JAMES | Journal of Advances in Modeling Earth Systems

RESEARCH ARTICLE
10.1029/2023MS003697

# Implementation and Evaluation of a Machine Learned Mesoscale Eddy Parameterization Into a Numerical Ocean Circulation Model

**Cheng Zhang[1], Pavel Perezhogin[2], Cem Gultekin[2], Alistair Adcroft[1], Carlos Fernandez-Granda[2,3], and Laure Zanna[2,3]**

[1]Program in Atmospheric and Oceanic Sciences, Princeton University, Princeton, NJ, USA, [2]Courant Institute of Mathematical Sciences, New York University, New York, NY, USA, [3]Center for Data Science, New York University, New York, NY, USA

**Key Points:**

- A stochastic-deep learning model is implemented in an ocean circulation model, MOM6
- We evaluate the online performance of the stochastic-deep learning model as a subgrid parameterization
- We identify certain limitations of the machine learned parameterization which otherwise has the potential to improve specific metrics

**Supporting Information:**

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

**Correspondence to:**

C. Zhang,
cheng.zhang@princeton.edu

**Citation:**

Zhang, C., Perezhogin, P., Gultekin, C., Adcroft, A., Fernandez-Granda, C., & Zanna, L. (2023). Implementation and evaluation of a machine learned mesoscale eddy parameterization into a numerical Ocean circulation model. *Journal of Advances in Modeling Earth Systems*, *15*, e2023MS003697. https://doi.org/10.1029/2023MS003697

Received 1 MAR 2023
Accepted 25 SEP 2023

**Author Contributions:**

**Conceptualization:** Cheng Zhang, Alistair Adcroft
**Data curation:** Cem Gultekin
**Formal analysis:** Cheng Zhang
**Funding acquisition:** Alistair Adcroft, Carlos Fernandez-Granda, Laure Zanna
**Methodology:** Cheng Zhang, Pavel Perezhogin
**Project Administration:** Alistair Adcroft, Carlos Fernandez-Granda, Laure Zanna
**Resources:** Alistair Adcroft



**Abstract** We address the question of how to use a machine learned (ML) parameterization in a general circulation model (GCM), and assess its performance both computationally and physically. We take one particular ML parameterization (Guillaumin & Zanna, 2021, https://doi.org/10.1002/essoar.10506419.1) and evaluate the online performance in a different model from which it was previously tested. This parameterization is a deep convolutional network that predicts parameters for a stochastic model of subgrid momentum forcing by mesoscale eddies. We treat the parameterization as we would a conventional parameterization once implemented in the numerical model. This includes trying the parameterization in a different flow regime from that in which it was trained, at different spatial resolutions, and with other differences, all to test generalization. We assess whether tuning is possible, which is a common practice in GCM development. We find the parameterization, without modification or special treatment, to be stable and that the action of the parameterization to be diminishing as spatial resolution is refined. We also find some limitations of the machine learning model in implementation: (a) tuning of the outputs from the parameterization at various depths is necessary; (b) the forcing near boundaries is not predicted as well as in the open ocean; (c) the cost of the parameterization is prohibitively high on central processing units. We discuss these limitations, present some solutions to problems, and conclude that this particular ML parameterization does inject energy, and improve backscatter, as intended but it might need further refinement before we can use it in production mode in contemporary climate models.

**Plain Language Summary** This paper discusses how machine learning can be used to make climate models more accurate. Specifically, we import an existing machine learning model that predicts how small eddies (in the order of 10–100 km) in the ocean affect larger currents. We test this machine learning model in a different ocean circulation model than the one it was originally designed for, and found that it worked well. However, we also found some limitations: the model works differently at different depths in the ocean, and it does not work as well near the coasts of the ocean. We also found that the model takes a long time to run on normal computers. Overall, we concluded that the model is promising, but more work is needed to make it work well in realistic situations.

## 1. Introduction

The numerical global circulation models used for climate research solve the governing equations at a finite resolution and are unable to resolve all dynamical scales that influence climate. The spatial resolution of global circulation models has been incrementally refined decade by decade, gradually resolving or admitting new processes. However, the closure problem of parameterizing the influence of unresolved subgrid processes will likely remain for many decades to come. Historically, the development of subgrid parameterizations has required a synergy of theory, observations, and large-eddy simulations, or even direct numerical simulations. Many of these parametrizations have been developed by suggesting a mathematical operator which mimics the bulk effect of the subgrid processes on the large-scale flow properties (e.g., Anstey & Zanna, 2017; Gent et al., 1995; Griffies et al., 1998; Juricke et al., 2017). To construct and then implement parameterizations, in production climate-simulation codes, has required teams of researchers to be funded, for example, the Climate Process Teams (Legg et al., 2009; MacKinnon et al., 2017). Despite tremendous

**Software:** Cheng Zhang, Cem Gultekin
**Supervision:** Cheng Zhang, Alistair Adcroft, Carlos Fernandez-Granda, Laure Zanna
**Validation:** Cheng Zhang
**Visualization:** Cheng Zhang, Pavel Perezhogin
**Writing – original draft:** Cheng Zhang, Pavel Perezhogin, Alistair Adcroft
**Writing – review & editing:** Pavel Perezhogin, Cem Gultekin, Alistair Adcroft, Carlos Fernandez-Granda, Laure Zanna

progress in the development of such parameterizations, they continue to be a source of error in climate simulations (Hewitt et al., 2020) and a source of uncertainty in climate projections (Zanna et al., 2018). Recently, there is growing interest in the use of machine learning to develop parameterizations directly from data, rather than building an ad-hoc mathematical operator for the bulk effect of the subgrid scales onto the large-scale (Beucler et al., 2021b; Bolton & Zanna, 2019; Guillaumin & Zanna, 2021; Krasnopolsky et al., 2010; Maulik et al., 2019; O'Gorman & Dwyer, 2018; Rasp et al., 2018; Ross et al., 2023; Zanna & Bolton, 2020). Many of these show significant skill in offline tests and online evaluation has been demonstrated in several cases, for example, Rasp et al. (2018), Brenowitz and Bretherton (2018), Guillaumin and Zanna (2021) and Yuval et al. (2021). However, few machine learned (ML) parameterizations have been fully implemented in a general circulation model (GCM), nor evaluated for effectiveness in realistic simulations. There are technical and practical hurdles that contribute to the current state of affairs, and we lay out and examine some of those issues in this study.

We set out to implement an ML parameterization in a conventional ocean circulation model, the Modular Ocean Model version 6 (MOM6, Adcroft et al., 2019). This study explores and documents the issues associated with implementing a pre-defined ML parameterization, as well as to further evaluate the ML parameterization beyond the assessment of the ML parameterization authors. The ML parameterization we chose to implement and evaluate is that of Guillaumin and Zanna (2021), hereafter referred to as GZ21. The ML parameterization takes the form of a stochastic-deep learning model and was designed to parameterize the upscale transfer of energy in the inverse cascade of mesoscale turbulence in the ocean. This parameterization is of particular interest for models with eddy-permitting resolution that must account for specific physics inherent to mesoscale eddies. Mesoscale eddies strengthen mean jet currents (Greatbatch et al., 2010) by upgradient momentum fluxes, and result in an inverse kinetic energy (KE) cascade (Balwada et al., 2022; Kjellsson & Zanna, 2017; Scott & Arbic, 2007). GCMs, at typical spatial resolutions, are missing a systematic energy exchange from subgrid to resolved scales. Both properties underline the energizing effect of subgrid mesoscale eddies on the resolved flow. Additionally, mesoscale eddies are responsible for large fraction of heat and salt transport (Delman & Lee, 2021), and thus failing to resolve or parameterize them results in significant biases in mean surface temperature and overturning circulation (Hewitt et al., 2020).

The deep learning model of GZ21, like the majority of other machine learning models, is developed in a high-level programming language Python. MOM6, however, like most of large-scale scientific computation models, is written in a low-level programming language Fortran. One approach to this language barrier is to "port" the code, translating from one language to another (e.g., the work in Sane et al. (2023)). In this case, this would entail rewriting some machine learning libraries in Fortran. While this solution works for some straightforward network architectures it is nevertheless time-consuming and not necessarily extensible. When new and more complex deep learning architectures are invented, more porting would be needed. There are some existing machine learning libraries in Fortran aiming to make this step easier, for example, Neural Fortran (Curcic, 2019) and Fortran-Keras Bridge (FKB, Ott et al., 2020). Such libraries are few in number and typically not up to date as their Python counterparts. One other challenge faced by porting to Fortran is the computational intensity of machine learning methods, which often require dedicated hardware devices for efficient and rapid computation (e.g., a graphics processing unit, graphical processing unit (GPU)), or tensor processing unit. Fortran is not widely used on such devices. An alternative to porting code is coupling the Fortran codes and Python scripts. There are several packages already available that facilitate interoperability between Fortran and Python. Recently, Partee et al. (2022) describe using a turn-key package called SmartSim to implement a parameterization in a large scale ocean model in a high-performance computing (HPC) environment. SmartSim provides a client library that is compiled into the Fortran model, with put/get/run semantics to communicate with a distributed database capable of handling machine learning and data sciences services, and an infrastructure library capable of executing simulation, visualization, and analysis workloads on a variety of HPC platforms. Forpy (https://github.com/ylikx/forpy) is a light-weight alternative to SmartSim. It is a Fortran module that enables the use of Python features in Fortran, and has been utilized for machine learning parameterizations in atmospheric global climate models (Liu et al., 2021), atmospheric gas-phase chemistry models (Espinosa et al., 2022), and computational fluid dynamics turbulence models (Beck & Kurz, 2021). As a Fortran module Forpy can be compiled on any computer or clusters that have Python and Fortran installed, and all locally available Python libraries are then accessible from the Fortran application. The demonstrated simplicity, compatibility, and versatility of this light-weight package led us to use Forpy for this study.

The paper is organized as follows. In Section 2, the ocean model MOM6, the stochastic-deep learning model and their bridge are introduced. To demonstrate the online performance of the system, Section 3 presents an idealized case: a wind-driven double gyre. In Section 4, some potential issues when applying the deep learning model to a numerical ocean model are highlighted, along with some potential solutions. Conclusions and ideas for future study are discussed in Section 5.

## 2. Methods

In this section, we describe the framework consisting of the numerical ocean model MOM6 and the stochastic-deep learning model for predicting mesoscale ocean dynamics. The numerical ocean model MOM6 that we use for the ML-based parameterizations is described in Section 2.1. In Section 2.2, we recapitulate the methodology of the Convolutional Neural Network (CNN) model in GZ21 and describe the inference stage in MOM6. Finally, in Section 2.2 we provide the workflow and techniques of online implementation.

### 2.1. Ocean Model Description

The numerical model employed in this study is the Modular Ocean Model version 6 (MOM6, Adcroft et al., 2019), a solver for ocean circulation written in Fortran used for ocean climate simulations. We use the model in an adiabatic limit with no buoyancy forcing which simplifies the equations of motion to the stacked shallow water equations. The layer momentum equations given in vector-invariant form are

$$\frac{\partial \mathbf{u}_k}{\partial t} + \frac{f+\zeta_k}{h_k}\hat{\mathbf{z}}\times h_k\mathbf{u}_k + \nabla K_k + \nabla M_k = \mathbf{F}_k \tag{1}$$

where $\mathbf{u}_k$ is the horizontal component of velocity, $h_k$ is the layer thickness, $f$ is Coriolis parameter, $\zeta_k$ is the vertical component of the relative vorticity, $K_k = (1/2)\mathbf{u}_k \cdot \mathbf{u}_k$ is the KE per unit mass in the horizontal, $M_k$ is the Montgomery potential (defined in Appendix A) and $\mathbf{F}_k$ represents the accelerations due to the divergence of stresses including the lateral parameterizations that are not inferred from ML-based models, $\hat{\mathbf{z}}$ is the unit vector pointing in the upward vertical direction, $k$ is the vertical layer index with $k = 1$ at the top and $\nabla$ is the horizontal gradient. The governing equations are discretized on C-type staggered rectangular grid with finite volume method, and the advection operator is energy-conserving (in our setup). We take the limit of an adiabatic fluid with a single constituent so that the governing equations simplify to the stacked shallow water equations. In the adiabatic limit used here, vertical advection of all quantities is represented by the Lagrangian motion of the model layers. Appendix A contains a full description of governing equations.

The oceanic mesoscale turbulence that interests us involves an upscale cascade of energy from small (unresolved) scales, so a finite resolution model needs a subgrid momentum forcing to account for nonlinear interactions with the unresolved eddies. This subgrid momentum forcing can be diagnosed by

$$\mathbf{S}_k = \begin{pmatrix} S_{kx} \\ S_{ky} \end{pmatrix} = (\bar{\mathbf{u}}_k \cdot \nabla)\bar{\mathbf{u}}_k - \overline{(\mathbf{u}_k \cdot \nabla)\mathbf{u}_k} \tag{2}$$

where the overbar is the horizontal filtering and coarse graining, and we make use of the identity $\mathbf{u}_k \cdot \nabla \mathbf{u}_k = \zeta_k \hat{\mathbf{z}} \times \mathbf{u}_k + \nabla K_k$. In Equation 2 we consider nonlinear interactions only due to momentum advection operator, and neglect subgrid forcing from nonlinearity in vertical transport, varying Coriolis parameter and subgrid dissipation. The coarse resolution flow evolves according to the equation

$$\frac{\partial \bar{\mathbf{u}}_k}{\partial t} + \frac{f+\bar{\zeta}_k}{\bar{h}_k}\hat{\mathbf{z}}\times \bar{h}_k\bar{\mathbf{u}}_k + \nabla \bar{K}_k + \nabla \bar{M}_k = \bar{\mathbf{F}}_k + \mathbf{S}_k \tag{3}$$

and the parameterization of subgrid forcing $\mathbf{S}_k$ is function of $\bar{\mathbf{u}}_k$. GZ21 developed a deep learning model to predict the statistical moments of $\mathbf{S}_k$ that can be used in a stochastic parameterization in the coarse resolution model.

### 2.2. Stochastic-Deep Learning Model

The stochastic-deep learning model of GZ21 is a Fully CNN with eight convolutional layers, where the kernel size of the first two layers is $5 \times 5$ and the kernel size of the rest layers is $3 \times 3$. Each of the convolutional layers has 128, 64, 32, 32, 32, 32, 32, and 4 filters, respectively. The ReLU activation function is used for hidden layers and no padding is used in the convolutional layers. The CNN architecture results in the stencil size of $21 \times 21$ for predicting the forcing on a single grid point. In contrast to a deterministic parameterization for predicting the momentum forcing, the CNN models the mean and standard deviation of a Gaussian probability distribution of the subgrid momentum forcing. The mean square error (MSE) loss function is replaced by a full negative Gaussian log-likelihood of the forcing. The CNN was trained and validated with surface velocity data from the high-resolution coupled climate model CM2.6 (Griffies et al., 2015) which has a nominal resolution in the ocean model of 1/10°. This resolution is considered sufficiently fine to resolve eddies in the tropics and mid-latitudes of the global ocean (Hallberg, 2013). The simulated ocean surface velocity fields from four subdomains are selected as representative of different dynamical regimes. More details about the model, training, and data can be found in Section 2 of GZ21.

The parameterization is evaluated at each time step in the ocean model using the velocity components as the inputs to the CNN model which returns the mean and standard deviation of a Gaussian probability distribution of the subgrid momentum forcing. The stochastic subgrid momentum forcing is then generated by

$$S_{C,i,j} = S^{(mean)}_{C,i,j} + \epsilon_{C,i,j} \cdot S^{(std)}_{C,i,j}; \quad C = x, y;\ i = 1, \ldots, m;\ j = 1, \ldots, n \tag{4}$$

where $i$ and $j$ are the ocean model spatial indices, $C$ indicates the component of momentum forcing (zonal "$x$" or meridional "$y$"), and $\epsilon_{C,i,j}$ are random 2D fields sampled from the standard normal distribution, independent for each grid cell, zonal/meridional component, vertical layer, and time step.

### 2.3. Online Implementation of CNN With MOM6

The MOM6 ocean circulation model is exclusively written in Fortran, while the stochastic-deep learning model was developed with the machine learning package PyTorch (many deep learning practitioners favor developing machine learning models in Python, and other recent languages, since machine learning tools are readily available). Computer language interoperability is a technical barrier that we overcome here by using the package Forpy. Python is an interpreted language, while Fortran is compiled. A system call from Fortran to run a Python script would require booting the Python interpreter each time the Python functions are needed. Most approaches to embed Python in a compiled language therefore use the C-language API to call the Python run-time library directly. This embedding method requires writing an intermediate software layer for all the possible combinations of arguments to functions and so is not readily extensible. Forpy is a Fortran module that provides that interface to the Python library, and appears to avoid any significant overheads. The module conveniently allows data to be passed from the calling Fortran code to functions in the Python script. In addition, Forpy allows us to use any Python libraries from Fortran, is independent of the computing environment, and does not require installing any other software that needs system privileges. Another benefit of using the Python language for inference in MOM6 is that the network can utilize the graphical processing units (GPUs) even though MOM6 exclusively executes on central processing units (CPUs).

In the hybrid model consisting of MOM6 and the CNN parameterization, the velocity field is computed by MOM6 first using all available terms in Equation 3. The Fortran array of the velocity is then wrapped up as a Numpy array by Forpy and transferred to Python as the input of the CNN model. The CNN returns the moments in Equation 4 and then random numbers are generated to yield the momentum forcing in a Numpy array. The momentum forcing is then transferred back to Fortran and Forpy provides an interface to read the data from the Numpy array in Fortran. The momentum is then updated with this stochastic forcing and the hybrid model continues as would the conventional MOM6. Figure 1 illustrates the flowchart of the whole hybrid model.

Not only does the language barrier complicate the implementation of a CNN into an ocean model, but it also complicates how computations are distributed among computing resources. The MOM6 ocean model utilizes data parallelism, where the computational domain is divided into subdomains with overlapping halo regions which are kept in-sync as needed by communication between adjacent processors using the message passing interface (MPI) communications libraries. In the conventional MOM6 model, the width of the halo region is

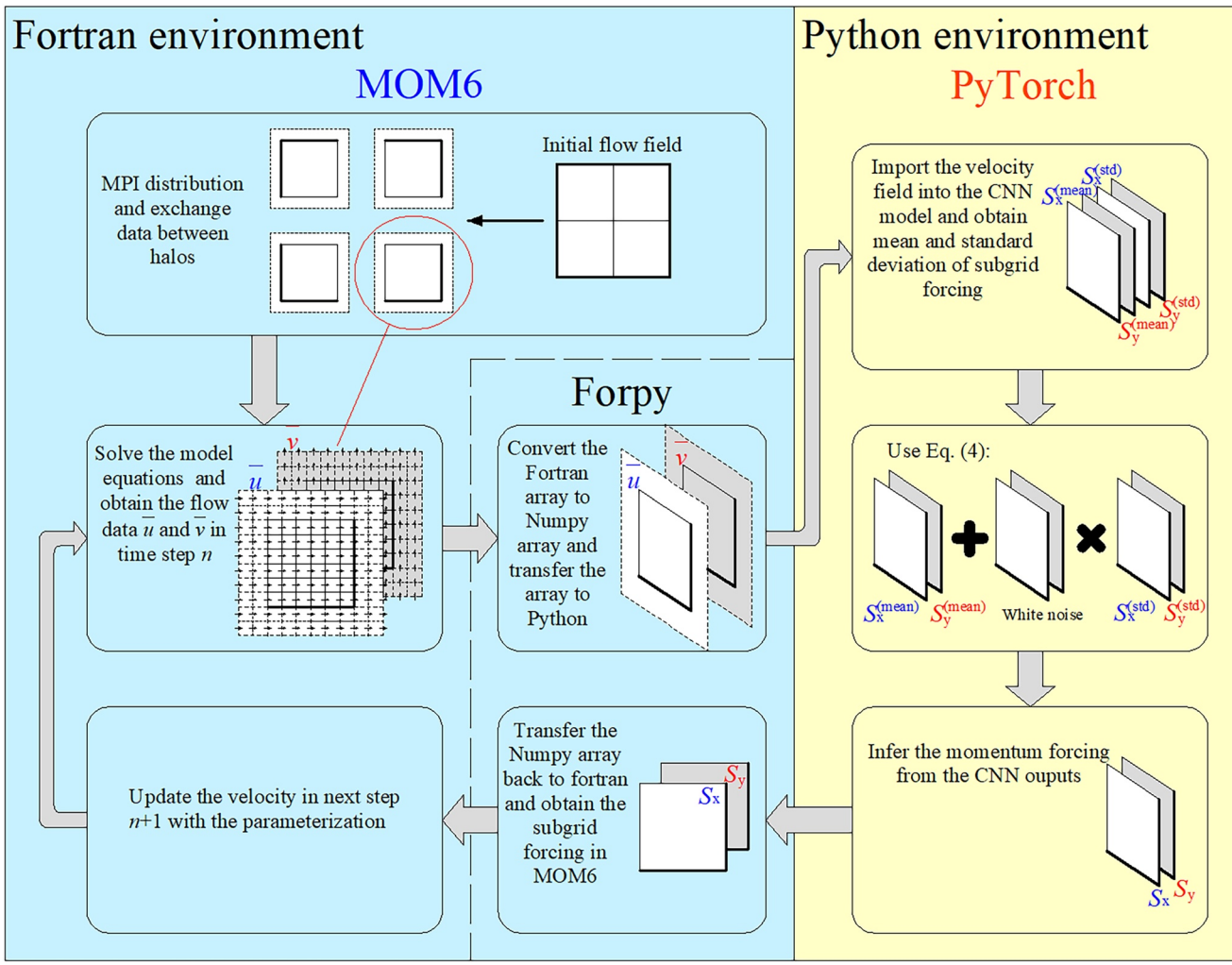


**Figure 1.** Flowchart of the coupling system between MOM6 and the Convolutional Neural Network model, using Forpy.

determined by the stencil of numerical discretization and is typically on the order of three or four cells. A computation involving spatial stencils generally needs to be preceded or followed by a halo synchronization (MPI exchange). Optimal scaling of MOM6 is obtained when the costs of communication, additional memory, and extra computation, associated with the halos are minimized. On contemporary platforms this typically leads to using the number of cores such that the width of halo is less than a quarter of the sub-domain width/height belonging to each core. The CNN has a stencil of 21 × 21 cells which is far wider than any discretized terms in MOM6 and which requires expanding the width of the halos to 10, and sometimes violating the less-than-quarter rule. We discuss this further in Section 4.5.

For the treatment of land, wherever the CNN parameterization would return momentum forcing on land (dry points), the velocity and forcing are set to zero.

## 3. Online Performance: Wind-Driven Double Gyre

GZ21 evaluated the CNN parameterization in a barotropic model and showed good online performance. Here, we test the online performance of the parameterizations in a baroclinic model, applying the closure in the ocean interior for which it was not trained. In this paper, we focus on different metrics from those used in the network training, and evaluate the parameterization from the perspective of the large scale model solution and not the details of the processes being parameterized. We examine the effect of spatial resolution and tuning, in which the parameterization is attenuated or amplified. We also make qualitative comparisons between parameterized coarse

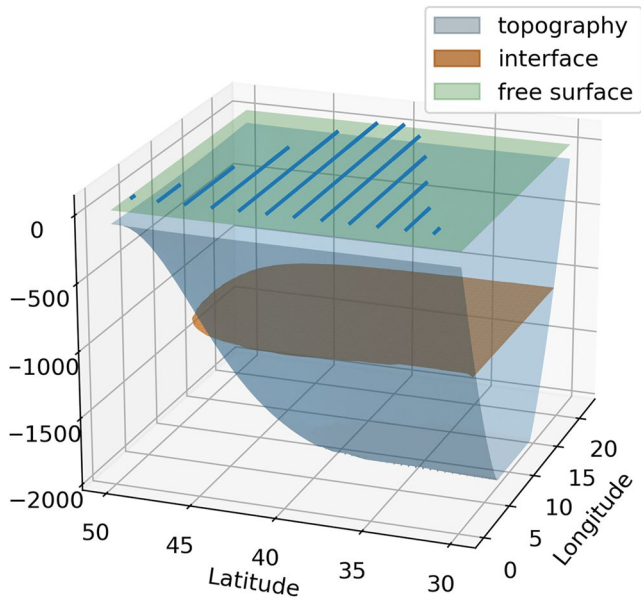


**Figure 2.** Sketch of the wind-driven double gyre configuration: free surface (green), layer interface (brown) and bowl-shaped topography (blue/gray). Blue lines indicate the strength and direction of imposed fixed zonal wind-stress.

grid results and fine grid results. It should be noted that in this work, the term "online" refers to the process of inferring from a trained deep learning model rather than the process of continuously updating a deep learning model as simulations progress, which is referred to as "online learning." For the offline evaluation of the CNN model performance on the double-gyre case, please refer to Appendix B.

### 3.1. Case Setup

The ocean model is configured to simulate a wind-driven double gyre in a bowl-shaped basin (Hallberg & Rhines, 2000) and a vertical wall at the southern boundary (Figure 2). The coordinate system is spherical, with computational domain ranging from 0 to 22° in longitude and from 30 to 50° in latitude. Coriolis parameter is given by $f = 2\,\Omega\sin(\phi)$, where $\Omega = 7.2921 \cdot 10^{-5}$ s$^{-1}$ is planetary rotation rate and latitude $\phi$. Although we use a primitive equation model, in this configuration the governing equations are simplified to a two-layer shallow water model without thermodynamics (no computations involving equation of state, temperature and salinity). The maximum depth is 2000 m and an interface between layers is initially located at the depth of 1000 m (at rest). Let $h_1$ and $h_2$ be the upper and lower fluid layer thickness, respectively. The density of the upper layer is $\rho_1 = 1035$ kg/m$^3$, and the density of lower layer is $\rho_2 = 1{,}036.035$ kg/m$^3$, and corresponding reduced gravity for the interior interface is $g' = g(\rho_2 - \rho_1)/\rho_1 = 0.0098$ m/s$^2$, where $g = 9.8$ m/s$^2$. The Rossby deformation radius $R_d = c/f$ decreases from 30 km in the south to 15 km in the north (approximate), where $c = \sqrt{g' \frac{h_1 h_2}{h_1 + h_2}}$ is the gravity wave speed of the baroclinic mode (Gill & Adrian, 1982). The flow is forced by a constant (in time) surface wind stress $\tau_x$ and varies latitudinally with a maximum at center latitude ($\phi = 40$) and zero stress at borders ($\phi = 30, 50$):

$$\tau_x = \tau_0 \left[1 - \cos\left(2\pi \cdot \frac{\phi - 30}{20}\right)\right] \geq 0 \tag{5}$$

with $\tau_0 = 0.1$ N/m$^2$. The simulations last 10 years and are initialized from rest. We demonstrate that structures seen in any 5 year average, after the first 5 years, are reflected in a 100 years average in Text S4 in Supporting Information S1 The circulation and mesoscale turbulence reaches statistical equilibrium after about 5 years. The full specification of parameters is given in Zenodo (Zhang, 2023a). For the turbulence model we use a biharmonic viscosity with a Smagorinsky eddy viscosity following Griffies and Hallberg (2000), where the details are in Appendix A. Scale selective friction is required to remove small-scale numerical noise and stabilize the computations and is applied in both reference and parameterized simulations. The Smagorinsky constant in all experiments here is $C_S = 0.06$. We vary the spatial grid size and time step (see Table 1) in these experiments. The lateral boundary condition at the vertical wall is implicitly free-slip due to the vanishing of $h$ causing the layer-integrated stress to vanish.

**Table 1**
*Summary of the Spatial and Temporal Resolution of the Reference Simulations Used*

| Experiment | R4 | R32 |
|---|---|---|
| Grid spacing, degree | 1/4 | 1/32 |
| Grid spacing, km (approx.) | 24 | 3 |
| Time step, min | 18 | 2.25 |
| Cell count (Lon. × Lat.) | 88 × 80 | 704 × 640 |

Most evaluations we present will be in a model with 1/4° horizontal resolution, hereafter referred to as R4. R4 is "eddy permitting" in that it exhibits mesoscale variability that contributes to variability of the separating boundary current.

For the purposes of evaluating the CNN parameterization in R4, a 1/32° model is run to obtain a "truth" run (hereafter referred to as R32). R32 is fine enough to resolve some of the mesoscale cascade. Note that R32 is also finer than the training data from the global model used to construct the CNN parameterization.

Figure 3 shows the snapshots of the upper layer relative vorticity (normalized by the planetary vorticity) (a–c) and KE (KE, d–f), at the end of the run, and the 5-year averaged sea surface height (SSH, g–i), for coarse resolution

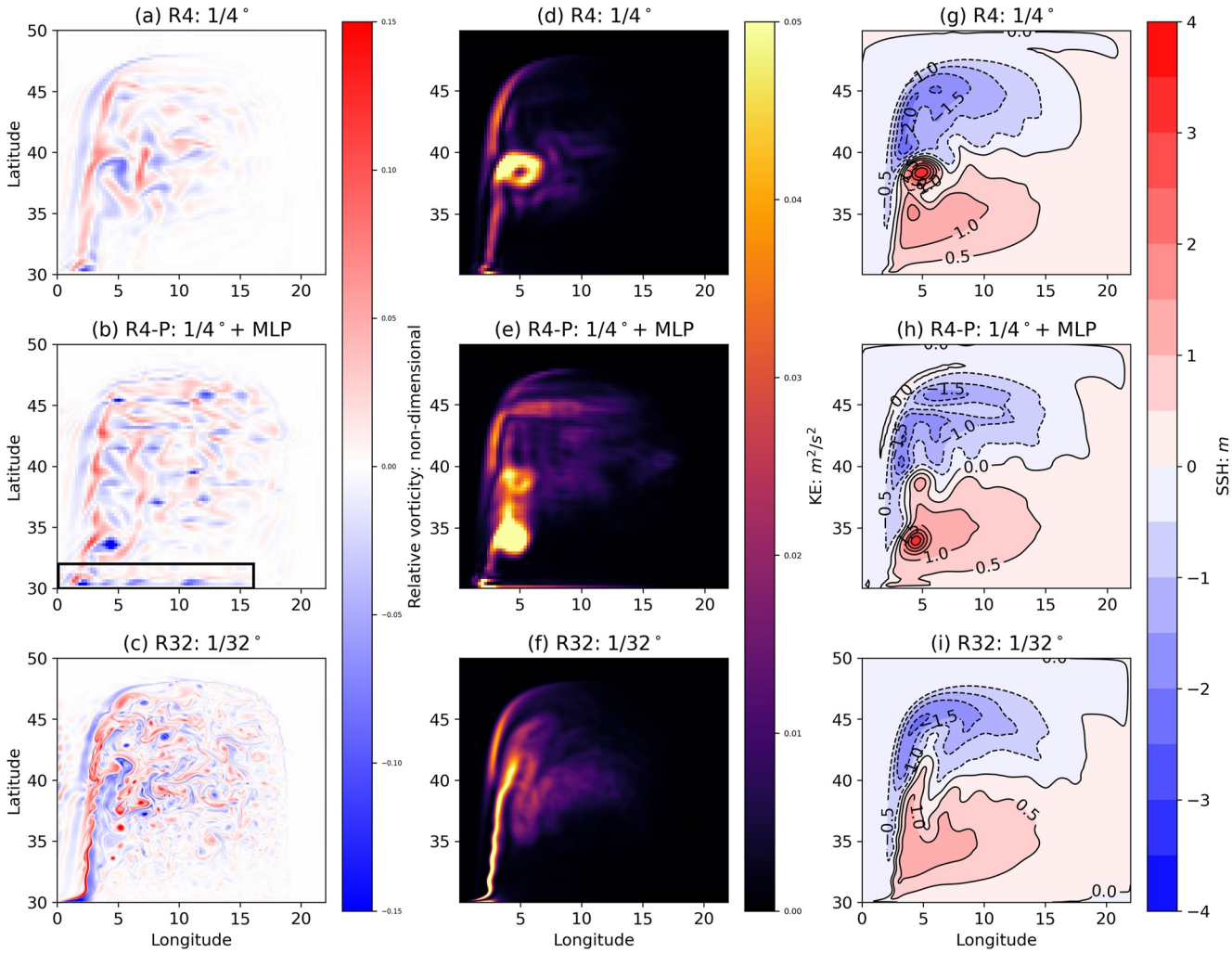


**Figure 3.** Snapshots of the upper layer relative vorticity (normalized by the planetary vorticity, a–c), 5-year averaged kinetic energy in [$m^2/s^2$](d)–(f), and 5-year averaged sea surface height in [m] (g)–(i), for coarse resolution model R4 (top row; a, d, and g), coarse resolution model with machine learned parameterizations R4-P (middle row; b, e, and h) and fine resolution model R32 (bottom row; c, f and i). The plots for R4-P (b, e, h) are from one ensemble member. Strong zonally sheared eddies highlighted by the black box in (b) will be discussed in Section 4.4.

model, R4, (a, d, and g) and fine resolution model, R32, (b, d, and f). The fine resolution model generates more energetic flow and finer-scale eddies. The time-mean flow, indicated by the time-mean sea-surface displacement, of R4 has a double gyre, but fails to simulate well the boundary current extension separating the gyres (see region around (5°E, 38°N)). In this section, we will focus on the performance of the stochastic parameterization in improving the boundary current and the under-energized flow for coarse grid models.

### 3.2. Results Without Tuning

The stochastic parameterization is implemented in R4, applied equally in both layers without tuning. To take advantage of the stochastic nature of the parameterization we run a 50 member ensemble with different random seeds. The models are run for the same duration as R32 and R4 to permit a direct comparison between runs at the same model time since rest. Examining and averaging an ensemble at the same model time avoids aliasing any systematic drift even though we did not find any significant long term trends. The stochastic subgrid momentum forcing implemented in MOM6 is calculated via Equation 4 in parameterized coarse-grid runs, but for runs without ML parameterizations only one trajectory is computed. We determined the ensemble sizes of 20 and 50 to be adequate for the various contexts, as explained in Text S3 in Supporting Information S1. In the figures throughout the rest of the paper, unless otherwise stated, KE time series and snapshots are obtained from a single

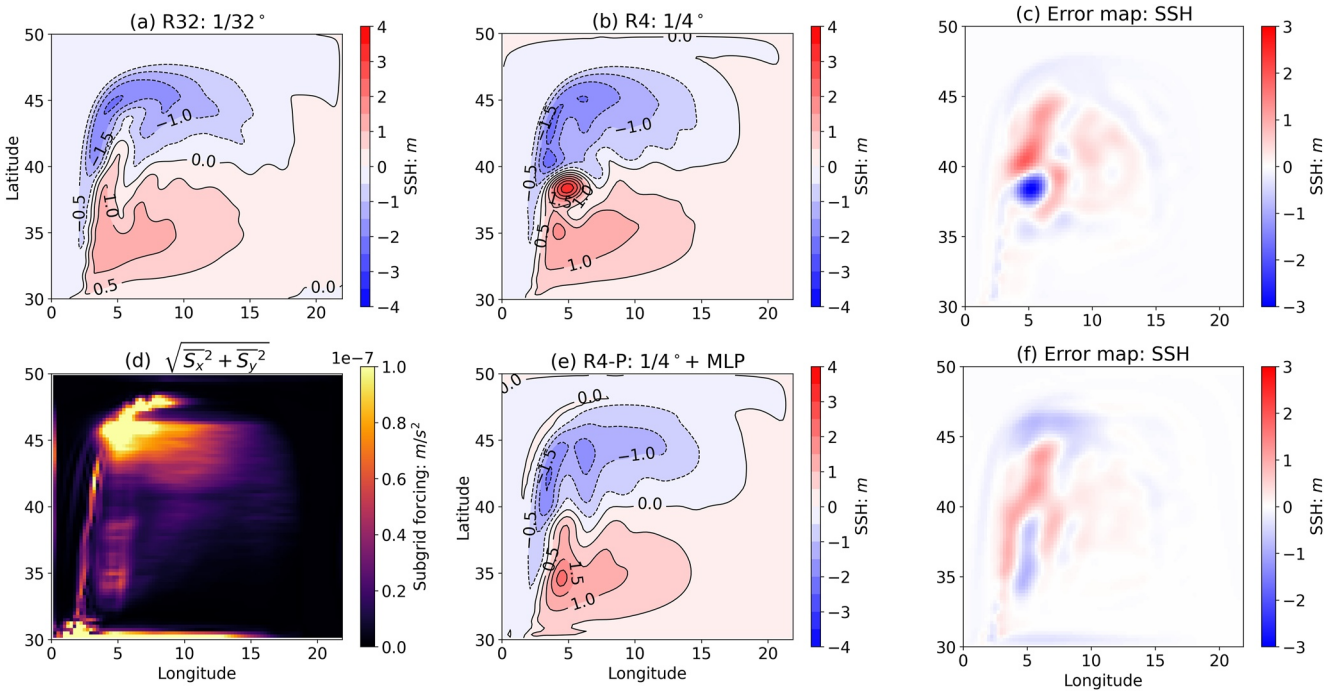


**Figure 4.** Comparison of 5-year averaged sea surface height (SSH) between the coarse resolution model with (R4-P,e) and without (R4,b) the subgrid parameterization and the target fine resolution model (R32,a). The error maps (c) and (f) are obtained by subtracting low-resolution SSH (with or without parameterization) from coarse-grained high-resolution SSH. The momentum forcing in the upper layer flow (d) is calculated as $\sqrt{\overline{S_x}^2 + \overline{S_y}^2}$, where $S_x$ and $S_y$ represent the subgrid momentum forcing in each direction averaged over the last 5 years. The momentum forcing (d) and R4-P SSH (e) are averaged from 50 ensemble members. MLP is short for the machine learned parameterization.

member of the ensemble, while the SSH maps are averaged across multiple ensemble members. The KE means are temporal means computed from the KE time series spanning the last 5 years. The SSH maps are averaged from SSH snapshots over the same 5-year period.

We use snapshots of upper layer relative vorticity and KE, shown in Figures 3b and 3e respectively, from the end of the one ensemble run, to present a qualitative assessment of the effect of the parameterization. We illustrate by showing only one of the ensemble members, but the other ensemble members produce similar statistics. Further details about the similarity of ensemble members are given in Text S2 in Supporting Information S1. The subgrid momentum forcing from the CNN model energizes the flow and introduces some small-scale eddies, and they are perhaps more comparable to the eddies in R32 (Figure 3c). Two striking features can be observed from the vorticity and KE maps. First, there is longitudinal stretching of some eddy features. It is possible that this is due to a statistical bias in the structure of eddies in the training data. Second, there are structures or artifacts on the southern boundary (highlighted by the black box, near a vertical wall), which are not observed near other boundaries where the topography is shallow. On the southern wall in both the vorticity and KE maps, for all members (we only show one example in the section), an unrealistic zonal structure is apparent. We will discuss the boundary condition problem in more details in Section 4.4. Figure 3h shows the SSH averaged over the last 5 years, for the same ensemble member. Randomness from Equation 4 leads to the different SSH patterns for each realization, especially in the region that we focus on (the separated boundary current). Broadly speaking, the patterns of SSH appear to be improved by the parameterization and more similar to the pattern of the fine resolution model (Figure 3i).

To more quantitatively assess the impact of the subgrid parameterization, we use two metrics, errors in the 5-year averaged SSH, and change in the KE spectra. The metrics used when training the CNN model's offline accuracy in GZ21 are to minimize the statistical moments of the momentum forcing. For individual realizations, a metric based on the local subgrid forcing is not meaningful. Instead, we use metrics more amendable to model evaluation that uses the model state. In Figures 4a–4c, we compare the 5-year averaged SSH between R4 and the fine resolution R32. To make a fair comparison between the results from different resolutions, both R4 and R32 SSH

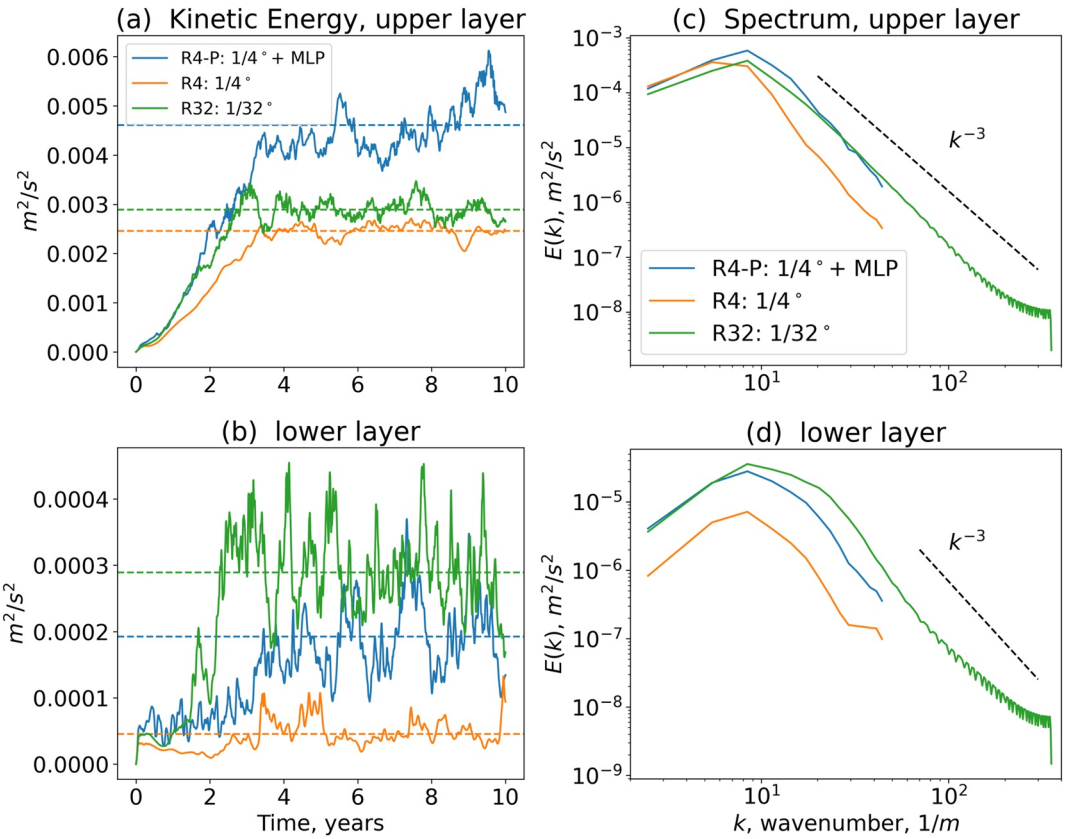


**Figure 5.** Comparison of kinetic energy (KE) time series (a) and (b) and spectra (c) and (d) for the flow upper layer (top row) and the lower layer (bottom row) between the coarse resolution model R4 (orange), fine resolution R32 (green) and the coarse resolution model with machine learned (ML) parameterizations R4-P (blue). The dashed lines in (a) and (b) are time mean values of KE over the last 5 years. The dashed lines in (c) and (d) are the spectral slope of KE spectrum corresponding to inertial interval of enstrophy. MLP is short for the ML parameterization.

are first filtered using a Gaussian kernel with the window size of 1°, and then the results of fine resolution R32 are subsampled to the grid of coarse resolution R4. In this paper, all comparisons between different resolutions undergo the above process. The fixed-size window of 1° facilitates a comparison of the parameterization performance across different resolutions since grid cells of 1° size do not resolve mesoscale eddies. The error map shows that the largest errors appear around the region of the separated boundary current near (5°E, 38°N). The CNN parameterization in the coarse model (hereafter R4-P) reduces the local error of the ensemble averaged SSH (Figures 4a, 4e, and 4f). The root mean square error (RMSE) of R4 SSH (relative to R32 SSH) is 0.2780 m and the RMSE of R4-P SSH is 0.2202 m. The magnitude of momentum forcing in the upper layer flow, calculated as $\sqrt{\overline{S_x}^2 + \overline{S_y}^2}$ using the time and ensemble averaged subgrid momentum forcing $S_x$ and $S_y$ (as described in Equation 4), is depicted in Figure 4d. The KE time series and spectra are compared between R4, R4-P and R32, in Figure 5. The coarse resolution model R4 has less energetic flow than R32. The CNN parameterization increases KE in both upper and lower layers, though the momentum forcing injects too much energy into the upper layer of the flow while injecting too little energy into the lower layer. The comparison of time-series made here uses only one of the parameterized ensemble members. The other ensemble members produce similar statistics (Text S2 in Supporting Information S1).

### 3.3. Applying the CNN Parameterization at Different Spatial Resolutions

In the CNN training procedure, the velocity from the fine resolution CM2.6 1/10° ocean grid was used to generate momentum forcing on the coarse resolution 1/4° grid of the CM2.5 model. As a result, the CNN might be considered "optimized" for the R4 resolution for the double gyre tests above. Parameterizations used in realistic

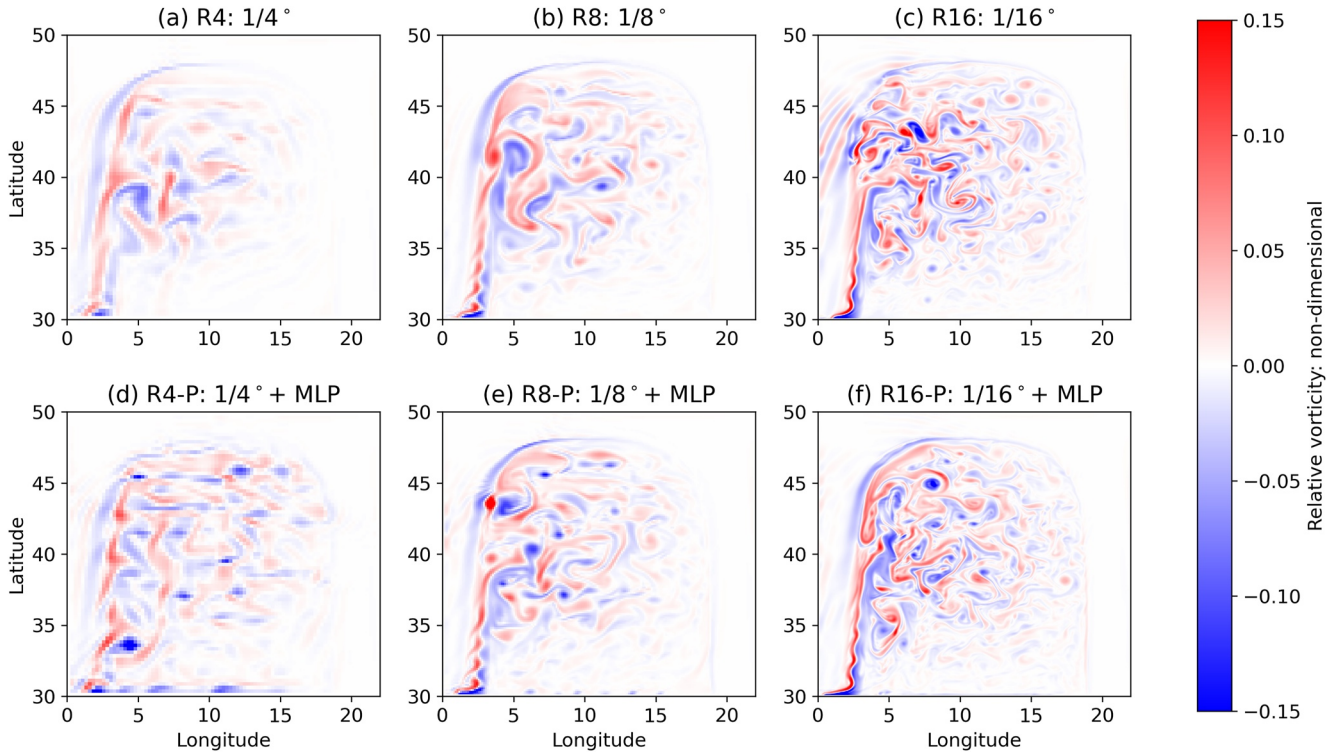


**Figure 6.** Snapshots of upper layer relative vorticity (normalized by the planetary vorticity) without any subgrid parameterizations (a–c) or with the machine learned (ML) parameterization (d–f) in simulations with 3 different horizontal resolutions. The plots for R4-P (d), R8-P (e) and R16-P (f) are from one ensemble member. The grid sizes of the simulations are 1/4° (R4, a and d), 1/8° (R8, b and e) and 1/16° (R16, c and f). MLP is short for the ML parameterization.

ocean circulation models will likely be deployed at a range of spatial resolutions and even need to accommodate variable spatial resolutions within one model.

To investigate the applicability of the CNN subgrid parameterization at different grid resolutions, we test the model against the grids ranging in size from 1/4° (R4) to 1/16° (R16). Figure 6 shows the snapshots of relative vorticity at the upper layer flow for different spatial resolutions. The three runs in (a–c) have no parameterized momentum forcing, and the three runs in (d–f) have the stochastic CNN parameterization. At all resolutions, small scales are qualitatively modified relative to the counterpart without ML parameterizations. As the spatial resolution is refined, the amplification by energy-injection appears to diminish; the CNN stochastic momentum forcing injects lots of energy in R4, but hardly any in R16. This is more obvious in the plots of the total KE time series (Figure 7). In the upper layer flow, the R4 case has significantly less KE (∼17%) than the R32 case, and the parameterization overcompensates for this so that R4-P has almost ∼50% too much KE. The intermediate resolution cases R8 and R16 have nearly identical total KE to that of R32. In the lower layer flow, R4 also has smaller KE than R32, and the parameterization does increase the KE (R4-P), but in contrast to the upper layer, the parameterization does not add enough. As for the upper layer, the parameterization has minor effects on the lower layer KE for both R8 and R16. The KE spectra in Figure 8 and the 5-year averaged SSH in Figure 9 show the similar diminishing trend that the finer the grid resolution, the less effect the parameterization has on the flow.

### 3.4. Tuning the Parameterization for Online Performance

It is a common practice to tune a simulation by scaling a parameterization to optimize some metrics. The simplest form of scaling is to multiply the parameterized accelerations by a fixed factor that will either amplify or attenuate depending on whether the scaling factor is larger or less than 1. As shown in Figure 5, the momentum parameterization in R4-P over-energizes the upper layer flow, but under-energizes the lower layer flow. We consider two strategies to tune the parameterization. In the first strategy, we attenuate the momentum forcing by multiplying it for both layers by the same constant coefficient, ranging from 0 to 1, as done in (Zanna & Bolton, 2020). The metric we use to measure the attenuation is the integrated KE for each layer, averaged over the last 5 years.

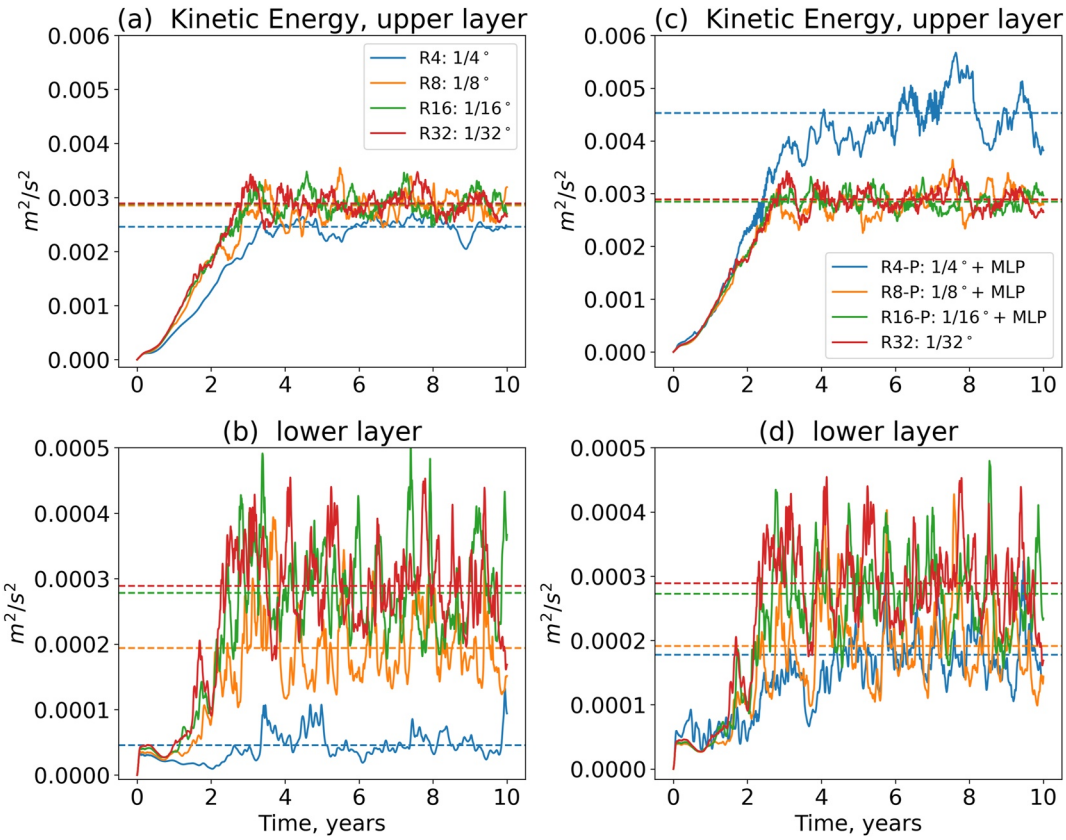


**Figure 7.** Kinetic energy (KE) time series comparison, across different horizontal resolutions, of coarse resolution simulations against the finest (truth) 1/32° resolution (red): (a), (b) coarse resolution simulations without subgrid parameterizations for the upper and lower layer; (c), (d): coarse resolution simulations with subgrid machine learned (ML) parameterizations. The dashed lines are mean values of KE over the last 5 years. MLP is short for the ML parameterization.

Figure 10 shows the sensitivity of the 5-year averaged KE to vertically uniform attenuation of the momentum parameterization. In general, an increase in the strength of parameterization results in more energization of the flow, that is, this subgrid parameterization represents KE backscatter, see Frederiksen and Davies (1997), Berner et al. (2009), Thuburn et al. (2014), Jansen and Held (2014), Zanna et al. (2017), Juricke et al. (2020), and Zanna and Bolton (2020). The sensitivity of time-averaged KE to parameterization strength appears to be different for the upper and lower layer flows. The upper layer flow becomes more strongly sensitive to the attenuation coefficient about 0.6, and provides optimal energization at ~0.75, whereas the lower layer flow is relatively insensitive until the attenuation of 0.8 and would apparently require an amplification coefficient greater than 1. Therefore, there is no shared value of scaling coefficient that can optimize the solution in both layers.

The second tuning strategy we consider uses two different scaling coefficients, one for each layer. Again, we use the time-averaged integrated KE for each layer as a metric. The attenuation coefficient for the upper layer forcing is varied from 0.5 to 0.9, while the amplifying coefficient for the lower layer is varied from 1.3 to 1.7. Figure 11 shows a 2D sensitivity map where the $x$-axis is the upper layer attenuation coefficient and the $y$-axis is the lower layer amplification coefficient. The color values are the KE difference relative to KE of R32, and we refer to it as relative KE. At each point (i.e., for each pair of upper and lower layer scaling numbers) 20 ensemble runs were carried out, and the change in ensemble-averaged KE is plotted. The energy in both layers does not increase in a strictly linear fashion as the scaling number increases. If the response to layer amplification was linear then the white band in Figure 11a would be parallel to a vertical line, and the band in Figure 11b would be parallel to a horizontal line. The energy increases slightly slower in the upper layer when the lower layer amplification coefficient is larger, while the energy increases in the lower layer somewhat slower when the upper layer attenuation coefficient is larger. In other words, the scaling of top layer can influence the lower flow, and vice versa. Despite the influence between layers, the sensitivity for each layer is dominated by that layer's scaling coefficient. For

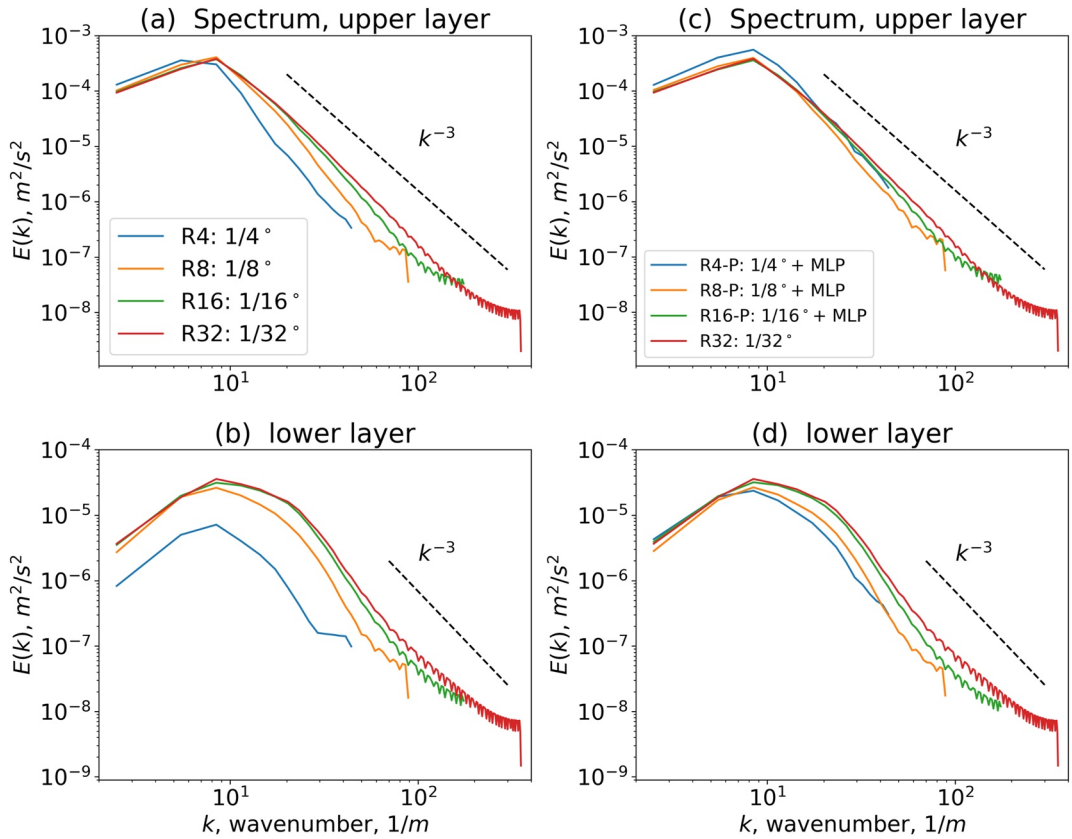


**Figure 8.** (a) and (b) Comparison of kinetic energy (KE) spectra for flow without subgrid parameterizations versus target fine resolution flow; (c) and (d) Comparison of KE spectra for flow with subgrid parameterizations versus target fine resolution flow. The dashed lines are the spectral slope of KE spectrum corresponding to inertial interval of enstrophy. MLP is short for the machine learned parameterization.

this case with the specific resolution and metric, the upper layer scaling number is 0.7827 and the lower layer number is 1.5164. Using this set of scaling numbers, the momentum forcing parameterization vastly improves mean KE and its spatial spectrum (see Figure 12). The mean of KE time series for R4-P almost exactly matches the KE mean for R32, and the KE spectra for R4-P are closer to the target. This sensitivity analysis shows that it is possible to retroactively tune the ML parameterization of momentum forcing to optimize some metric of the ocean model solution.

## 4. Problems and Remedies

In the two-layer double gyre tests of the GZ21 parameterization (Section 3), we find the parameterization can be made to work well but might be limited in generality and has some artifacts at boundaries. We will discuss distinct aspects of our results, noting challenges and suggest some remedies here or for future work.

### 4.1. Parameter Optimization and Vertical Structure

Without attenuation the GZ21 parameterization over-energizes the upper layer flow and under-energizes the lower layer flow. That tuning is needed at all is not unexpected, with many conventional and ML parameterizations performing differently between "offline" and "online." All parameterizations are ultimately tuned. In the online test by GZ21 it appears a parameterization trained on surface fields was reasonably effective for a barotropic model. Here, we essentially tested the hypothesis that the interior momentum forcing was functionally similar to the surface momentum forcing, and whether the momentum forcing could be treated independently layer by layer, that is, decoupled in the vertical except through correlations between the layer flows. We find that vertical structure is needed since tuning yielded significantly different scaling coefficients for the two layers (attenuation

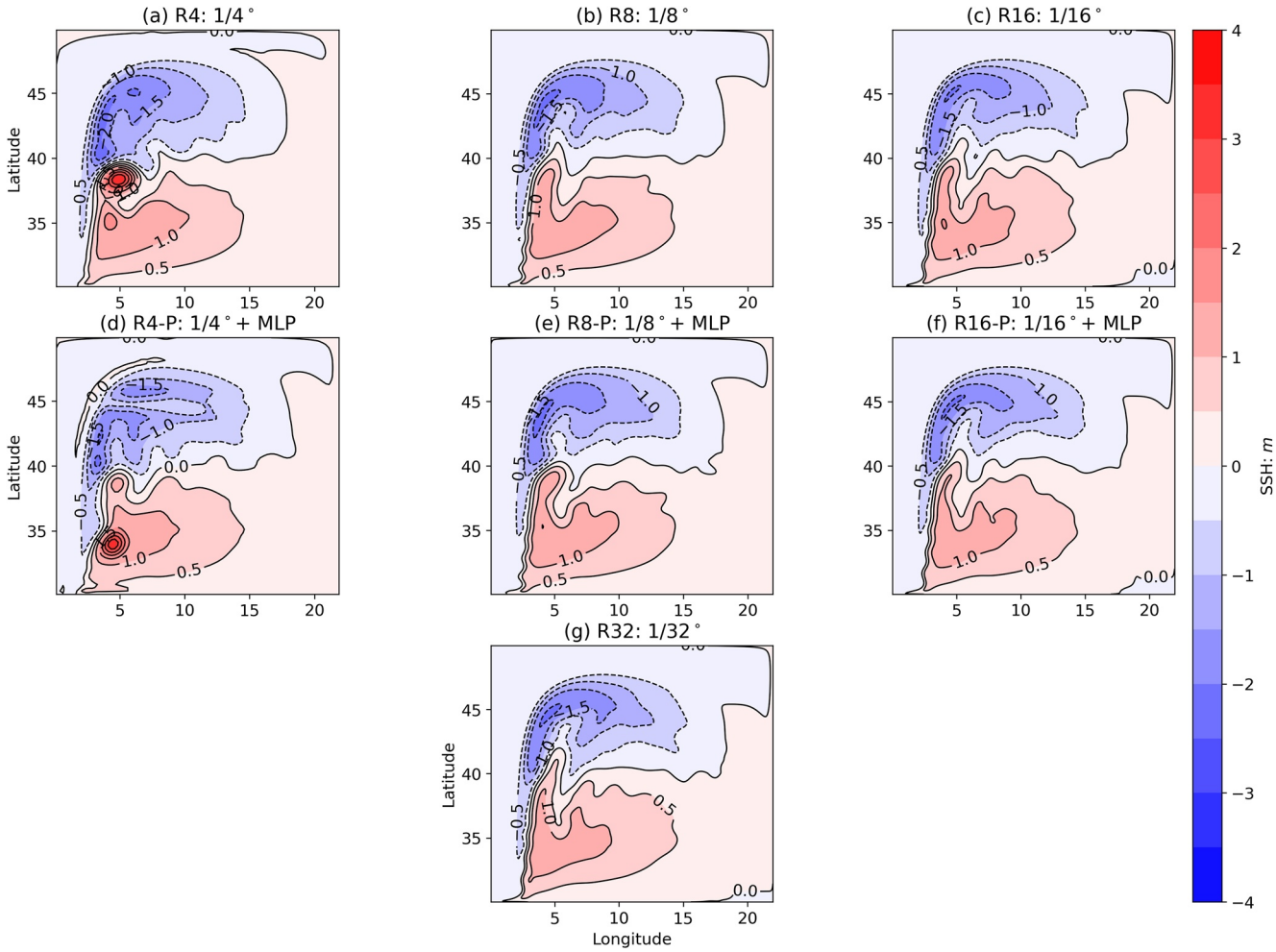


**Figure 9.** Five-year averaged sea surface height maps for the flow without subgrid parameterizations (a, b and c) or with subgrid parameterizations (d, e, and f). The grid sizes of the simulations are 1/4° (R4, a and d), 1/8° (R8, b and e) and 1/16° (R16, c and f). The plots for R4-P (d), R8-P (e) and R16-P (f) are from one ensemble member. The plot from the 1/32° simulation (R32, g) is included as the ground truth for comparison. MLP is short for the machine learned parameterization.

for the upper layer, amplification for the lower layer). Here, we could afford to find the optimal combination of just two scaling values that yield the "best" coarse resolution model with the CNN parameterization, using the time-averaged integrated KE as a metric (Figure 11).

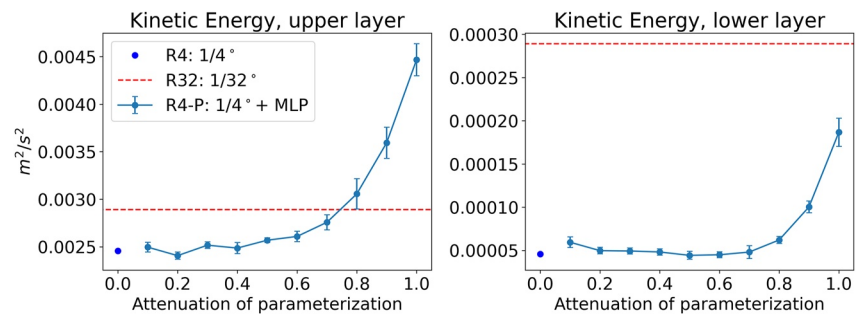


**Figure 10.** Sensitivity of 5-year time-averaged kinetic energy (KE) to vertically uniform attenuation of the momentum forcing. The error bar represents the standard deviation of KE among 20 ensemble members for each value of attenuation. MLP is short for the machine learned parameterization.

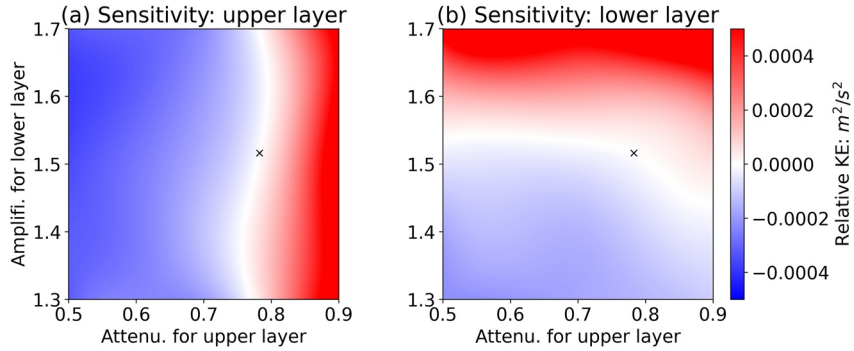


**Figure 11.** Sensitivity of 5-year time-averaged kinetic energy (KE) to vertically nonuniform scaling of the momentum forcing for the grid size of 1/4°. The *x*-axis is the attenuation applied to the upper layer forcing, and the *y*-axis is the amplification applied to the lower layer forcing. The KE is relative to KE for the target fine resolution case R32. The cross represents the scaling factor that minimize relative KE in both layers.

### 4.2. Resolution Dependence and Scale-Awareness

The optimal tuning indicated in Figure 11 is for the spatial resolution of 1/4°. In Section 3.3 we asked if the parameterization performed well at other spatial resolutions. We noted that at finer resolutions the parameterized momentum forcing is diminished. This resolution dependence might be coming from the change in flow structure and amplitude at different resolved scales. We repeat the tuning exercise for the spatial resolution

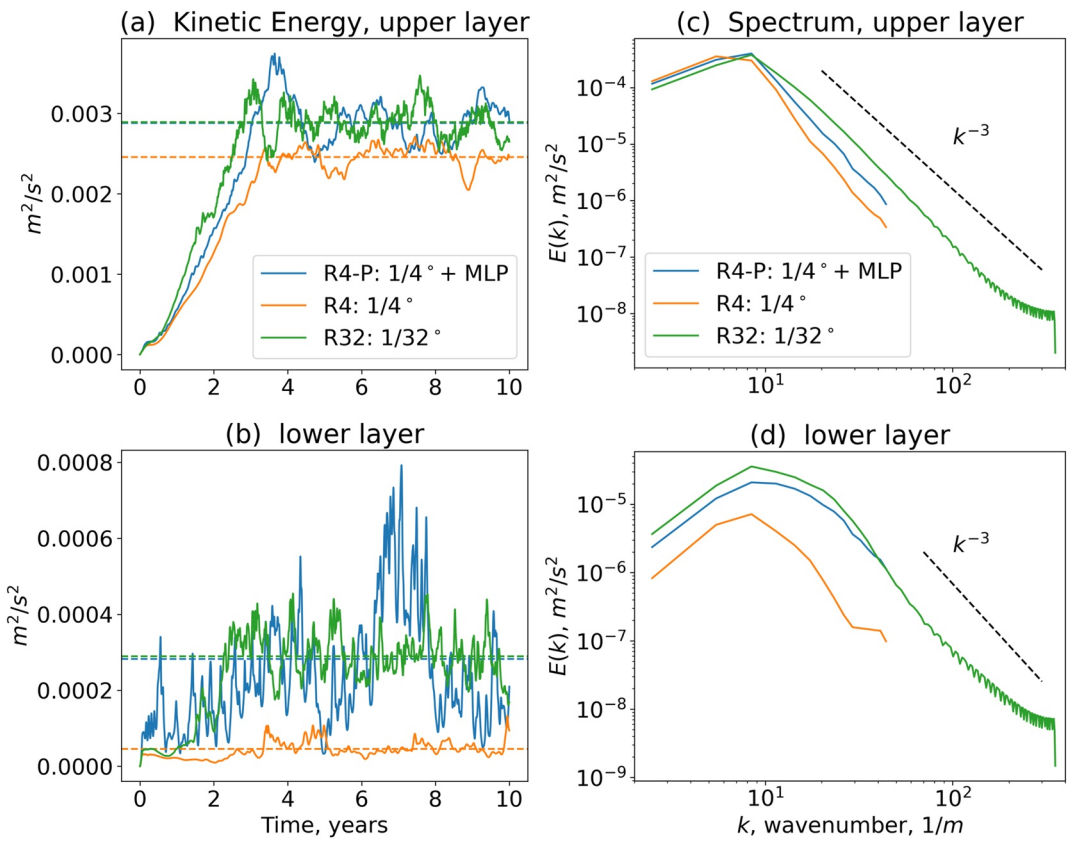


**Figure 12.** Comparison of kinetic energy (KE) time series (a) and (b) and spectra (c) and (d) for upper layer (top row) and the lower layer (bottom row) between the coarse resolution model R4 (orange), fine resolution R32 (green) and the coarse resolution model with the optimal scaling of momentum forcing R4-P (blue). The dashed lines in (a) and (b) are mean values of KE over the last 5 years. The dashed lines in (c) and (d) are the spectral slope of KE spectrum corresponding to inertial interval of enstrophy. MLP is short for the machine learned parameterization.

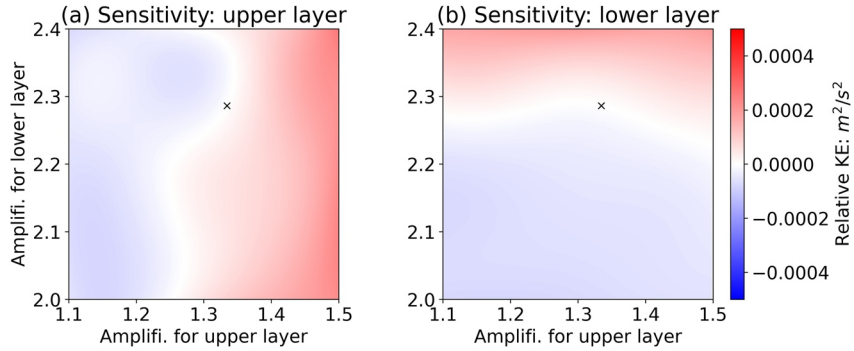


**Figure 13.** Sensitivity of 5-year averaged kinetic energy to scaling of the momentum parameterization for the grid size of 1/8°. The definitions of *x*-axis, *y*-axis, the variable in the map and the cross are same to Figure 11.

of 1/8°, varying the layer-wise scaling coefficients to optimize the time-mean intergrated KE (Figure 13). The sensitivity patterns are broadly similar to those in Figure 11 but with smaller amplitudes, indicating less sensitivity. The coefficients that optimize the time-mean KE of R8-P to be most similar to that of R32 (cross in Figure 13) are an upper layer amplification of 1.3345 and a lower layer amplification of 2.2862. Here, the upper layer in R8-P needs amplification while in R4-P the upper layer needed attenuation. If the relationship between grid size and scaling factor is assumed to be linear, the slope of the regression line for upper layer scaling numbers to the grid sizes is −4.6987 (scaling number = −4.6987 × grid size + 1.9048), while the slope of the lower layer scaling numbers to the grid sizes is −5.9207 (scaling number = −5.9207 × grid s ize + 2.9635). We repeat the tuning at the spatial resolutions of 1/5°, 1/6° and 1/7°, and plot the optimal scaling coefficients in Figure 14a. We find a broadly linear fit with increasing amplification as resolution is refined. This aligns with our expectations that the parameterization impact will taper off as the resolution gets finer. However, the energy injection from the CNN model decreases with refined resolution faster than needed to correct the energy gap between the model without parameterization and the 'truth'. The stronger amplification with finer resolution compensates for the imperfect scaling with resolution. Figure 14b shows the difference between KE from the optimal scaled parameterization ($KE_r$-P) and from no parameterization ($KE_r$) which is an integral measure of how much work the parameterization has done. The measure of work tends to decrease with finer resolution even though the scaling factor gets larger. In comparison to the trend, the KE difference for R5 is relatively small. We have spent time examining these experiments more closely and still do not have any good ideas to explain this result. Our intention here is not to establish an empirical formula for future optimization, but rather to analyze the results and uncover potential patterns in the optimization process across resolutions ranging from 1/4° to 1/8°. The observed trend may vary when applying the optimization to a broader range of resolutions or different test cases, beyond the scope of the double gyre case that was examined.

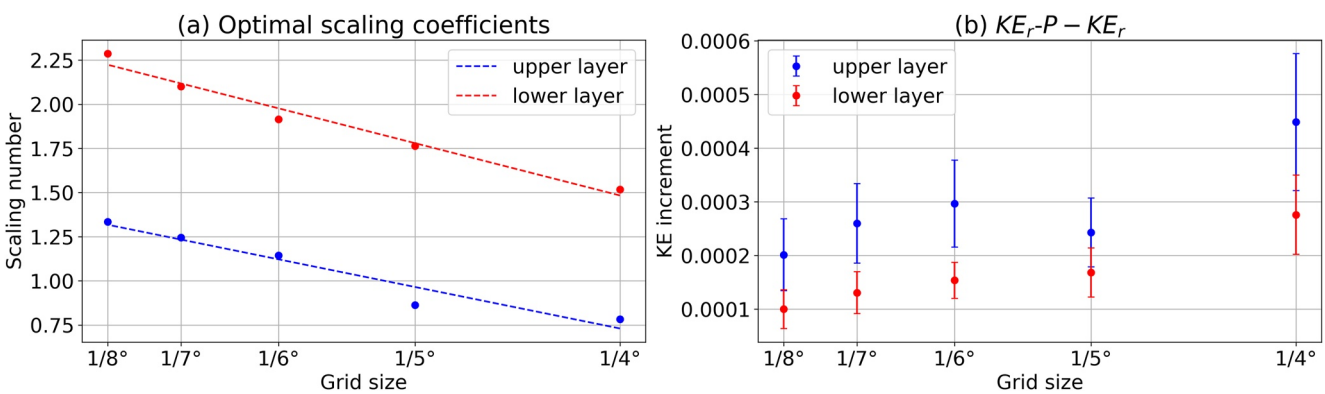


**Figure 14.** (a) Relation between optimal scaling numbers to subgrid parameterization and grid sizes of the double gyre simulations; (b) Relation between kinetic energy (KE) increment from the optimal scaled parameterization and grid sizes; the error bar represents the standard deviation of KE among 20 ensemble members for optimal scaling number.

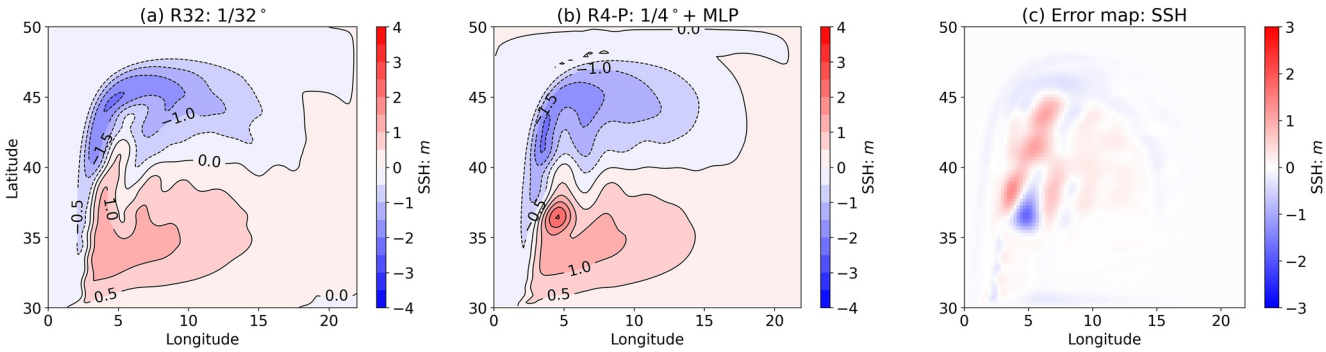


**Figure 15.** Comparison of 5-year averaged sea surface height (SSH) between the coarse resolution model with the scaled subgrid parameterization (the upper layer scaling number is 0.7827 and the lower layer number is 1.5164, indicated by the cross in Figure 11) and the target fine resolution model R32. The error map is obtained by subtracting low-resolution SSH with the optimal parameterization from coarse-grained high-resolution SSH. The map of R4-P SSH (b) is averaged from 20 ensemble members. MLP is short for the machine learned parameterization.

### 4.3. Role of Metric in Evaluation

So far we have only used the time-averaged integrated KE as a metric for tuning the scaling of momentum forcing. Qualitatively, other aspects of the solution improve when the total KE is optimized. Figure 15 shows the difference between the 5-year averaged SSH of R4-P and R32 using the best scaling of momentum forcing based on the optimized KE, where the upper layer scaling number is 0.7827 and the lower layer number is 1.5164 (indicated by the cross in Figure 11). The scaled parameterization improves this metric if we look at RMSE of the error map where the RMSE value is now 0.2034 m, down from 0.2202 m for the parameterization without scaling (Figure 4f). Table 2 shows the SSH improvement based on the RMSE of error maps for the various grid sizes from 1/4° (R4) to 1/8° (R8). For all resolution models, we find that the best-scaled parameterizations based on the metric of KE also improve the metric of SSH. While the best scaling numbers for KE also improve SSH, these numbers are not the best scaling numbers for SSH. Figure 16 depicts the optimal scaling numbers for R4 and R8 based on another metric, that is, RMSE of SSH deviation. The process of generating the figure is similar to Figure 10, but with a change of variable from KE to time-averaged SSH. The scaling numbers used to optimize the SSH metric are different from those used to optimize the KE metric. Furthermore, the patterns in the 2D maps are less coherent, in contrast to the patterns in KE sensitivity maps in Figures 11 and 13.

As for many conventional parameterizations, we find the parameterization of momentum forcing to be able to improve different aspects of the solution but to different degrees and not necessarily optimally together. The parameterization injects momentum and KE so we should expect to be able to have a direct effect on total KE. The parameterization has a more indirect control over time-mean sea-surface height (through geostrophy if any) and we find less coherent response in the RMSE SSH. Neither metric was used in the training of the CNN in GZ21, so the result that we can optimally tune total KE, whilst observing a modest reduction in RMSE SSH, is therefore a success for the parameterization.

**Table 2**
*Improvement of Time-Mean Sea Surface Height for Scaled Momentum Forcing at Various Spatial Resolutions, Based on Optimal Scaling Factors for Kinetic Energy*

| | No Param. | bg Param. Without scaling | | bg Param. With best scaling | |
|---|---|---|---|---|---|
| Res. | RMSE (m) | RMSE (m) | Improved(%) | RMSE (m) | Improved(%) |
| 1/4° | 0.2780 | 0.2202 | 20.7914 | 0.2034 | 26.8345 |
| 1/5° | 0.2093 | 0.1827 | 12.7090 | 0.1706 | 18.4902 |
| 1/6° | 0.1432 | 0.1305 | 8.8687 | 0.1253 | 12.5000 |
| 1/7° | 0.1167 | 0.1088 | 6.7695 | 0.1001 | 14.2245 |
| 1/8° | 0.0971 | 0.0905 | 6.7971 | 0.0891 | 8.2389 |

### 4.4. Effect of Lateral Boundaries on CNN Inference

In Section 3.2 we noted the CNN parameterization induced artifacts at the wall boundaries. Strong zonally sheared eddies highlighted by the black box in the left plot of Figure 6 are not realistic, with no counterpart in the fine resolution model results. The training data used by GZ21 was from limited regions of the CM2.6 model and deliberately excluded any coastal waters or land. Therefore, by construction the parameterization was not trained to "know" what to do near boundaries. We hypothesize that the four regions selected exhibit a tendency toward zonal flows and that this might explain the zonal elongation of eddies when using the parameterization, and the exclusion of coastal waters in the four selected regions contributes to the boundary artifacts. To better illustrate the boundary artifacts near model coastlines due to the CNN parameterization, we perturb the

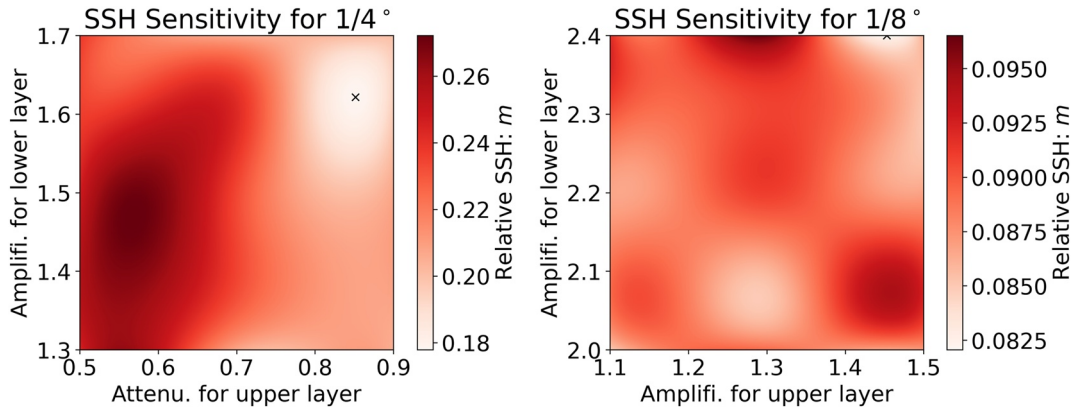


**Figure 16.** Scaling optimization of momentum parameterization based on 5-year averaged sea surface height (SSH). The *x*-axis is the amount of prescribed scaling for upper layer flow and the *y*-axis is the amount of the scaling for lower layer flow. The cross represents the fitting numbers that minimize root mean square error of this SSH to R32 SSH.

double gyre test by adding a box in the middle of the domain (positioned from 8.5° to 13.5° in longitude and 37.5°–42.5° in latitude, see Figure 17) with vertical walls. This is a severe topographic obstacle in the path of the wind-driven jet and we expect it to test the limits of the CNN parameterization. A snapshot of the upper layer relative vorticity shows how much the new geometry affects the coarse R4-P model using the parameterization. Strong sheared structures can be seen both around the box island as well as at the southern boundary as before (Figure 18a). Introducing the box island to the fine resolution R32 model does not develop any comparable structures (Figure 18b). The KE time series and spectra (Figure 19) also suggest that the parameterization over-energizes the flow close to the boundary. As before without the box island, the CNN parameterization injects too much energy into the upper layer, but also now in the lower layer. The limit of the parameterization near wall boundaries is also evident from the time-mean SSH (Figure 20). The RMS difference between R4-P SSH and R32 SSH is increased to 0.2503 m from 0.1765 for R4 SSH (without parameterization), which makes matters worse.

We believe that re-training the same CNN model using velocity data from the entire globe might address the issue. However, extending to the global domain raises several questions. First, the volume of data that will be used in the retaining process is roughly 40 times greater than that from the four subdomains used to train the current CNN model (GZ21). This extension in geographic coverage presumably increases the cost of training the CNN, but also extending the training to cover multiple depths adds at least another order of magnitude to volume and cost. Secondly, CNN models are not the most straightforward ML models to incorporate boundary conditions. Convolutional Neural Network models involve sliding fixed-size kernels over images to extract local features, which restricts their applicability for problems with complex boundary conditions and topography. When training a CNN model with global data, handling land points consistently between training and inference becomes critical. One option is to set the velocity components at the land points to 0 (which is what we did here for inference). However, the precision of training at the wet points that have land points in their $21 \times 21$ stencil will be reduced or lost entirely. The choice used in GZ21 is to exclude from the training data anywhere that the stencil includes land points, and for consistency then similarly not do inference near land points. This leads to a bias in momentum forcing wherever the parameterization is essentially zeroed out; for the GZ21 network this is losing missing out points within 20 cells of the coasts (which is of order 120–200 km in distance). Another option to handle land and lateral boundary conditions is to provide a mask channel but it is unclear how the out-of-sample problem would manifest when encountering a mask pattern unseen in the training data. Physical lateral boundary conditions are often expressed in terms of normal or tangential gradients; CNN models generally lack understanding of the underlying grid metrics and often assume convolution on an equally spaced grid tensor. This raises another limitation

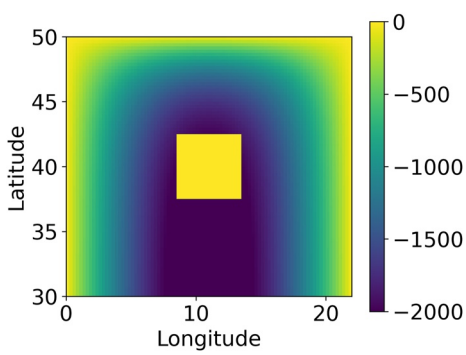


**Figure 17.** Plan view (latitude, longitude) of the bathymetry with a box in the middle of the domain for wind-driven double gyre.

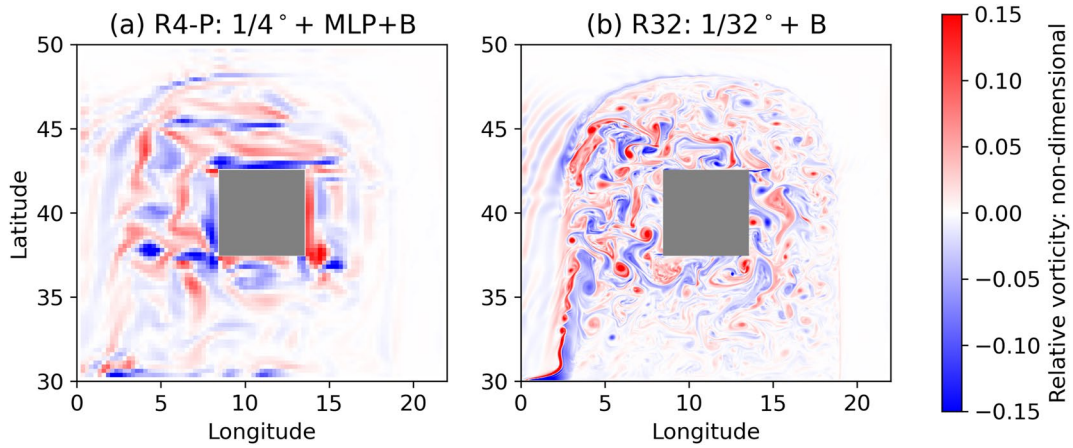


**Figure 18.** Snapshots of the upper layer relative vorticity (normalized by the planetary vorticity) at the end of perturbed-topography simulations from the coarse resolution model with machine learned (ML) parameterizations R4-P (a) and the fine resolution R32 (b). The gray rectangle indicates the region where the unrealistic eddies are generated. MLP is short for the ML parameterization and B is short for the box (gray rectangle).

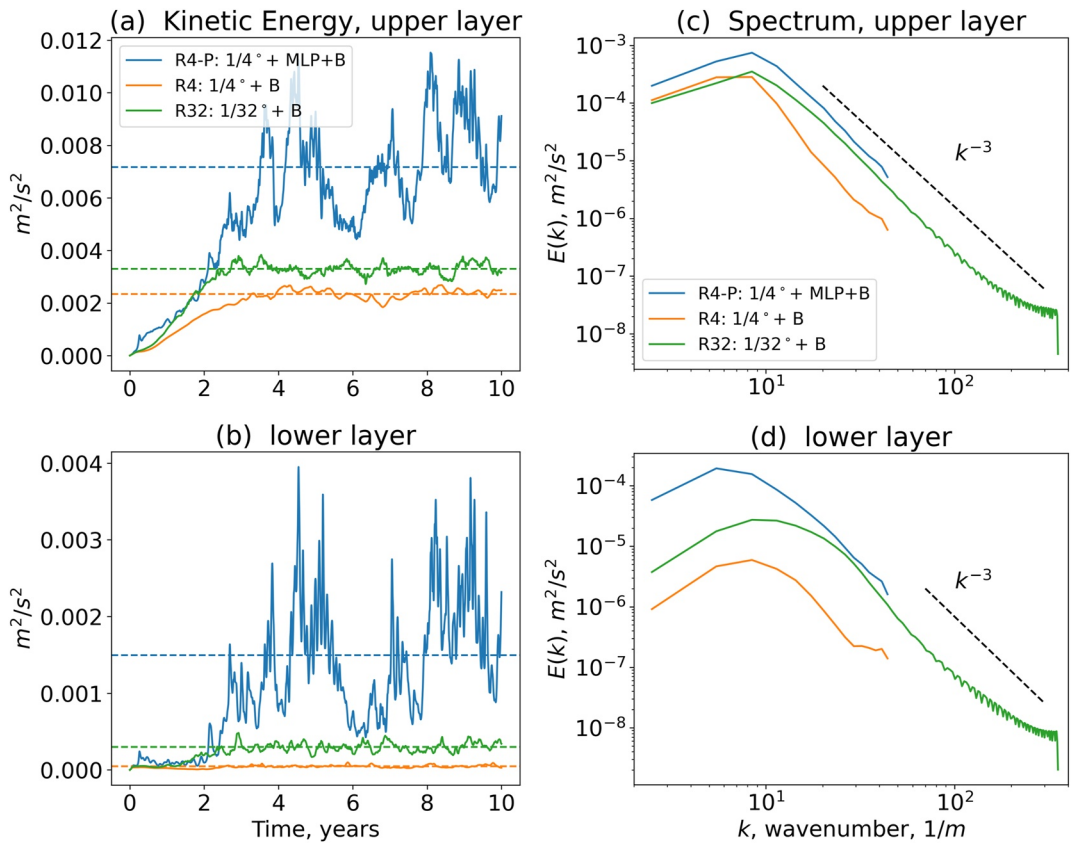


**Figure 19.** Comparison of kinetic energy (KE) time series (a) and (b) and spectra (c) and (d) for perturbed-topography tests at the flow upper layer (top row) and the lower layer (bottom row) between the coarse resolution model R4 (orange), fine resolution R32 (green) and the coarse resolution model with machine learned (ML) parameterizations R4-P (blue). The dashed lines in (a) and (b) are mean values of KE over the last 5 years. The dashed lines in (c) and (d) are the spectral slope of KE spectrum corresponding to inertial interval of enstrophy. MLP is short for the ML parameterization.

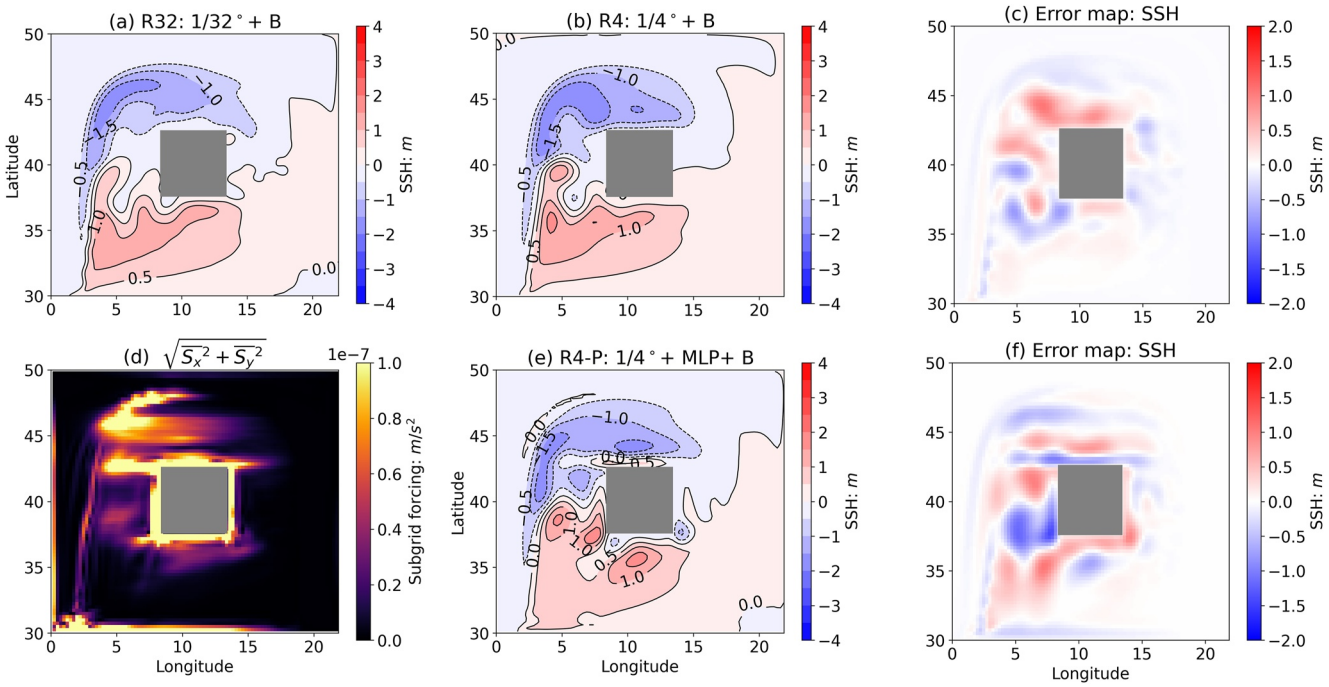


**Figure 20.** Comparison of 5-year averaged sea surface height (SSH) for perturbed-topography tests between the coarse resolution model with (R4-P,e) and without (R4,b) the subgrid parameterization and the target fine resolution model (R32,a). The error maps (c) and (f) are obtained by subtracting low-resolution SSH (with or without parameterization) from coarse-grained high-resolution SSH. The momentum forcing in the upper layer flow (d) is calculated as $\sqrt{\overline{S_x}^2 + \overline{S_y}^2}$, where $S_x$ and $S_y$ represent the subgrid momentum forcing in each direction averaged over the last 5 years. The momentum forcing (d) and R4-P SSH (e) are averaged from 50 ensemble members. MLP is short for the machine learned parameterization and B is short for the box (gray rectangle).

of CNNs: while assuming the grid in a 21 × 21 stencil is approximately equally spaced is a reasonable start, an orthogonal curvilinear grid needed for sphere does not have equidistant celles, especially near the two poles.

A natural next step to address the boundary artifacts would be to re-train the same CNN model with the global data so that it may interpolate between "known" states (instead of extrapolating the forcing in the current CNN model) in the model inference process and avoid this possible "out of sample" issue. However, before initiating the training process for this global CNN model, we recommend carefully considering and solving all the aforementioned problems.

### 4.5. Cost Model for CPU and GPU Implementations

The computation of the CNN model inference may involve many more floating point computations than the dynamical model itself. Many conventional closed-form parameterizations typically cost a small fraction of the dynamical model so the potentially high cost may appear to be prohibitive to adopting neural network based parameterizations. The total time complexity (He & Sun, 2014) of one time step inference is

$$O\left(\sum_{l=1}^{d}\left(w_{l-1}\cdot s_l^2+1\right)w_l\right) \tag{6}$$

where $l$ is the index of a convolutional layer, $d$ is the depth (number of convolutional layers), $w_l$ is the number of output channels (also known as "width") for the $l$-th layer, $w_{l-1}$ is the number of input channels of the $l$-th layer, and $s_l$ is the spatial size of the filter. This formula counts the numbers of weights needed to describe the neural network and allows us to estimate the approximate number of floating point operations (FLOPs) assuming for convenience that a multiply add pair counts as a single operation. The network of GZ21 we use has $s_l = 5, 5, 3, 3, 3, 3, 3, 3$ and $w_l = 128, 64, 32, 32, 32, 32, 32, 4$. The first layer has two inputs (namely the $u$ and $v$ components of flow) and the four outputs of the last layer correspond to the mean and standard deviation of the zonal and meridional momentum forcing. The inference for our CNN model requires at least 268,005 in FLOPs for each grid point of the dynamical model (each point requires a 21 × 21 stencil), which is significantly more operations

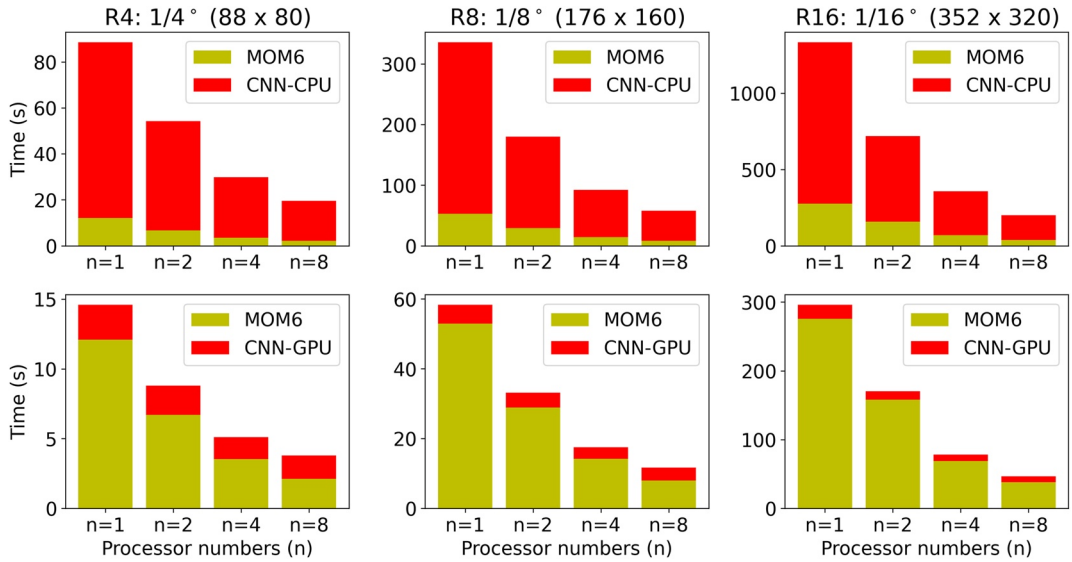


**Figure 21.** Cumulative processing time of the Convolutional Neural Network inference on central processing unit (top panel) or graphical processing unit (bottom panel) and MOM6 simulations at various grid resolutions and parallel processor counts.

than what is required by conventional parameterizations, and is even more than that of the dynamical model itself (typically on the order of hundreds to thousands). The stacked bar charts in Figure 21 show the measured processing time spent computing the CNN inference and dynamical core, for various spatial resolutions and parallel MPI processes. The upper panels of Figure 21 show the CNN inference processing time is around $O(10)$ times that of the dynamical core, for this simple two-layer double gyre case. Note that the CNN inference is on the same CPUs as those the dynamical core is running on. This ratio of times is essentially constant over a range of grid resolutions.

The above results for cost of inference on CPUs are prohibitive for most applications for global or regional simulations. Most machine learning applications utilize GPUs which work well on the tensor-like operations within a neural network. Typically, one GPU is only accessible by one CPU processor at a time, and the rest of the CPU processors must wait in queue. CUDA (a computing platform developed by NVDIA for GPUs) provides the Multi-Process Service (MPS) which allows multiple CPU processors to access a GPU card. This allows us to run the dynamical model on multiple CPU processors and move the CNN inference to a shared GPU and that can call CNN computation asynchronously. With this strategy, we find the processing time for inference is dramatically decreased (around 1/5 in wall-clock time). As shown in Figure 21, the cumulative processing time required for CNN inference on GPUs (lower panels) is considerably less than that of the dynamical core running on the CPUs, and the ratio of time on CNN decreases as the grid resolution is increased.

Although utilizing GPUs for the CNN inference is efficient, various challenges remain to prevent widespread adoption. Currently, MPS only permits a maximum of 16 CPU processors per GPU (https://docs.nvidia.com/deploy/mps/index.html). This restriction complicates the implementation of data transfer because subdomains for MPI exchange on CPU would necessarily be different to the subdomains for CNN inference on GPUs when more that 16 CPUs are used. The solution to address this issue of having only 16 CPUs per GPU was to develop an interface in MOM6 that enables the collection of data in specific root processors from other MPI processors, and subsequently transferring this data to Python. This interface effectively overcomes the limitation, allowing us to utilize more CPUs per GPU. However, in this study, we only used a maximum of 32 CPUs and 2 GPUs, as they were sufficient for all the cases examined. Nonetheless, we intend to employ this interface for global simulations in the future. A practical matter is that not every computer or cluster has a GPU card directly attached, or indirectly available. In this instance, inference has to occur on the CPUs and so the number of weights in the network (which most directly controls the computational cost) has to be limited. We could make trade-offs between the depth, number of filters, and filter size within the CNN model in order to balance accuracy and implementation costs. Such considerations are usually part of the hyper-parameter

tuning during the training process but the restrictions imposed by the cost of inference on CPUs would significantly change the balance of factors.

Related to the number of weights is the volume of data that need to be communicated laterally between parallel processes so that the inputs to the CNN are all valid across the full stencil. For the GZ21 network, each output point has a stencil of $21 \times 21$ input points. This requires a halo of width 10 to surround each computational subdomain which must be updated prior to passing to the CNN for inference. In our implementation, we could have made the halo wider for all variables in the model but this would have increased the cost of communication for the whole model which generally has halo widths of three or 4. Instead we made two temporary arrays (one for each of $u$ and $v$) with wide halos of 10 and the cost of updating these halos proved to not be significant.

## 5. Conclusions

We have described an investigation into how well a stochastic-deep learning parameterization of subgrid momentum forcing performs in an idealized ocean model. We set out to explore how to use a pre-defined ML parameterization in a general use, global ocean circulation model written in Fortran. We focused on one particular parameterization, GZ21, that targets the backscatter of energy from unresolved flows. However, the tests, lessons learned, and recommendations apply broadly to any deep learning ocean or atmosphere parameterizations developed (Beucler et al., 2021b; Bolton & Zanna, 2019; Christensen & Zanna, 2022; Krasnopolsky et al., 2010; Maulik et al., 2019; O'Gorman & Dwyer, 2018; Rasp et al., 2018; Yuval et al., 2021).

The ML parameterization was originally trained on a geographic sub-sample of surface flow from a realistic, relatively fine-resolution, fully coupled climate model (CM2.6). We applied the ML parameterization "as is" in a coarse-resolution, idealized wind-driven baroclinic model for which we could afford to run a fine-resolution "truth" simulation. We employed several metrics, that is, KE, spectra, and SSH error, to access the performance. Out of the box, the ML parameterization did improve some aspects of the coarse-resolution solution. However, some artifacts were apparent that were not evident in the original online testing in a barotropic model with flat bottom by Guillaumin and Zanna (2021). Despite these negative aspects, the network produced results that improve some of the model physics without generating infinities or nonsense, so our results are evidence of some underlying robustness of the parameterization. We found the overall energization to be too efficient and that global tuning could be used to yield better results, similarly to Zanna and Bolton (2020). Our results are improved if we tune layer-by-layer, which re-enforces a notion that surface currents and interior currents have different dynamics. Tuning was able to optimize one metric (in our case we used mean KE), and while separate metrics (such as SSH) improved they were not always optimal nor was it obvious they were robustly sensitive to the tuning. The geographic sub-sample used for training seems to have selected sheared flow structures that led to sheared artifacts near boundaries in our tests. This might be a classic example of the "out of sample" problem whereby a network should be trained with enough samples that it is interpolating between "known" states rather than extrapolating beyond. However, as just stated, the network did not "blow up" which is a more common manifestation of "out of sample" problems. The robustness may be connected to the use of the stochastic method with the network, since to exhibit an uncontrolled "blow up" both the network has to be out of sample and the random numbers need to be consistently large (which is statistically unlikely).

We propose that re-training with the surface currents across the whole globe and at different depths, including near boundaries, might eliminate sheared artifacts and potentially address the need for layer-by-layer tuning. The local resolution was not explicitly encoded in the network and we found that the parameterization returned reduced forcing at finer resolutions so did not adversely modify the finer resolution solutions to a major degree. This resolution-dependent behavior suggests that the network is ignoring the absolute values of the input velocities. The network is thus recovering a property of traditional parameterizations that use spatial derivatives. However, tuning at each resolution suggests a weak nonlinear response to the inputs at different resolutions since we had to moderately scale up the parameterization as we refined the resolution. We found the optimal scaling as a function of resolution to be relatively predictable and so suspect that scale-awareness is achievable with this parameterization if the network were trained with multiple resolutions.

The network we used is deep (8 convolutional layers) and thus has a wide stencil (21 × 21) relative to most lateral spatial operators found in a conventional ocean model. This proved to not cause much overhead in our model but is nevertheless a consideration since some infrastructure frameworks may not work so easily or it may even be prohibitive. The wide stencil means that the many near-coast ocean points could feel the choice of how "land values" are handled. Our test results reveal obvious artifacts near the boundary. Improvement is needed in the treatment of coastlines by this parameterization, and we propose that the parameterization would benefit if the network was trained with the global data with more flow regimes including data near coastlines. It is the common practice in ocean models to stagger variables in space. The MOM6 model uses the Arakawa C-grid with flow components normal to the cell used for the continuity budget. The network was trained with co-located variables (B-grid) and so similarly to online tests in Guillaumin and Zanna (2021) we had to interpolate the MOM6 variables to the same point, then interpolate the momentum forcing back. There is a null-space in this approach; structures near the grid-scale that are neither felt by the parameterization nor influenced by the parameterization. We did not investigate the consequences of our interpolation choices but recognize there is potentially wasted resolution below the scales that are affected by the parameterization. The wide stencil and the width of the network (number of channels in the hidden layers) was such that there were 268,005 weights making the number of FLOPs per grid point per time step very large. We were able to offload the network inference to GPUs which made the network affordable. Nevertheless, the wall-clock time spent on the GPUs was still a finite fraction of the wall clock of the model (on CPUs) and so reducing the size of the network will very likely be beneficial. Given the growing propensity of GPUs, and the challenges of porting existing models to GPUs, utilizing GPUs for ML parameterizations seems a viable opportunity (Partee et al., 2022). We tackled the inter-language barrier with a lightweight Fortran module (https://github.com/ylikx/forpy). There are various solutions available for inter-language coupling and using Forpy we found we had to understand various technical aspects of the hardware, for example, CPU and GPU configurations. Turn-key solutions such as SmartSim that handle much of the technical work will likely prove more and more useful in this arena. We encountered a hardware constraint that prohibited us from evaluating the ML parameterization for the full-scale realistic model, OM4 (Adcroft et al., 2019), which requires more processors and GPUs than we had available.

This paper focuses on utilizing a CNN model trained with data from open ocean regions to parameterize the momentum forcing in an idealized double gyre case. Using a simple test case has revealed certain issues with the ML parameterization, such as the need to tune the CNN outputs and the presence of artifacts near wall boundaries. These artifacts near the boundaries could manifest as noise or other artifacts with more complex settings, making diagnosing them challenging, and the task of tuning harder. As a first step, we propose training the same model with global data and re-testing the new model against this idealized case. Until the ML parameterization has been successfully validated in this benchmark it is unlikely to be acceptable in realistic configurations. We speculate that to pass this benchmark the question of how to handle boundary conditions needs to be addressed. Recently, a new line of ML models called neural operators has emerged, which aims to learn mesh-free, infinite-dimensional operators using neural networks. These models, such as the Fourier neural operator (FNO, Li et al., 2020) and Laplace neural operator (LNO, Chen et al., 2023), show potential for problems with complex boundary conditions. While there may be more suitable tools available to handle complex lateral boundaries in oceanic GCMs, we believe that exploring the use of a CNN in our ML parameterization is still helpful since common issues arise, such as the dependence on wide halos or global data. One aspect about the CNN we evaluated is it has an extremely wide stencil that has contributed to its high cost. If a conventional parameterization had such a wide stencil, it would presumably be very sophisticated but also be significantly more expensive than the more common low order parameterizations. Whether the information in the wide stencil is necessary to parameterize the physics is a question for the training, but our results (not unexpectedly) suggest that the impact on performance of the stencil size should be considered in the ML parameterization design.

The neural network is treated as a "black box" in our study; we implicitly trust the parameterization and the pre-calculated quarter million weights. We can make an analogy with the individual weights used in the polynomial expressions for the Gibbs free-energy of sea-water (Feistel, 2008); in this case as well, we implicitly trust the authors to have calculated those weights appropriately when we readily use their weights (and software). When we import a new equation of state, we test the implementation in our model to both evaluate our implementation as well as the new equation of state itself. Here, we conducted such tests with the neural network backscatter

parameterization and made an assessment: the original network performed better than we might have anticipated given that it was trained only on surface data and for limited geographic regions, but there was room for improvement which will need to be assessed in future studies.

Choices made for the network architecture do leave open questions. For instance, the parameterization calculates a momentum forcing written as a body force and not as the divergence of a stress tensor. Model developers often rely on integral constraints or conservation principles to test and evaluate their models but this parameterization conserves neither momentum nor energy. Constraints can be imposed during training as done in Beucler et al. (2021a), Zanna and Bolton (2020), Ross et al. (2023), through a choice of architecture design. In addition, other strategies such as post-processing can achieve similar results (Bolton & Zanna, 2019). Ensuring such conservation can help model developers during the implementation stage, since the properties of the terms in a conventional closed-form parameterization often lend themselves to analysis, which is undeniably harder here unless imposed. Despite no direct imposition of property conservation, we find the network used and revised here to show considerable promise, and the exercise of importing into a conventional model to be manageable. We fully expect to see more widespread use of ML parameterizations in the future.

## Appendix A: Model Equations

We use the model in an adiabatic limit with no buoyancy forcing which simplifies the equations of motion to the stacked shallow water equations. The equations are written in vector-invariant form as

$$\frac{\partial \bar{\mathbf{u}}_k}{\partial t} + \frac{f + \bar{\zeta}_k}{\bar{h}_k}\hat{\mathbf{z}} \times \bar{h}_k \bar{\mathbf{u}}_k + \nabla \bar{K}_k + \nabla \bar{M}_k = \bar{\mathbf{F}}_k + \mathbf{S}_k \tag{A1}$$

$$\frac{\partial \bar{h}_k}{\partial t} + \nabla \cdot \left(\bar{\mathbf{u}} \bar{h}_k\right) = 0 \tag{A2}$$

where the overbar is the horizontal filtering and coarse graining, $\mathbf{u}_k$ is the horizontal component of velocity, $h_k$ is the layer thickness, $f$ is Coriolis parameter, $\zeta_k$ is the vertical component of the relative vorticity, $K_k = (1/2)\mathbf{u}_k \cdot \mathbf{u}_k$ is the KE per unit mass in the horizontal, $\hat{\mathbf{z}}$ is the unit vector pointing in the upward vertical direction, $k$ is the vertical layer index with $k = 1$ at the top and $k = N$ at the bottom, $\nabla$ is the horizontal gradient and $\nabla\cdot$ is the horizontal divergence. $M_k = \sum_{l=1}^{k} g'_{l-1/2}\eta_{l-1/2}$ is the Montgomery potential, where $g'_{k-1/2}$ is the reduced gravity of each layer, $\eta_{k-1/2}$ is the interface position. $\mathbf{F}_k = \frac{1}{\rho_0 h_k}\left(\boldsymbol{\tau}_{k-1/2} - \boldsymbol{\tau}_{k+1/2}\right) - \nabla \cdot \nu_4 \nabla\left(\nabla^2 \mathbf{u}\right)$ represents the accelerations due to the divergence of stresses including the lateral parameterizations that are not inferred from ML-based models. $\mathbf{S}_k$, which is defined in Equation 2, is the subgrid momentum forcing from the ML parameterizations. $\rho_0$ is the reference density, $\boldsymbol{\tau}_{k-1/2}$ is the vertical stress, and $\nabla^2 = \nabla \cdot \nabla$ is the horizontal Laplacian. The turbulence model that we use is a biharmonic friction with a Smagorinsky eddy viscosity following (Griffies & Hallberg, 2000). The eddy viscosity reads

$$\nu_4 = C_S \Delta^4 \sqrt{D_T^2 + D_S^2} \tag{A3}$$

where $D_T = \partial_x u - \partial_y v$ and $D_S = \partial_y u + \partial_x v$ (in Cartesian coordinates) are horizontal tension and shearing strain, respectively, $\Delta = \sqrt{\frac{2(\Delta x)^2(\Delta y)^2}{(\Delta x)^2 + (\Delta y)^2}}$ is a measure of grid spacing.

## Appendix B: Offline Evaluation

In this section, we check the generalization capability of the GZ21 CNN model, which was trained on CM2.6 data, to the double gyre case. To evaluate this, we conducted an offline test using data from the high resolution model R16. This is only an option because we have an affordable idealized experiment and would not be feasible if evaluating in a realistic GCM. The simulation in this model spanned a duration of 10 years, with a total of 121 snapshots taken at 30 days intervals. The procedure to obtain the 'true' forcing and velocity components on the R4 grid is identical to that described in Section 2 for the GZ21 paper and involves filtering and coarse-graining the data from the grid points of R16. Figure B1 illustrates a comparison between the

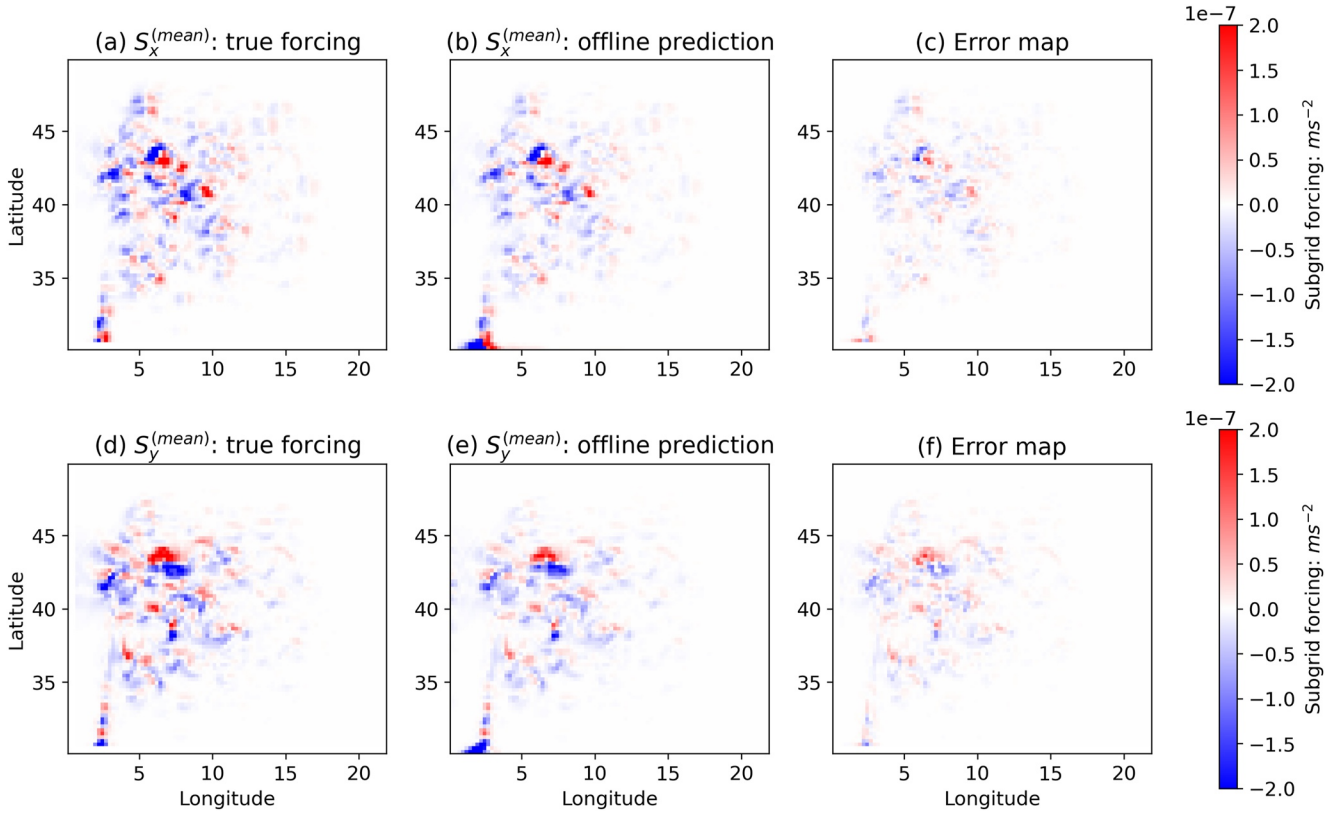


**Figure B1.** Snapshots of the upper layer momentum forcing derived from high resolution model (a and d) and predicted from the GZ21 Convolutional Neural Network model(b and c). The error maps (c and f) are obtained by subtracting the forcing of offline prediction from true forcing.

true forcing and the predicted forcing for a snapshot of the upper layer flow. Overall, the true forcing and the predicted forcing exhibit similar patterns, indicating a reasonable agreement, though there are discrepancies in the amplitude at some locations.

Following the analysis in Figure C1 of GZ21, the time-average $R^2$ coefficient of the forcing at each grid point (as GZ21 defined and used in Figure C1) is shown in Figure B2(a-d) using the formula:

$$R^2_{C,i,j} = 1 - \frac{\sum_{t=1}^{121}\left(S^{(mean)}_{C,i,j,t} - S_{C,i,j,t}\right)^2}{\sum_{t=1}^{121} S^2_{C,i,j,t}};\ C = x, y \tag{B1}$$

where $t$ is the time index of the snapshots. The $R^2$ map for the upper layer demonstrates overall better performance compared to the lower layer, with most regions having coefficient values above 0.8. However, in the quiescent regions where the flow is relatively calm, the coefficient value drops significantly which is consistent with GZ21 in regions of weak forcing. The lower layer exhibiting relatively lower $R^2$ values is consistent with the fact that the CNN model was trained on surface fields. Also replicating the analysis of GZ21 Figures C2 and C3, the time series of the predicted mean forcing is compared to the true forcing, along with 95% confidence intervals derived from the predicted standard deviation, in Figures B2(e) and B2(f), at two locations: one in the active region of the flow ((11°, 40°), white dot), and the other in the quiescent regions ((18°, 40°), green dot). The predicted forcing demonstrates an overall good agreement with the true forcing.

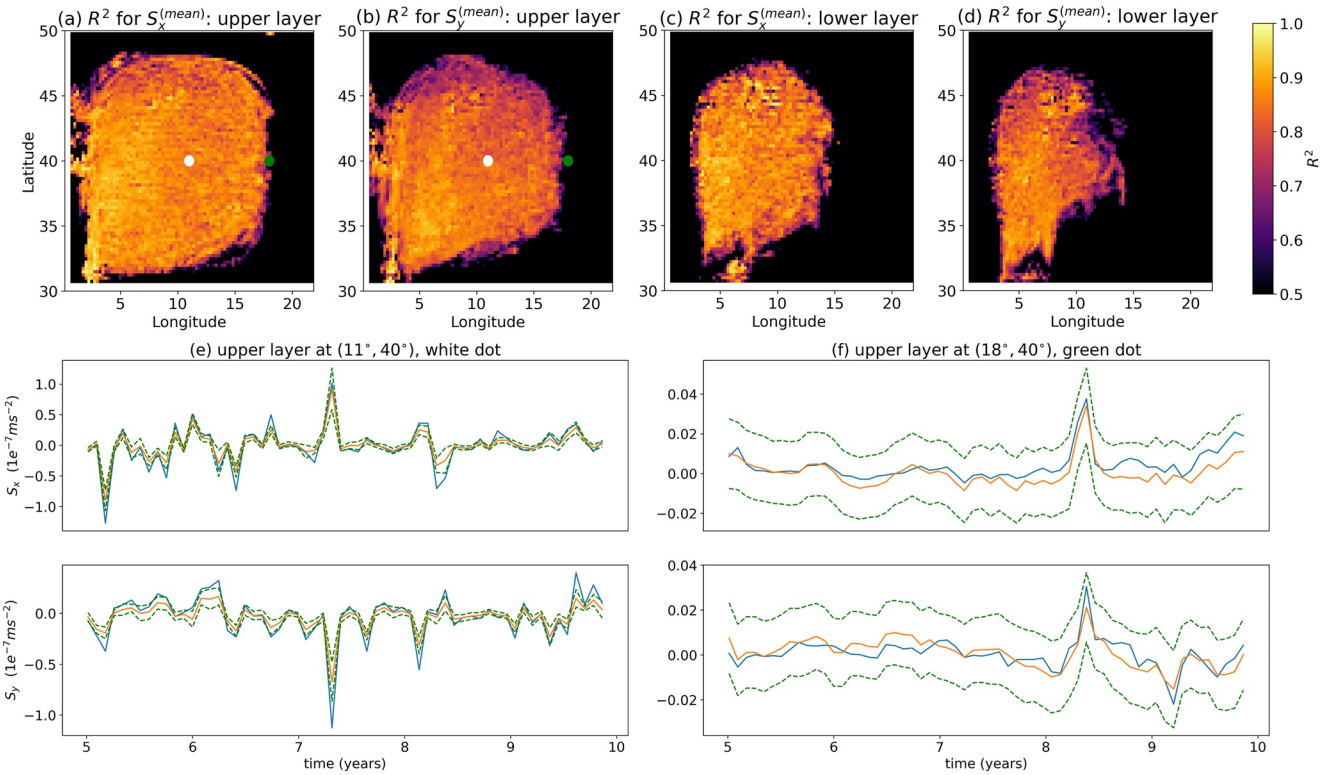


**Figure B2.** Time-average $R^2$ coefficient maps for the forcing at the upper layer flow (a) and (b) and the forcing at the lower layer flow (c) and (d) for offline tests on simulation data from the double gyre case. The white dot represents a location in the active region of the flow, while the green dot represents a location in the quiescent region. Plots (e) and (f) are time series of $R^2$ coefficient at the white dot and green dot, respectively, where the blue lines represent true forcing, the orange lines represent the Convolutional Neural Network predictions, and the dashed green lines represent the 95% confidence interval.

## Data Availability Statement

The source code of the MOM6 version used for implementing the ML parameterization is accessible through Zenodo (Hallberg et al., 2023), while the CNN model files used for the online evaluation in this study (GZ21) can also be accessed via Zenodo (Zhang, 2023b). To facilitate the setup process for the wind-driven double gyre case in the study, we have made the setup files available online (Zhang, 2023a).

**Acknowledgments**
We thank all members of the M²LInES team for helpful discussions and their support throughout this project. We thank Marshall Ward and Wenda Zhang for useful comments on a draft of this manuscript, and Arthur Guillaumin for assistance with the networks. This research received support through the generosity of Eric and Wendy Schmidt by recommendation of the Schmidt Futures program. AA was also supported by award NA18OAR4320123, from the National Oceanic and Atmospheric Administration (NOAA), U.S. Department of Commerce and which funded the Princeton Stellar computer resources used for the inference stage of the research. The statements, findings, conclusions, and recommendations are those of the author(s) and do not necessarily reflect the views of the National Oceanic and Atmospheric Administration, or the U.S. Department of Commerce. CG was supported by a MacCracken Fellowship. CFG was partially supported by NSF DMS award 2009752. This research was also supported in part through the NYU IT High Performance Computing resources, services, and staff expertise.

## References

Adcroft, A., Anderson, W., Balaji, V., Blanton, C., Bushuk, M., Dufour, C. O., et al. (2019). The GFDL global ocean and sea ice model OM4. 0: Model description and simulation features. *Journal of Advances in Modeling Earth Systems*, *11*(10), 3167–3211. https://doi.org/10.1029/2019MS001726

Anstey, J. A., & Zanna, L. (2017). A deformation-based parametrization of ocean mesoscale eddy reynolds stresses. *Ocean Modelling*, *112*, 99–111. https://doi.org/10.1016/j.ocemod.2017.02.004

Balwada, D., Xie, J.-H., Marino, R., & Feraco, F. (2022). Direct observational evidence of an oceanic dual kinetic energy cascade and its seasonality. arXiv preprint arXiv:2202. https://doi.org/10.48550/arXiv.2202.08637.08637

Beck, A., & Kurz, M. (2021). A perspective on machine learning methods in turbulence modeling. *GAMM-Mitteilungen*, *44*(1), e202100002. https://doi.org/10.1002/gamm.202100002

Berner, J., Shutts, G., Leutbecher, M., & Palmer, T. (2009). A spectral stochastic kinetic energy backscatter scheme and its impact on flow-dependent predictability in the ECMWF ensemble prediction system. *Journal of the Atmospheric Sciences*, *66*(3), 603–626. https://doi.org/10.1175/2008jas2677.1

Beucler, T., Pritchard, M., Rasp, S., Ott, J., Baldi, P., & Gentine, P. (2021a). Enforcing analytic constraints in neural networks emulating physical systems. *Physical Review Letters*, *126*(9), 098302. https://doi.org/10.1103/PhysRevLett.126.098302

Beucler, T., Pritchard, M., Yuval, J., Gupta, A., Peng, L., Rasp, S., et al. (2021b). Climate-invariant machine learning. arXiv preprint arXiv:2112.08440. https://doi.org/10.48550/arXiv.2112.08440

Bolton, T., & Zanna, L. (2019). Applications of deep learning to ocean data inference and subgrid parameterization. *Journal of Advances in Modeling Earth Systems*, *11*(1), 376–399. https://doi.org/10.1029/2018MS001472

Brenowitz, N. D., & Bretherton, C. S. (2018). Prognostic validation of a neural network unified physics parameterization. *Geophysical Research Letters*, *45*(12), 6289–6298. https://doi.org/10.1029/2018gl078510

Chen, G., Liu, X., Li, Y., Meng, Q., & Chen, L. (2023). Laplace neural operator for complex geometries. arXiv preprint arXiv:2302.08166.

Christensen, H., & Zanna, L. (2022). Parametrization in weather and climate models. In *Oxford research encyclopedia of climate science*. https://doi.org/10.1093/acrefore/9780190228620.013.826

Curcic, M. (2019). A parallel Fortran framework for neural networks and deep learning. *ACM SIGPLAN Fortran Forum*, *38*(1), 4–21. https://doi.org/10.1145/3323057.3323059

Delman, A., & Lee, T. (2021). Global contributions of mesoscale dynamics to meridional heat transport. *Ocean Science*, *17*(4), 1031–1052. https://doi.org/10.5194/os-17-1031-2021

Espinosa, Z. I., Sheshadri, A., Cain, G. R., Gerber, E. P., & DallaSanta, K. J. (2022). Machine learning gravity wave parameterization generalizes to capture the QBO and response to increased $CO_2$. *Geophysical Research Letters*, *49*(8), e2022GL098174. https://doi.org/10.1029/2022gl098174

Feistel, R. (2008). A Gibbs function for seawater thermodynamics for −6 to 80°C and salinity up to 120 g $kg^{-1}$. *Deep Sea Research Part I: Oceanographic Research Papers*, *55*(12), 1639–1671. https://doi.org/10.1016/j.dsr.2008.07.004

Frederiksen, J. S., & Davies, A. G. (1997). Eddy viscosity and stochastic backscatter parameterizations on the sphere for atmospheric circulation models. *Journal of the Atmospheric Sciences*, *54*(20), 2475–2492. https://doi.org/10.1175/1520-0469(1997)054⟨2475:evasbp⟩2.0.co;2

Gent, P. R., Willebrand, J., McDougall, T. J., & McWilliams, J. C. (1995). Parameterizing eddy-induced Tracer transports in ocean circulation models. *Journal of Physical Oceanography*, *25*(4), 463–474. https://doi.org/10.1175/1520-0485(1995)025⟨0463:PEITTI⟩2.0.CO;2

Gill, A. E., & Adrian, E. (1982). In *Atmosphere-ocean dynamics* (Vol. 30). Academic press.

Greatbatch, R., Zhai, X., Claus, M., Czeschel, L., & Rath, W. (2010). Transport driven by eddy momentum fluxes in the gulf stream extension region. *Geophysical Research Letters*, *37*(24), L24401. https://doi.org/10.1029/2010gl045473

Griffies, S. M., Gnanadesikan, A., Pacanowski, R. C., Larichev, V. D., Dukowicz, J. K., & Smith, R. D. (1998). Isoneutral diffusion in a z-coordinate ocean model. *Journal of Physical Oceanography*, *28*(5), 805–830. https://doi.org/10.1175/1520-0485(1998)028⟨0805:IDIAZC⟩2.0.CO;2

Griffies, S. M., & Hallberg, R. W. (2000). Biharmonic friction with a Smagorinsky-like viscosity for use in large-scale eddy-permitting ocean models. *Monthly Weather Review*, *128*(8), 2935–2946. https://doi.org/10.1175/1520-0493(2000)128⟨2935:bfwasl⟩2.0.co;2

Griffies, S. M., Winton, M., Anderson, W. G., Benson, R., Delworth, T. L., Dufour, C. O., et al. (2015). Impacts on ocean heat from transient mesoscale eddies in a hierarchy of climate models. *Journal of Climate*, *28*(3), 952–977. https://doi.org/10.1175/jcli-d-14-00353.1

Guillaumin, A., & Zanna, L. (2021). Stochastic deep learning parameterization of ocean momentum forcing. *Journal of Advances in Modeling Earth Systems*, *13*(9), e2021MS002534. https://doi.org/10.1002/essoar.10506419.1

Hallberg, R. (2013). Using a resolution function to regulate parameterizations of oceanic mesoscale eddy effects. *Ocean Modelling*, *72*, 92–103. https://doi.org/10.1016/j.ocemod.2013.08.007

Hallberg, R., Adcroft, A., Marques, G., Ward, M., Hedstrom, K., Shao, A., et al. (2023). chzhangudel/mom6: Initial release (forpy2/2023.02.22) [software]. Zenodo. https://doi.org/10.5281/zenodo.7663075

Hallberg, R., & Rhines, P. B. (2000). Boundary sources of potential vorticity in geophysical circulations. In R. M. Kerr & Y. Kimura (Eds.), *IUTAM symposium on developments in geophysical turbulence* (pp. 51–65). Springer Netherlands. https://doi.org/10.1007/978-94-010-0928-7_5

He, K., & Sun, J. (2014). Convolutional neural networks at constrained time cost. Retrieved from https://arxiv.org/abs/1412.1710

Hewitt, H. T., Roberts, M., Mathiot, P., Biastoch, A., Blockley, E., Chassignet, E. P., et al. (2020). Resolving and parameterising the ocean mesoscale in Earth system models. *Current Climate Change Reports*, *6*(4), 137–152. https://doi.org/10.5281/zenodo.3685918

Jansen, M. F., & Held, I. M. (2014). Parameterizing subgrid-scale eddy effects using energetically consistent backscatter. *Ocean Modelling*, *80*, 36–48. https://doi.org/10.1016/j.ocemod.2014.06.002

Juricke, S., Danilov, S., Koldunov, N., Oliver, M., & Sidorenko, D. (2020). Ocean kinetic energy backscatter parametrization on unstructured grids: Impact on global eddy-permitting simulations. *Journal of Advances in Modeling Earth Systems*, *12*(1), e2019MS001855. https://doi.org/10.1029/2019ms001855

Juricke, S., Palmer, T. N., & Zanna, L. (2017). Stochastic subgrid-scale ocean mixing: Impacts on low-frequency variability. *Journal of Climate*, *30*(13), 4997–5019. https://doi.org/10.1175/jcli-d-16-0539.1

Kjellsson, J., & Zanna, L. (2017). The impact of horizontal resolution on energy transfers in global ocean models. *Fluid*, *2*(3), 45. https://doi.org/10.3390/fluids2030045

Krasnopolsky, V., Fox-Rabinovitz, M., Hou, Y., Lord, S., & Belochitski, A. (2010). Accurate and fast neural network emulations of model radiation for the NCEP coupled climate forecast system: Climate simulations and seasonal predictions. *Monthly Weather Review*, *138*(5), 1822–1842. https://doi.org/10.1175/2009mwr3149.1

Legg, S., Briegleb, B., Chang, Y., Chassignet, E. P., Danabasoglu, G., Ezer, T., et al. (2009). Improving oceanic overflow representation in climate models: The gravity current entrainment climate process team. *Bulletin of the American Meteorological Society*, *90*(5), 657–670. https://doi.org/10.1175/2008BAMS2667.1

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., & Anandkumar, A. (2020). Fourier neural operator for parametric partial differential equations. arXiv preprint arXiv:2010.08895.

Liu, C., Zhang, H., Cheng, Z., Shen, J., Zhao, J., Wang, Y., et al. (2021). Emulation of an atmospheric gas-phase chemistry solver through deep learning: Case study of chinese mainland. *Atmospheric Pollution Research*, *12*(6), 101079. https://doi.org/10.1016/j.apr.2021.101079

MacKinnon, J. A., Zhao, Z., Whalen, C. B., Waterhouse, A. F., Trossman, D. S., Sun, O. M., et al. (2017). Climate process team on internal wave–driven ocean mixing. *Bulletin of the American Meteorological Society*, *98*(11), 2429–2454. https://doi.org/10.1175/BAMS-D-16-0030.1

Maulik, R., San, O., Rasheed, A., & Vedula, P. (2019). Subgrid modelling for two-dimensional turbulence using neural networks. *Journal of Fluid Mechanics*, *858*, 122–144. https://doi.org/10.1017/jfm.2018.770

O'Gorman, P. A., & Dwyer, J. G. (2018). Using machine learning to parameterize moist convection: Potential for modeling of climate, climate change, and extreme events. *Journal of Advances in Modeling Earth Systems*, *10*(10), 2548–2563. https://doi.org/10.1029/2018ms001351

Ott, J., Pritchard, M., Best, N., Linstead, E., Curcic, M., & Baldi, P. (2020). A Fortran-Keras deep learning bridge for scientific computing. *Scientific Programming*, *2020*, 1–13. https://doi.org/10.1155/2020/8888811

Partee, S., Ellis, M., Rigazzi, A., Shao, A. E., Bachman, S., Marques, G., & Robbins, B. (2022). Using machine learning at scale in numerical simulations with SmartSim: An application to ocean climate modeling. *Journal of Computational Science*, *62*, 101707. https://doi.org/10.1016/j.jocs.2022.101707

Rasp, S., Pritchard, M. S., & Gentine, P. (2018). Deep learning to represent subgrid processes in climate models. *Proceedings of the National Academy of Sciences of the United States of America*, *115*(39), 9684–9689. https://doi.org/10.1073/pnas.1810286115

Ross, A. S., Li, Z., Perezhogin, P., Fernandez-Granda, C., & Zanna, L. (2023). Benchmarking of machine learning ocean subgrid parameterizations in an idealized model. *Journal of Advances in Modeling Earth Systems*, *15*(1), e2022MS003258. https://doi.org/10.1029/2022MS003258

Sane, A., Reichl, B. G., Adcroft, A., & Zanna, L. (2023). Parameterizing vertical mixing coefficients in the ocean surface boundary layer using neural networks. https://doi.org/10.48550/arXiv.2306.09045

Scott, R. B., & Arbic, B. K. (2007). Spectral energy fluxes in geostrophic turbulence: Implications for ocean energetics. *Journal of Physical Oceanography*, *37*(3), 673–688. https://doi.org/10.1175/jpo3027.1

Thuburn, J., Kent, J., & Wood, N. (2014). Cascades, backscatter and conservation in numerical models of two-dimensional turbulence. *Quarterly Journal of the Royal Meteorological Society*, *140*(679), 626–638. https://doi.org/10.1002/qj.2166

Yuval, J., O'Gorman, P. A., & Hill, C. N. (2021). Use of neural networks for stable, accurate and physically consistent parameterization of subgrid atmospheric processes with good performance at reduced precision. *Geophysical Research Letters*, *48*(6), e2020GL091363. https://doi.org/10.1029/2020gl091363

Zanna, L., & Bolton, T. (2020). Data-driven equation discovery of ocean mesoscale closures. *Geophysical Research Letters*, *47*(17), e2020GL088376. https://doi.org/10.1002/essoar.10503535.1

Zanna, L., Brankart, J. M., Huber, M., Leroux, S., Penduff, T., & Williams, P. D. (2018). Uncertainty and scale interactions in ocean ensembles: From seasonal forecasts to multidecadal climate predictions. *Quarterly Journal of the Royal Meteorological Society*, *0*(0), 160–175. https://doi.org/10.1002/qj.3397

Zanna, L., Mana, P. P., Anstey, J., David, T., & Bolton, T. (2017). Scale-aware deterministic and stochastic parametrizations of eddy-mean flow interaction. *Ocean Modelling*, *111*, 66–80. https://doi.org/10.1016/j.ocemod.2017.01.004

Zhang, C. (2023a). chzhangudel/double_gyre: Initial release (v1.0.1) [software]. Zenodo. https://doi.org/10.5281/zenodo.7663128

Zhang, C. (2023b). chzhangudel/forpy_cnn_gz21: Initial release (v1.0.0) [software]. Zenodo. https://doi.org/10.5281/zenodo.7663062