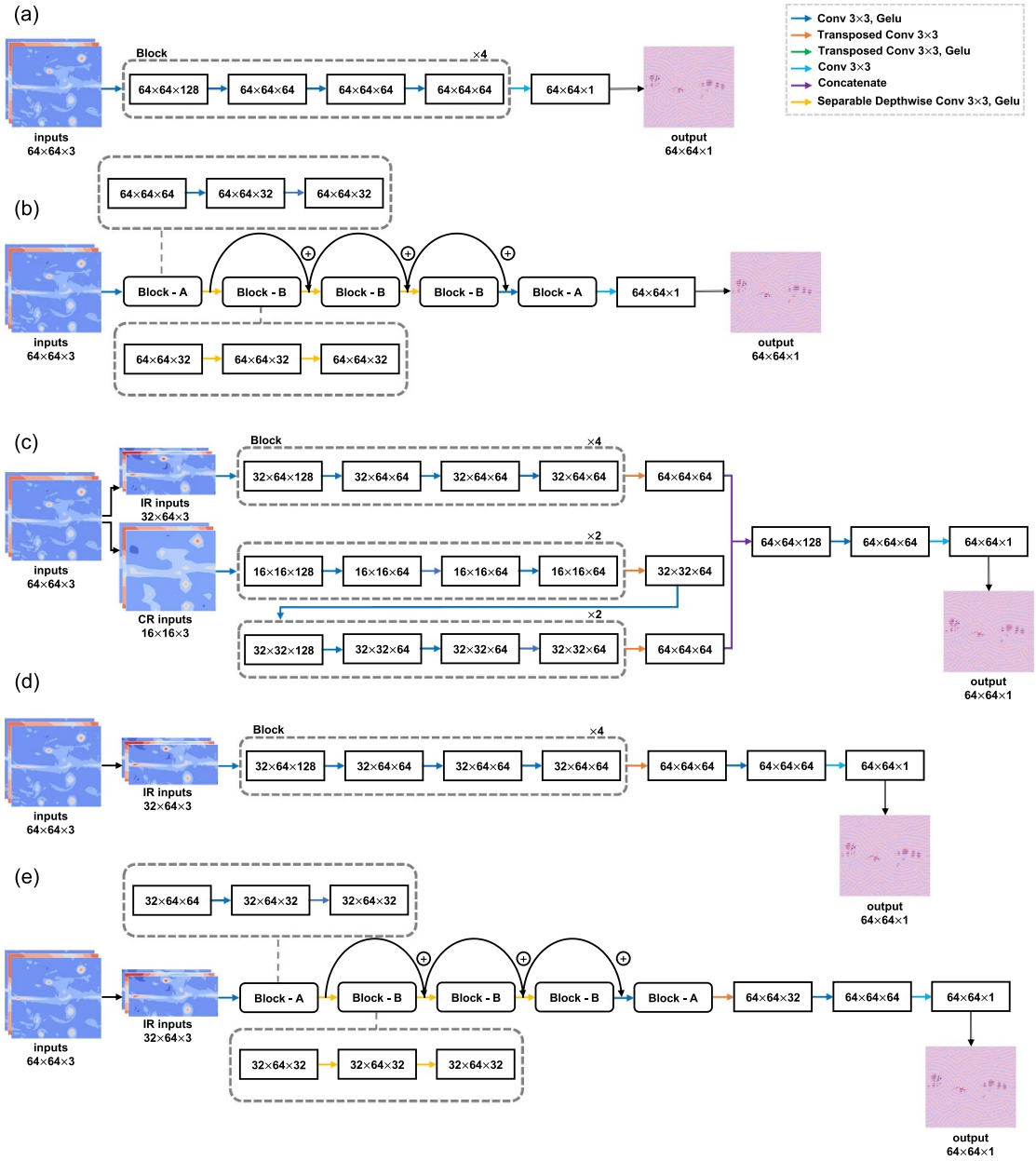

Figure 2. Model structures for the original CNN-QS (a), CH**x**Net (b), SGM for CNN-QS (c), CFS for CNN-QS (d), and CFS for CH**x**Net (e), where $a \times b \times c$ means width×length×channel (i.e. feature maps). IR inputs are inputs with the initial resolution, and CR inputs are inputs with a coarser resolution.

the sustained excellence of forecast quality. It is therefore expected that the DL model with SGM can mitigate excessive information loss when the amount of input data is reduced.

Inspired by SGM, a central-focus scheme (CFS, figure 2(d)) is considered for further reducing the amount of data used for training by applying the DL model to subdomains of the simulation. This scheme only trains the DL model to parameterize half of the full domain and aims to explore if it is practicable to solely train the DL model based on the physical states in the key area to reduce the input data.

To simultaneously compress both the model and the input data, CFS is applied to CH**x**Net (figure 2(e)) as the third method to train a compact model with less physical state data.

The local artificial NNs (ANNs), where multiple and smaller NNs are each assigned to learn the dynamics of a localized region of the entire domain, are known to be an efficient NN structure. This paper also tests a local ANN with its computational complexity similar to that of the initial CNN. However, results show that such a DL model structure is not suitable for this case, though they do run at quite high speeds (figure S1).

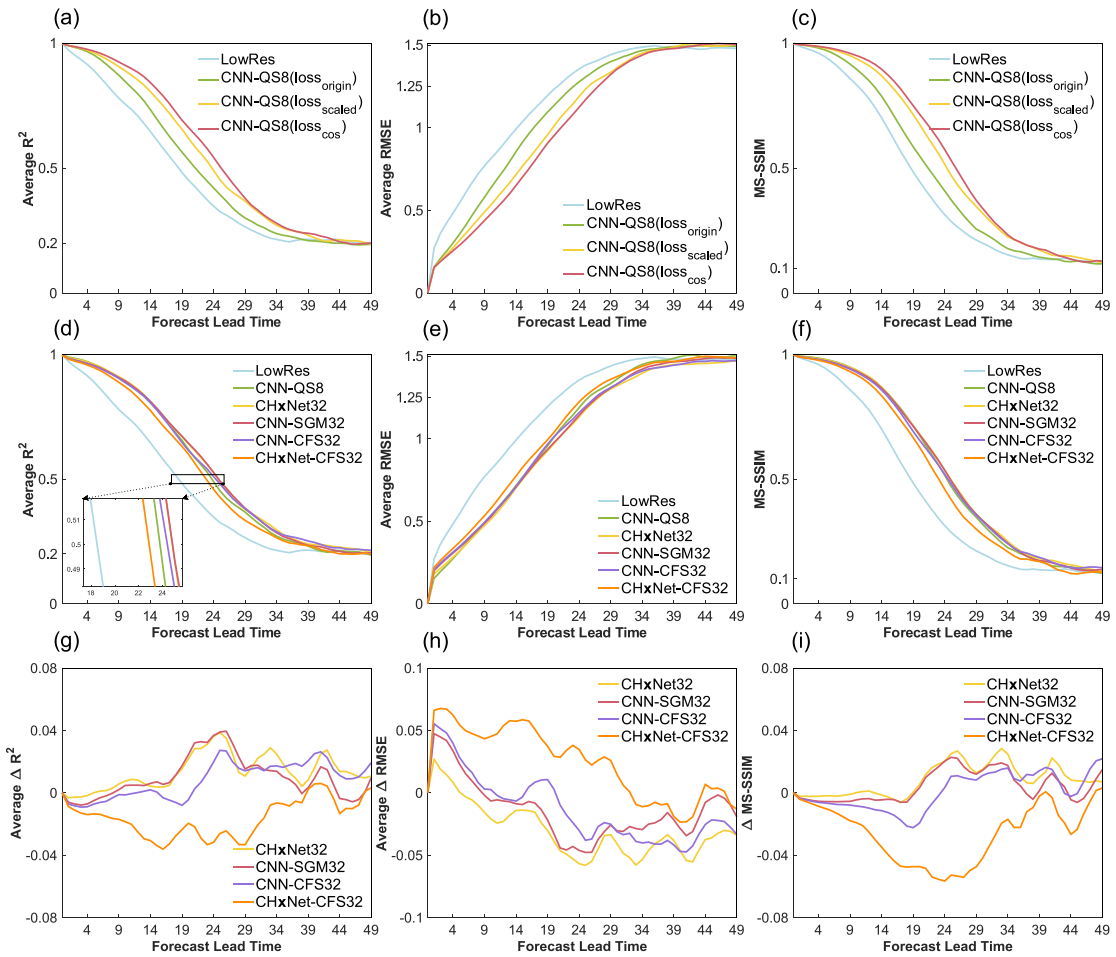

Figure 3. Curves of averaged R^2, RMSE and MS-SSIM over time for the previous CNN model (CNN-QS) with different loss functions (a)–(c) and for the refined models (d)–(f), where R is the correlation coefficient, RMSE is the root mean square error, and MS-SSIM is the multi-scale structural similarity. (g)–(i) show the differences between each refined model and the baseline CNN-QS8 for R^2, RMSE, and MS-SSIM, respectively. The horizontal dotted lines in (a), (c), (d) and (f) mark the R^2 and MS-SSIM thresholds of 0.5, as well as their respective minima near 0.2. In (g)–(i), the horizontal dotted lines indicate the zero baseline relative to CNN-QS8. The vertical dotted lines in (b) and (e) indicate the maximum effective forecast lead time for the LowRes simulation and the baseline CNN-QS8, and those in (g)–(i) only indicate the maximum effective forecast lead time for the baseline CNN-QS8.

3. Results

3.1. Effectiveness of newly-designed loss functions

Figures 3(a)–(c) show the metrics evaluating the accuracy of trained models that use the newly designed loss functions. Those include the squared correlation coefficient (R^2), root-mean-square error, and multi-scale structural similarity [40]. Same with QS23, the threshold of 0.5 of the averaged R^2 is used as the minimum R^2 value of the prediction that we consider as sufficiently accurate.

According to the R^2 threshold, the effective forecast lead time of the LowRes simulation is 19.5 time units, and those of the simulations using the CNN-QS8 turbulence model with the original loss function, scaled loss function, and the cosine-shaped one are 22.1, 24.8, and 26.2 time units respectively. The extension of effective forecast lead time compared with LowRes is 13.6%, 27.1% and 34.6%, respectively. It is evident that the same DL model performs much better with the scaled loss function than the original one without weighting, where the effective forecast lead time extension increases from 13.6% to 27.1%, nearly doubling the relative gain. The valid effective forecast lead time of the model trained with the cosine-shaped loss function can be further extended by another 26% compared with that of the simulation using the scaled loss function. Additionally, the original loss function and the scaled ones are tested with all other DL models (figure S2), and the results all prove the effectiveness of the scaled loss functions.

Table 1. Forecast lead time, and comparisons between different models, where models for testing the forecast lead time are trained and evaluated on a 64×64 domain.

Simulation	Forecast lead time	Forecast lead time improvement (%)	
		Compared to LowRes	Compared to CNN-QS8
LowRes	19.48	N / A	N / A
CNN-QS8	24.76	27.10	N / A
CH**x**Net32	25.80	32.44	4.20
CNN-SGM32	25.81	32.49	4.24
CNN-CFS32	25.38	30.29	2.50
CH**x**Net-CFS32	23.86	22.48	−3.63

Table 2. Computational time consumption, FLOPs, the number of model parameters, and comparisons between different models. FLOPs, model parameters and the computational time consumption are calculated with a 10 000-time-step simulation with one initial condition on the 256×256 domain.

Simulation	Resolution	Total time (s)	DL time (s)	Time saved (%)		FLOPs (M)	Params (M)
				Total	DL		
HighRes	1024× 1024	739.56	N / A	N / A	N / A	N / A	N / A
CNN-QS8	256× 256	259.70	246.95	N / A	N / A	54 252	0.815
CH**x**Net32	256× 256	98.29	85.54	62.15	65.36	5152	0.0786
CNN-SGM32	256× 256	237.01	224.26	8.74	9.19	51 959	1.920
CNN-CFS32	256× 256	174.57	161.82	32.78	34.47	31 046	0.889
CH**x**Net-CFS32	256× 256	74.21	61.46	71.42	75.11	4417	0.107

The cosine loss function is also tested with CH**x**Net32, but it does not show significantly better performance than the scaled one (figure S2). By checking some snapshots of the simulations using CNN-QS8 and CH**x**Net32 with the cosine loss function (figures S3 and S4), we found that when the scaled loss function fails to make DL models give precise predictions at boundaries, adding a cosine-shaped amplification function to the loss helps make the situation better; but if models with the scaled loss function already have minimal errors at boundaries, adding this amplification function makes no change or may even cause larger errors instead. Thus, it is likely that with longer look-ahead steps integrated, CH**x**Net32 can already give more precise predictions at boundaries, resulting in the reduced impact of the cosine-shaped amplification function.

For the scaled loss function, it should be further mentioned that whether updating the scaling parameter β_k at every training step or maintaining a pre-computed one, which is based on the difference in the order of magnitude for various physical quantities at the very beginning, is better depends on the specific DL models. We tested CNN-QS8 and CH**x**Net32 by respectively using a scaling parameter changing every batch and a fixed pre-estimated one with them (figure S5), and the results vary for different models. For CNN-QS8, a changing β_k enhances its performance slightly, but for CH**x**Net32, there is almost no difference. Overall, whether using a changing or fixed scaling parameter gives minimal differences in the models' extended effective forecast lead time.

3.2. Evaluation of newly-designed models

Due to the outstanding and steady performance of the scaled loss function (equation (2)), it is chosen to be applied for all models, and CNN-QS8 is regarded as the baseline for further comparisons. For training, a batch size of eight and the Adam [14] optimizer are chosen. It starts with an initial learning rate of 10^{-3}, and after some epochs, the training will be restarted with a smaller initial learning rate. All the models are evaluated with an NVIDIA A6000 GPU.

Figures 3(d)–(f) show the performance of all the modified DL models, and table 1 lists their extended effective forecast lead time. Both CH**x**Net32 and CNN-SGM32 perform the best among all the modified models, and they are slightly better than CNN-CFS32. CH**x**Net applied with CFS is a little bit worse than the baseline model CNN-QS8, but still in an acceptable range. The differences between each refined model and the baseline are shown in figures 3(g)–(i). Interestingly, most refined models (except CH**x**Net-CFS32) initially perform slightly worse than the baseline but will quickly catch up and maintain better performance over subsequent time steps.

Table 2 shows the number of floating point operations (FLOPs) and model parameters, effective forecast lead time extension, the computational time (walltime), and comparative metrics between the

improved models and the baseline model. FLOPs is a measurement to evaluate a DL model's computational complexity, and the number of model parameters is used for evaluating the DL model size [19]. It should be mentioned that FLOPs only objectively describe the computational complexity of a DL model, but the actual cost of its inference time is influenced by many other factors, such as the memory access cost, memory transfer rate, hardware, parallelization algorithm and so on [22, 31]. Thus the FLOPs values listed in table 2 are only indicative values of the runtime reduction of various models.

Furthermore, the 64×64 grid domain is too small for meaningful computing time evaluation, because a significant portion of the total runtime consists of fixed computational overhead (e.g. from JAX operations) rather than the model inference itself (see table S1). Thus in table 2, the results regarding the computing time evaluation are tested on a larger 256×256 domain.

As shown in table 2, the hybrid physics-DL models all complete the simulation significantly faster than the higher-resolution numerical simulation. Moreover, the newly designed models all get faster inference speed than the baseline CNN model, especially for CH**x**Net, CNN-CFS, and CH**x**Net-CFS, whose improvement is substantial.

CH**x**Net can reduce the number of model parameters from 0.815 M to 0.0786 M and thus reduce the previous FLOPs from 54 252 M to 5,152 M, which brings 62.15% reduction of total walltime for a whole simulation and 65.36% reduction of walltime when considering the DL model's inference only. CNN-CFS with the model structure shown in figure 2(d) has slightly more model parameters (0.889 M) than the baseline (0.815 M). However, because of the reduction of input data, its FLOPs can still be reduced from 54 252 M to 31 046 M, resulting in the reduction of 32.78% and 34.47% for total and DL inference walltime, respectively. The model CH**x**Net-CFS, which applies CFS to CH**x**Net, further reduces both model parameters (from 0.815 M to 0.107 M) and FLOPs (from 54 252 M to 4,417 M) by combining a compact model with reduced input data size. And it saved 71.42% of total time and 75.11% of DL model's inference time.

All three models above succeed in effectively reducing the computational time while keeping a similar level of model performance compared with the baseline DL model. The only exception is CNN-QS with SGM (CNN-SGM), whose model structure is shown in figure 2(c). According to tables 1 and 2, though its model parameters become larger from 0.815 M to 1.92 M, the FLOPs can still be reduced from 54 252 M to 51 959 M, which shows CNN-SGM does reduce the number of operations. However, little improvement of computational time is recorded (only 8.74% and 9.19% reduction of total and DL time). This is mainly because of the great impact of overhead caused by matrix operations at every step to split the original domain, which is the biggest drawback of SGM. And this is where the idea of CFS originates. The resolution of the coarse domain in SGM can be progressively reduced to lower computational cost, and the most extreme extent is to directly abort the coarse part, which is the design principle of CFS.

To sum up, by training with longer look-ahead steps, CH**x**Net, CNN-CFS and CH**x**Net-CFS all work to improve the previous model's efficiency without significant loss of model accuracy. We further test all the models on the 256×256 domain with an NVIDIA H800 GPU (table S3). Reductions in DL inference time are observed, showing the promising application value of DL parameterizations, and all the refined models also perform at a relatively faster computing speed than the baseline model. All these testing experiments reflect the great potential of our new methods, especially for CH**x**Net, CFS and the combination (CH**x**Net-CFS), the last of which takes 70.15% of total model integration (dynamical core + DL model) time on the H800 GPU.

4. Summary and discussion

Conventional numerical models nowadays still need to use a relatively coarse grid mesh to reduce computational costs and enable rapid forecasts. Thus, they struggle to accurately resolve small-scale processes. In recent years, data-driven models have emerged as a promising direction to address this limitation. Models such as MetNet-3 [2] demonstrate that data-driven approaches can achieve higher resolution and faster inference than traditional models. Looking beyond pure prediction, recent work has even demonstrated the discovery of analytical subgrid-scale closures from data with machine learning [13], and the optimization of parameters in conventional models [45].

However, purely data-driven DL models can sometimes be unreliable due to their lack of the correct physical constraints [42]. Thus, hybrid models, which combine physics-based numerical models with DL models, become another development direction and are thought to be useful for enhancing the model performance in comparison with pure DL models [25, 28, 33]. Moreover, though because of numerical integration, the computational cost of a hybrid model increases accordingly, it might enable the use of a

simpler DL model because some physics have already been captured by its numerical model component, thereby having a computational speed comparable to a pure DL weather model.

In this background, we explore strategies to improve the computational efficiency of our hybrid model, which previously used a CNN model for the parameterization of the subgrid-scale process, while avoiding a significant loss of forecast accuracy. We employed an ideal case with the barotropic vorticity model and the solver-in-loop method to train our DL-based SGS parameterizations. By including more look-ahead steps into the training process of our newly-designed models (CH**x**Net, CNN-SGM, CNN-CFS and CH**x**Net-CFS), forecast accuracy similar to the previous model can be obtained while those models are simplified and their computational time can be successfully reduced by up to 75% for DL model inference.

The effective strategies include the use of depthwise separable convolution and residual layers in the DL model and focusing on sub-domain information in the turbulence parameterization. In our simulation case, the middle part of the domain is where the barotropic instability is repeatedly created by the prescribed forcing term. Thus, the subgrid-scale processes in the middle sub-domain are critical for the generation and growth of small-scale eddies. The success of the CFS method suggests that it is a viable approach to reduce the computational cost of DL parameterizations by discarding information in less critical regions of the simulation domain. In realistic weather or climate models, this approach means that the input data source selection should be process-dependent. For example, a deep-learning parameterization of turbulence might be sufficient to take in data in the planetary boundary layer (PBL) only.

Overall, our work provides some promising approaches for improving the computational efficiency of DL-based turbulence models. While the proposed methods are developed and tested only for a shear-forced configuration and may not be directly transferable to other settings, we view them as proofs of concept that illustrate broader design principles, such as focusing computational resources on physically informative sub-domains to reduce potentially extra computational costs, and designing loss functions that emphasize physically important regions to enhance forecast skill. These principles are relevant to both pure DL and hybrid approaches. Many modern DL-based weather and climate models are trained on large datasets and require substantial computational resources. During training, it may be beneficial to focus more on key regions relevant to the target application, which is an approach conceptually similar to CFS. For instance, in PBL parameterizations, turbulence is most active near the surface, and thus training a model primarily on data from that region could probably reduce computational cost without sacrificing too much predictive skill. Similarly, spatially weighted loss functions, such as the cosine-shaped function used in this paper, could be adapted to emphasize region-specific physical processes in other contexts. By demonstrating the potential of these ideas, we hope to inspire further research into efficient and region-aware training strategies for data-driven parameterizations.

Increasing look-ahead steps was a way to maintain the modified DL models' capability in this study. However, it should be noted that using more look-ahead steps is not always effective in improving a model's forecast ability. We also tested an autoencoder (AE) model (figure S6), which is known for its compact architecture and has been widely used for data reconstructions [39]. When we tried to add more look-ahead steps to train the AE model, enhancement was first observed when we increased the number of look-ahead steps from 16 to 32, but deterioration of model performance appeared later when the number of look-ahead steps was increased from 32 to 64. There exists an optimal number of the look-ahead steps for each model, beyond which the DL model may not be able to converge to the best parameterization. Moreover, the best AE model's performance is worse than the baseline model in this study, suggesting that not all the compact models are suitable for achieving performance and efficiency at the same time. Our newly designed models have their unique values.

Lastly, though we have made some progress, similar inconsistencies mentioned in QS23 are still observed. For example, we find that a later checkpoint model sometimes can perform worse than an earlier one, while both training and testing losses kept descending. This might have something to do with the myopic policy of our training strategy. Though several look-ahead steps are included in the training process, that number is still limited, and it might be essentially myopic for the complex task of making a forecast, leading to the model's failure in reducing the long-term impact of error accumulation [6, 12, 24, 43, 44]. Or even if the look-ahead steps included are non-myopic for this task, the training strategy, such as the choice of loss functions, might be unable to allow the trained model to converge to a perfect state that can benefit the model for many steps beyond the included look-ahead steps [17, 37, 38].

Acknowledgments

XZ and XS acknowledge the fund support by the Research Grants Council of Hong Kong SAR, China (AoE/P-601/23-N, HKUST-16301322 and HKUST-16307323). YQ acknowledges funding from NSF through the Learning the Earth with Artificial Intelligence and Physics (LEAP) Science and Technology Center (STC) (Award #2019625). NW and HZ acknowledge the support from the China Meteorological Administration through its Young Innovation Team project (CMA2024QN01).

Data availability statement

Codes for the case setup, dataset generation, and all of our DL models' training and testing can be accessed from [46]. The data that support the findings of this study are openly available at the following URL/DOI: https://doi.org/10.5281/zenodo.11 065 154 [47].

Supporting Information available at https://doi.org/10.1088/3049-4753/ae6dbe/data1.

Funding

The Research Grants Council of Hong Kong SAR, China (AoE/P-601/23-N, HKUST-16 301 322 and HKUST-16 307 323). NSF through the Learning the Earth with Artificial Intelligence and Physics (LEAP) Science and Technology Center (STC) (Award #2019 625). Young Innovation Team project from the China Meteorological Administration (CMA2024QN01).

Author contribution

Xingyu Zhu: Data curation, Software, Methodology, Formal analysis, Investigation, Visualization, Validation, Writing—original draft

Yongquan Qu: Software, Data curation, Resources, Validation, Writing—original draft

Naigeng Wu: Resources, Funding acquisition

Hualong Zhang: Resources, Funding acquisition

Xiaoming Shi: Conceptualization, Funding acquisition, Methodology, Project administration, Supervision, Validation, Writing—original draft

ORCID iDs

Xingyu Zhu 0009-0001-1232-0461
Yongquan Qu 0000-0002-9941-3977
Naigeng Wu 0000-0001-9748-7465
Xiaoming Shi 0000-0002-5329-7851

References

[1] Alapaty K, Herwehe J A, Otte T L, Nolte C G, Bullock O R, Mallard M S, Kain J S and Dudhia J 2012 Introducing subgrid-scale cloud feedbacks to radiation for regional meteorological and climate modeling *Geophys. Res. Lett.* **39** L24809
[2] Andrychowicz M, Espeholt L, Li D, Merchant S, Merose A, Zyda F, Agrawal S and Kalchbrenner N 2023 Deep learning for day forecasts from sparse observations
[3] Kaifeng B, Xie L, Zhang H, Chen X, Xiaotao G and Tian Q 2023 Accurate medium-range global weather forecasting with 3d neural networks *Nature* **619** 533–8
[4] Bradbury J, Frostig R, Hawkins P, Johnson M J, Leary C, Maclaurin D, Necula G, Paszke A, VanderPlas J, Wanderman-Milne S and Zhang Q 2018 JAX: composable transformations of Python+NumPy programs
[5] Chen K *et al* 2025 The operational medium-range deterministic weather forecasting can be extended beyond a 10-day lead time *Commun. Earth Environ.* **6** 518
[6] Cheon M, Byun H and Lee J H 2024 Non-myopic Bayesian optimization using model-free reinforcement learning and its application to optimization in electrochemistry *Comput. Chem. Eng.* **184** 108624
[7] Chollet F 2017 Xception: deep learning with depthwise separable convolutions *2017 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (IEEE Computer Society) pp 1800–7
[8] Cuomo S, Di Cola V S, Giampaolo F, Rozza G, Raissi M and Piccialli F 2022 Scientific machine learning through physics-informed neural networks: where we are and what's next *J. Sci. Comput.* **92**

[9] Guan Y, Chattopadhyay A, Subel A and Hassanzadeh P 2022 Stable a posteriori les of 2d turbulence using convolutional neural networks: backscattering analysis and generalization to higher re via transfer learning *J. Comput. Phys.* **458** 111090

[10] Guan Y, Subel A, Chattopadhyay A and Hassanzadeh P 2023 Learning physics-constrained subgrid-scale closures in the small-data regime for stable and accurate les *Physica D* **443** 133568

[11] Guo Y, Li Y, Feris R, Wang L, and Rosing T. T 2019 Depthwise convolution is all you need for learning multiple visual domains,.

[12] Haghtalab N, Lykouris T, Nietert S and Wei A 2022 Learning in stackelberg games with non-myopic agents

[13] Jakhar K, Guan Y and Hassanzadeh P 2026 Analytical and AI-discovered stable, accurate and generalizable subgrid-scale closure for geophysical turbulence *Phys. Rev. Lett.* **136** 064201

[14] Kingma D P and Jimmy B 2017 Adam: a method for stochastic optimization

[15] Kochkov D, Smith J A, Alieva A, Wang Q, Brenner M P and Hoyer S 2021 Machine learning-accelerated computational fluid dynamics *Proc. Natl Acad. Sci.* **118** e2101784118

[16] Kochkov D *et al* 2024 Neural general circulation models for weather and climate

[17] Krishnapriyan A S, Gholami A, Zhe S, Kirby R M and Mahoney M W 2021 Characterizing possible failure modes in physics-informed neural networks

[18] Lam R *et al* 2023 Learning skillful medium-range global weather forecasting *Science* **382** 1416–21

[19] Hengyi Li, Wang Z, Yue X, Wang W, Tomiyama H and Meng L 2023 An architecture-level analysis on deep learning models for low-impact computations *Artif. Intell. Rev.* **56** 1971–2010

[20] Antonio Mantovani Júnior J, Antonio Aravéquia J, Carneiro R G and Fisch G 2023 Evaluation of PBL parameterization schemes in WRF model predictions during the dry season of the central Amazon Basin *Atmosphere* **14** 850

[21] McFarlane N 2011 Parameterizations: representing key processes in climate models without resolving them *Wiley Interdiscip. Rev.: Clim. Change* **2** 482–97

[22] Molchanov P, Tyree S, Karras T, Aila T, and Kautz J 2017 Pruning convolutional neural networks for resource efficient inference

[23] Müller M and Scherer D 2005 A grid and subgrid-scale radiation parameterization of topographic effects for mesoscale weather forecast models *Mon. Weather Rev.* **133** 1431–42

[24] Kexin Qin L J H and Fan W 2022 Non-myopic knowledge gradient policy for ranking and selection *2022 Winter Simulation Conf. (WSC)* pp 3051–62

[25] Yongquan Q, Bhouri M A and Gentine P 2024 Joint parameter and parameterization inference with uncertainty quantification through differentiable programming

[26] Yongquan Q and Shi X 2023 Can a machine learning-enabled numerical model help extend effective forecast range through consistently trained subgrid-scale models? *Artif. Intell. Earth Syst.* **2** e220050

[27] Raissi M, Perdikaris P and Karniadakis G E 2019 Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations *J. Comput. Phys.* **378** 686–707

[28] Schultz M G, Betancourt C, Gong B, Kleinert F, Langguth M, Leufen L H, Mozaffari A and Stadtler S 2021 Can deep learning beat numerical weather prediction? *Phil.R Soc. A* **379** 2194

[29] Sherwood S C, Bony S and Dufresne J-L 2014 Spread in model climate sensitivity traced to atmospheric convective mixing *Nature* **505** 37–42

[30] Shi X 2020 Enabling smart dynamical downscaling of extreme precipitation events with machine learning *Geophys. Res. Lett.* **47** e2020GL090309

[31] Shuvo M M H, Islam S K, Cheng J and Morshed B I 2023 Efficient acceleration of deep learning inference on resource-constrained edge devices: a review *Proc. IEEE* **111** 42–91

[32] Skamarock W C, Klemp J B, Dudhia J, Gill D O, Barker D, Duda M G, Huang X, Wang W and Powers J G 2008 A description of the advanced research WRF version 3 *Technical Report* (University Corporation for Atmospheric Research)

[33] Slater L J *et al* 2023 Hybrid forecasting: blending climate predictions with AI models *Hydrol. Earth Syst. Sci.* **27** 1865–89

[34] Sun Y Q, Hassanzadeh P, Zand M, Chattopadhyay A, Weare J and Abbot D S 2025 Can AI weather models predict out-of-distribution gray swan tropical cyclones? *Proc. Natl Acad. Sci.* **122** e2420914122

[35] Kiwon U, Brand R, Fei Y, Holl P and Thuerey N 2020 Solver-in-the-loop: learning from differentiable physics to interact with iterative pde-solvers *Advances in Neural Information Processing Systems*

[36] Vallis G K 2017 *Atmospheric and Oceanic Fluid Dynamics: Fundamentals and Large-Scale Circulation* 2nd edn (Cambridge University Press)

[37] Wang C, Li S, He D and Wang L 2022 Is l^2 physics-informed loss always suitable for training physics-informed neural network?

[38] Wang S, Teng Y and Perdikaris P 2021 Understanding and mitigating gradient flow pathologies in physics-informed neural networks *SIAM J. Sci. Comput.* **43** A3055–81

[39] Wang Y, Shi X, Lei L and Fung J C-H 2022 Deep learning augmented data assimilation: reconstructing missing information with convolutional autoencoders *Mon. Weather Rev.* **150** 1977–91

[40] Wang Z, Simoncelli E P and Bovik A C 2003 Multiscale structural similarity for image quality assessment *The 37th Asilomar Conf. on Signals, Systems & Computers, 2003* vol 2 pp 1398–402

[41] 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 *J. Adv. Modeling Earth Syst.* **12** e2020MS002109

[42] Wong C 2024 How ai is improving climate forecasts *Nature* **628** 710–2

[43] Yifan X, Tan Y, Lian Z and Renjie H 2010 A non-myopic approach based on reinforcement learning for multiple moving targets search *The 2010 IEEE Int. Conf. on Information and Automation* pp 1672–7

[44] Yue X and Kontar R A 2022 Why non-myopic Bayesian optimization is promising and how far should we look-ahead? A study via rollout

[45] Zhou Z, Yoon J W, Nguyen-Xuan T, Hur J, Park S K and Eun-Soon I 2026 Coupling a micro-genetic algorithm with RegCM5 for improving extreme precipitation simulations over southeast Asia *Environ. Modelling Softw.* **197** 106871

[46] Zhu X and Shi X 2024 Bvex codes

[47] doi: https://doi.org/10.5281/zenodo.11065154