Stochastic Deep Learning parameterization of Ocean Momentum Forcing

Arthur Guillaumin[1] and Laure Zanna[1]

[1]New York University

November 22, 2022

Abstract

Coupled climate simulations that span several hundred years cannot be run at a high-enough spatial resolution to resolve mesoscale ocean dynamics. These mesoscale dynamics backscatter to macroscales. Recently, several studies have considered Deep Learning to parameterize subgrid forcing within macroscale ocean equations using data from idealized simulations. In this manuscript, we present a stochastic Deep Learning parameterization that is trained on data generated by CM2.6, a high-resolution state-of-the-art coupled climate model with nominal resolution 1/10° . We train a Convolutional Neural Network for the subgrid momentum forcing using macroscale surface velocities from a few selected subdomains. At each location and each time step of the coarse grid, rather than predicting a single number, we predict the mean and standard deviation of a Gaussian probability distribution. This approach requires training our neural network to minimize a negative log-likelihood loss function rather than the Mean Square Error, which has been the standard in applications of Deep Learning to the problem of parameterizations. Each prediction of the mean subgrid forcing can be associated with an uncertainty estimate and can form the basis for a stochastic subgrid parameterization. Offline tests show that our parameterization generalizes well to the global oceans, and a climate with increased CO_2 levels, without further training. We test our stochastic parameterization in an idealized shallow water model. The implementation is stable and improves some statistics of the flow. Our work demonstrates the potential of combining Deep Learning tools with a probabilistic approach in parameterizing unresolved ocean dynamics.

Stochastic-Deep Learning parameterization of Ocean Momentum Forcing

Arthur P. Guillaumin[1], **Laure Zanna**[1]

[1]Courant Institute of Mathematical Sciences, New York University, New York, NY 10012, USA

Key Points:

- We use data from a realistic high-resolution coupled climate model to train a neural network
- We learn a stochastic parameterization of subgrid momentum forcing with the neural network
- The parameterization generalizes well and results in a stable implementation

Corresponding author: Arthur P. Guillaumin, ag7531@nyu.edu

Abstract

Coupled climate simulations that span several hundred years cannot be run at a high-enough spatial resolution to resolve mesoscale ocean dynamics. These mesoscale dynamics backscatter to macroscales. Recently, several studies have considered Deep Learning to parameterize subgrid forcing within macroscale ocean equations using data from idealized simulations. In this manuscript, we present a stochastic Deep Learning parameterization that is trained on data generated by CM2.6, a high-resolution state-of-the-art coupled climate model with nominal resolution $1/10°$. We train a Convolutional Neural Network for the subgrid momentum forcing using macroscale surface velocities from a few selected subdomains. At each location and each time step of the coarse grid, rather than predicting a single number, we predict the mean and standard deviation of a Gaussian probability distribution. This approach requires training our neural network to minimize a negative log-likelihood loss function rather than the Mean Square Error, which has been the standard in applications of Deep Learning to the problem of parameterizations. Each prediction of the mean subgrid forcing can be associated with an uncertainty estimate and can form the basis for a stochastic subgrid parameterization. Offline tests show that our parameterization generalizes well to the global oceans, and a climate with increased CO_2 levels, without further training. We test our stochastic parameterization in an idealized shallow water model. The implementation is stable and improves some statistics of the flow. Our work demonstrates the potential of combining Deep Learning tools with a probabilistic approach in parameterizing unresolved ocean dynamics.

Plain Language Summary

Numerical predictions for the next century are pivotal to understanding the impact of climate change. However, those predictions are limited in accuracy by the trade-off between the models' spatio-temporal resolution and their time span, due to the large computational power involved. Since small-scale dynamics impact larger-scale dynamics, a common approach is to use idealized equations, based on the practitioner's understanding of physics, to account for the impact of unresolved small-scale dynamics on the large-scale flow. However, this approach has shown its limits. Recently, several studies have considered the use of Deep Learning — a set of techniques designed to learn high-dimensional complex functions from large amounts of data — to learn the impact of small-scale dynamics on the large-scale flow. Here we apply Deep Learning methods using simulated data from a state-of-the-art climate model. Additionally, we account for the uncertainty associated with the learned representation of the impact of the small scales on the large scale. Our tests using this representation in a simple ocean model show that some metrics are improved. Much work remains to be done to assess the success of Deep Learning in improving climate models.

1 Introduction

The climate system is governed by highly non-linear equations, making them inherently multiscale, with small-scale processes backscattering to large scales. Fluid dynamics equations are known and valid in a continuum. However, climate models solve fluid dynamics equations on a grid, resulting in approximate solutions. Ideally, increasing the spatio-temporal resolution could improve these truncated simulations. However, even with the increasing available computational power, running climate models over decades or centuries is not a viable approach within the near future (Balaji, 2021). Typically, the impact of unresolved small-scale processes on coarse quantities is accounted for via parameterizations. These parameterizations are commonly based on first principles (Gent & McWilliams, 1989), and despite vastly improving the physics and the simulations, they continue to induce biases in simulations, e.g. IPCC (2013).

The era of Machine Learning offers an opportunity to improve the parameterization of unresolved processes using available data from observations and limited high-resolution simulations. While some progress has been made towards *online* learning of unresolved processes in partial differential equations (Sirignano et al., 2020), the approach is not yet ready for complex climate simulations and might not be generalizable due to model dependence. Therefore, the typical approach in atmosphere and ocean modeling consists in training Machine Learning algorithms offline, with a subgrid forcing term that is diagnosed via a filtering operation over high-resolution simulation data. Some recent studies have shown the potential of Machine Learning approaches for atmospheric (Rasp et al., 2018; Yuval & O'Gorman, 2020) and ocean parameterizations (Bolton & Zanna, 2019; Zanna & Bolton, 2020) to improve simulations. So far, most studies on ocean parameterizations that use Machine Learning have been limited to the use of data from idealized models. The viability of deep learning parameterizations using data from realistic coupled or uncoupled models and their potential to generalize to different climates remain open questions. The stability and the physical behavior of the implementation of Deep Learning parameterizations in models have also been a subject of debate (Yuval & O'Gorman, 2020; Brenowitz et al., 2020).

Here we address these questions by showing the high performance of a Deep Neural Network in offline predictions of subgrid momentum forcing in different climates using data from a high-resolution coupled climate model, which resolves ocean mesoscale eddies in many regions (Hallberg, 2013; Griffies et al., 2015). Our work focuses on parameterizing the interaction between mesoscale eddies and large-scale flow, which is key to establishing the transfer of energy between reservoir and scales (Ferrari & Wunsch, 2009) and to establishing the large-scale ocean circulation (Waterman & Jayne, 2011). In particular, we propose a stochastic parameterization that aims to represent the inherent uncertainty of the subgrid forcing, stabilize the online implementation of the parameterization (Zanna et al., 2017; Palmer, 2012) and reduce systematic biases (Berner et al., 2017; Gagne II et al., 2020). Stochastic parameterizations become especially needed in what has been called the *gray zone* (Gerard, 2007; Jones et al., 2019), where subgrid processes are partly resolved such that laws of large numbers do not apply (Berner et al., 2017). In our study, our neural network model outputs the mean and standard deviation for the predicted momentum forcing, which forms the basis of a stochastic parameterization that we will implement in an idealized ocean model. Our contribution therefore establishes a bridge between recent developments on Deep Learning approaches to the problem of parameterizations and stochastic approaches (Mason & Thomson, 1992; Zanna et al., 2017, 2018).

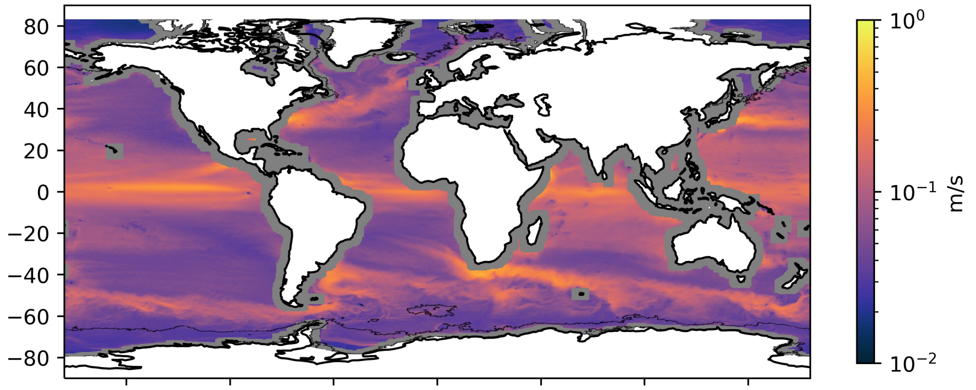
The manuscript is structured as follows. In Section 2, we describe the data, the neural network architecture and the training procedure — which uses a probabilistic loss function. In Section 3, we conduct an offline test on a global scale, showing the ability of our neural network to generalize to regions not seen during training. We also show the ability of our neural network to generalize to a different climate in which CO_2 levels are higher and have affected the mesoscale variability. In Section 4, we demonstrate the potential for increased stability via a stable implementation of our stochastic parameterization into an idealized ocean model. Finally, in section 5 we conclude and discuss the implications of our work and future directions.

2 Methods

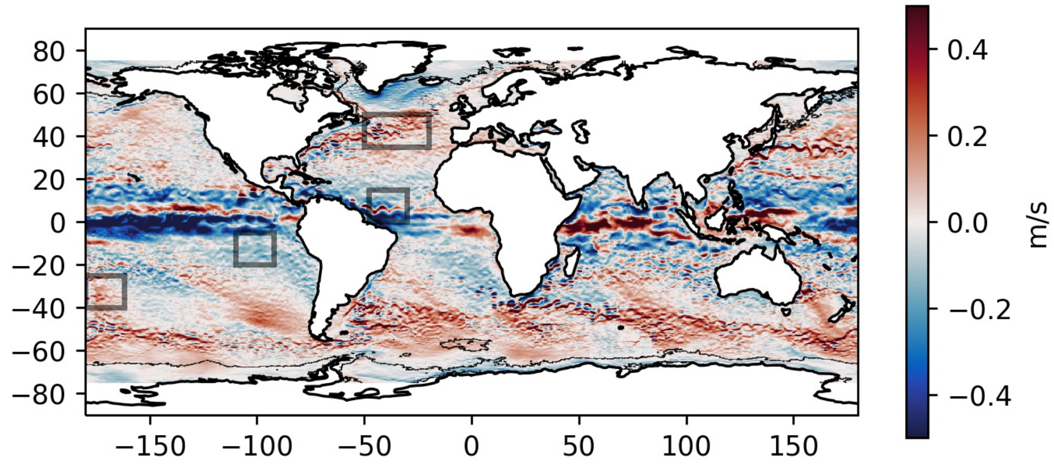
In Sections 2.1 and 2.2, we describe the filtering and subsequent coarsening of the data in order to diagnose the corresponding subgrid momentum forcing necessary to force a coarse-resolution model. In Section 2.3 and 2.4, we describe a procedure that enables us to represent the uncertainty associated with the forcing using a probabilistic loss function for training. In Section 2.5 we review the structure of our proposed neural network. Finally, in Section 2.6 we provide details about our training procedure.

2.1 Data for Training and Validation

Applications of Deep Learning to the parameterization of subgrid ocean momentum forcing have been limited to very idealized models of the ocean dynamics so far (Bolton & Zanna, 2019; Zanna & Bolton, 2020). In contrast, here we investigate the use of Deep Learning using data from a state-of-the-art high-resolution coupled climate model, CM2.6 (Delworth et al., 2012; Griffies et al., 2015). The nominal horizontal resolution of the ocean component of CM2.6 is 1/10°, therefore resolving mesoscale eddies in many regions of the ocean (Hallberg, 2013). The data and tools for analysis were obtained from the Pangeo platform (Abernathey et al., 2021).



(a) Standard deviation of $\overline{\mathbf{u}}$.



(b) Snapshot of zonal velocity

Figure 1: Filtered and coarse-grained surface velocity $\overline{\mathbf{u}}$ in $[m/s]$ from piControl used as training data: (a) standard deviation of surface velocity norm and (b) snapshot of the zonal component. The grey rectangle identify the training subdomains used in this study.

The data used in the present work consists of the high-resolution simulated ocean surface velocity field $\mathbf{u}$ with components u (zonal) and v (meridional). The model grid is configured according to an Arakawa B-grid (Griffies, 2015), with velocity points (both zonal and meridional) placed to the North-East of tracer (T) points, i.e. the top-right corner of a T-cell. The temporal resolution of the surface velocity data is daily, and the available data span over approximately 7000 days (about 20 years) for each of the two available simulations — a control simulation with pre-industrial atmospheric CO_2 levels, referred to as piControl, and a forced simulation with a 1% CO_2 increase per year, referred to as 1ptCO2 (Griffies et al., 2015). The 1ptCO2 simulation experiences a one percent increase of CO_2 per year from the levels of the control simulation until it reaches doubling after 70 years, at which point the CO_2 levels remain constant. The 1ptCO2 simulation data available from Pangeo corresponds to years 60-80.

2.2 Filtering and Coarse-Graining Procedure

In this section, we describe the processing necessary to generate the training data for our neural network. The procedure follows the two steps presented in Zanna and Bolton (2020): low-pass area-weighted Gaussian filtering, followed by coarse-graining. Based on the high-resolution surface velocities $\mathbf{u}$ from the CM2.6 simulations, this procedure generates coarse-resolution velocity data that mimics the simulation data from coarser models that will serve as the input to our neural network. In addition, given the high-resolution and coarsened velocity data, we diagnose the missing subgrid forcing of a coarse-resolution model (e.g., CM2.5) compared to its high-resolution counterpart (here, CM2.6). This missing forcing is the subgrid parameterization needed at coarse resolution to mimic the effect of unresolved scales on the large-scale flow that will be learned by the neural network.

Unlike data used in previous machine learning studies (Bolton & Zanna, 2019; Zanna & Bolton, 2020; Yuval & O'Gorman, 2020), the CM2.6 grid is on a sphere. In the zonal direction, the spacing is uniform at $1/10°$ longitude spacing, but in the meridional direction, the grid spacing is not uniform. The grids of CM2.5, with $1/4°$ nominal resolution, and CM2.1, with $1°$ nominal resolution, have a similar structure. Therefore, the meridional length scale used to define the subgrid eddy forcing should depend on the latitude. In contrast with the typical approach, rather than selecting a uniform length scale to filter the data and generate a coarse-resolution field, we select a uniform and unitless integer *scaling factor* σ, that defines the number of grid boxes from the high-resolution grid that map to a single grid box of the low-resolution grid. This is the simplest and most consistent definition of subgrid scale for the purpose of data-driven parameterization. This unitless scaling factor applies to both the filtering and coarse-graining steps.

We now describe in details the two steps of our low-resolution data generation procedure given the fixed scaling factor σ. As a first step we apply a low-pass weighted Gaussian filter, denoted by $\overline{(\;)}$, to the high-resolution surface velocity data, with weights provided by grid box areas, to separate the subgrid from the resolved field (Bolton & Zanna, 2019). The standard deviation of the Gaussian kernel is set to $\sigma/2$, such that approximately 80% of its weight falls within the interval $[-\sigma/2, \sigma/2]$ of length σ. Note that in using a uniform scaling factor we also allow the use of standard convolution algorithms for regularly-spaced data. This would not be possible if we were using a uniform length scale as we would then have to adapt the size of the filter in terms of number of grid points as a function of latitude, incurring a high computational cost to generate the training data. The second step simply consists of a coarse-graining procedure. We down-sample the data by a factor of σ along each axis, where the down-sampling is based on the mean function applied over squares of side length σ – equivalent to area-weighted average. After coarse-graining, the resulting grid consists of approximately σ^2 times less points than the high-resolution grid.

This filtering and coarse-graining procedure is applied to the surface velocity from CM2.6 control simulation. Figure 1b shows a snapshot of the filtered and coarse-grained surface zonal velocity. The subgrid momentum forcing on the high-resolution grid, denoted $\mathbf{S} = [S_X,\ S_Y]^T$, is diagnosed via,

$$\mathbf{S} = (\overline{\mathbf{u}} \cdot \nabla)\,\overline{\mathbf{u}} - \overline{(\mathbf{u} \cdot \nabla)\,\mathbf{u}}, \tag{1}$$

and is then coarse-grained. For exact implementation details of the entire procedure in the form of pseudo-code, please see Appendix B or the online code.

In our work, we primarily target parameterization for the eddy-permitting regime, in which momentum parameterizations are in demand to mimic the inverse energy cascade and backscatter processes (Treguier et al., 1997; Zanna et al., 2017; Bachman, 2019; Jansen et al., 2019; Zanna et al., 2020). Here, we present experiments in which we set $\sigma = 4$, such that, irrespective of the subdomains of study, the coarse-grained grid has approximately 4 times less grid boxes along each horizontal dimension. This choice of 4 grid boxes leads to a subgrid forcing of an ocean model at resolution 0.4°, close to the resolution of ESM4 which has a nominal resolution of 0.5° (Dunne et al., 2020).

The filtering and coarse-graining procedure is applied to both data from the piControl and 1pctCO2 simulations. The piControl dataset will be used both for training and offline testing, while the 1ptCO2 dataset will be used for testing only.

2.3 Prediction: Conditional Distribution of Subgrid Scale Forcing

Our goal is to learn a parameterization, denoted by $\hat{\mathbf{S}}$, of the diagnosed *true* subgrid momentum forcing, $\mathbf{S}$ (eqn. 1), using deep learning. We propose a neural network that uses maps of coarse surface velocities, $\overline{\mathbf{u}}$, at a given time as inputs, and estimates the subgrid momentum forcing components at that same time as outputs. Here *estimates* is to be understood in a broad sense: it could be a single-value prediction or a probability distribution as we now explain.

Specifically, in this work we present a stochastic parameterization of the subgrid momentum forcing. To do so, we assume that at each grid box, the distribution of the forcing is Gaussian, conditionally on the coarse surface velocities (we do not assume that the marginal distribution of the forcing is Gaussian). We also assume that the forcing at distinct grid boxes and times are conditionally independent given the coarse surface velocities.

The rationale behind stochastic approaches to the modeling of the subgrid-scale forcing is the following: firstly they can partly account for the uncertainty in the representation for the subgrid forcing (Brankart, 2013; Berner et al., 2017; Zanna et al., 2018; Juricke & Zanna, 2017; Williams et al., 2016; Stanley et al., 2020) ; secondly, they have proven potential in stabilizing numerical simulations (Palmer, 2012; Zanna et al., 2017; Berner et al., 2017).

One of the main sources of uncertainty in the predicted subgrid forcing comes from the fact that we only use the resolved coarse velocities to make a prediction. Given a surface velocity field over a subdomain of the oceans at a given time, we do not necessarily expect the subgrid momentum forcing to be given by a deterministic mapping. To illustrate this statement, consider Equation 1; for the problem at hand, $\mathbf{u}$ is unknown, and while $\mathbf{u} \mapsto \overline{\mathbf{u}}$ is well-defined as a mapping, it is not invertible. Thus the parameterization problem can be viewed as an inverse problem, for which probabilistic representations are a common approach (Bishop, 1991). If $\mathbf{u}$ is seen as a random variable, we may want to represent the probability distribution $P(\mathbf{u}|\overline{\mathbf{u}})$, and the same applies to the forcing. Hence we may want our neural network's output to determine a parametric probability distribution rather than a single number.

Besides, a stochastic parameterization of the subgrid forcing can also account for the fact that what we call the true subgrid forcing, Equation 1, depends on our choice of filter which may not adequately represent the "missing forcing" from any given numerical model at coarse-resolution. One could train our neural network with subgrid forcing generated from a variety of methods, to *partially* account for the fact that the exact subgrid forcing is not known.

The output Gaussian distribution at each location can be interpreted as an estimate of the conditional distribution of the subgrid momentum forcing given the local velocity field. Its mean represents the expectation of that conditional probability distribution, while its standard deviation represents the uncertainty around the mean. Such representation will allow deriving confidence intervals of the predicted subgrid momentum forcing (see Section 3.2). It also forms the basis of our stochastic parameterization, see Section 4 about implementation. In the next section, we will show how to learn the mean and standard deviation of the subgrid forcing from data.

2.4 Probabilistic Loss Function: From MSE to Gaussian Log-Likelihood

To train our neural network, we aim to find a local minimum to a loss function $L(\mathbf{S}, \hat{\mathbf{S}}(\boldsymbol{\theta}))$ — summed over all the samples of the training dataset — that represents the *mismatch* between our prediction $\hat{\mathbf{S}}$ and the true value $\mathbf{S}$ given the current state of the parameters of the neural network, represented here by the vector of parameters $\boldsymbol{\theta}$. Here, $\mathbf{S}$, the target tensors of the neural network corresponding to a single sample at a given time, has dimensions (n_C, n_x, n_y), where $n_C = 2$ for the zonal and meridional component of the velocity field, and $(n_x,\, n_y)$ is the size of the domain considered, i.e., the number of grid boxes in the zonal and meridional direction, respectively. We have ignored the number of mini-batches here for simplicity, which will be discussed in Section 2.6.

The most common loss function used in regression is the Mean Square Error (MSE), which in our case would take the form of, for a single sample,

$$L_{\mathrm{MSE}}(\mathbf{S}, \hat{\mathbf{S}}(\boldsymbol{\theta})) = \sum_{k=1}^{n_C} \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} (\hat{\mathbf{S}}_{k,i,j} - \mathbf{S}_{k,i,j})^2, \tag{2}$$

where k denotes the index of the component of the subgrid momentum forcing (here, 1 corresponds to the zonal component and 2 to the meridional component). Despite its widespread use within the Deep Learning community for regression, the MSE loss function is not always appropriate. To justify the above claim briefly, it is common to interpret the MSE loss function from a probabilistic perspective. For simplicity, we limit the discussion to univariate random variables, but this can be easily extended to multivariate variables. Let ψ, ξ be random variables; assume that ψ is observed, and ξ is such that its conditional probability density function given ψ is a Gaussian distribution with mean μ and constant standard deviation σ,

$$p(\xi|\psi; \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(\xi - \mu(\psi))^2}{2\sigma^2}\right\}. \tag{3}$$

If we assume that $\mu(\psi)$ can be modeled by a parametric function f (in our case the neural network) with parameter $\boldsymbol{\theta}$, the log-likelihood of the parameters $\boldsymbol{\theta}, \sigma$ for an independent and identically distributed (i.i.d.) sample $\{\psi_i, \xi_i\}_{i=1,\dots,n}$ will be given by,

$$l(\boldsymbol{\theta}, \sigma) = \sum_{i=1}^{n} \left\{-\frac{1}{2}\log 2\pi\sigma^2 - \frac{(\xi_i - f(\psi_i, \boldsymbol{\theta}))^2}{2\sigma^2}\right\}. \tag{4}$$

Maximizing this log-likelihood[1] over $\boldsymbol{\theta}, \sigma$ can be achieved in a separable way (Davison, 2003): we first maximize over $\boldsymbol{\theta}$, which corresponds to training the neural network using the MSE loss, and we then estimate σ by simply computing the standard deviation of the residuals $\{f(\psi_i, \boldsymbol{\theta}) - \xi_i\}_{i=1,\dots,n}$. Hence, from a probabilistic point of view, by minimizing the MSE loss function, we are assuming a *constant* standard deviation (i.e. that does not depend on the velocity field).

[1] Maximizing the log-likelihood results in estimating the parameters of a probability distribution, so that under the assumed statistical model f the observed data ψ is most probable.

In this paper, we propose to relax this common assumption based on the literature and our understanding of the data. We replace the MSE loss function by a full negative Gaussian log-likelihood. Referring back to our univariate example, this would lead to replacing Equation 4 by,

$$l(\boldsymbol{\theta}) = \sum_{i=1}^{n} \left\{ -\frac{1}{2} \log 2\pi f_2(\psi_i, \boldsymbol{\theta})^2 - \frac{(\xi_i - f_1(\psi_i, \boldsymbol{\theta}))^2}{2 f_2(\psi_i, \boldsymbol{\theta}))^2} \right\}, \tag{5}$$

where our function f —which would correspond to our neural network — now has two components, one for the mean, f_1, of the Gaussian distribution, and the other one for the standard deviation, f_2. In particular, the term corresponding to the standard deviation of the Gaussian in Equation 5, $f_2(\psi_i, \boldsymbol{\theta}))$, does depend on the input ψ_i. In order to apply this to the problem of subgrid momentum forcing, we build our neural network to output the two moments of a Gaussian distribution, at each location and for both (zonal and meridional) components of the subgrid momentum forcing. The output tensor, $\hat{\mathbf{S}}$, now has dimension $(2{\times}n_C, n_x, n_y)$: we have four output channels $(2\ \times n_C)$— the first two correspond to the means of the two components of the subgrid momentum forcing, the last 2 correspond to the associated standard deviations. Our loss function therefore takes the form of (ignoring constant terms),

$$L_{\mathrm{G}}(\mathbf{S}, \hat{\mathbf{S}}(\boldsymbol{\theta})) = \sum_{k=1}^{n_C} \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} \left\{ \log \hat{\mathbf{S}}_{k+2,i,j} + \frac{(\hat{\mathbf{S}}_{k,i,j} - \mathbf{S}_{k,i,j})^2}{2\hat{\mathbf{S}}^2_{k+2,i,j}} \right\}, \tag{6}$$

For ease of reading, we introduce a more natural notation, where we denote $\mathbf{S}_{C,i,j}$ the true value of the forcing, $\hat{\mathbf{S}}^{(\mathrm{mean})}_{C,i,j}$ the mean of the predicted gaussian distribution, and $\hat{\mathbf{S}}^{(\mathrm{std})}_{C,i,j}$ its standard deviation, for component $C = X(\text{zonal}), Y(\text{meridional})$ at location i, j. With this notation, Equation 6 takes the form of,

$$L_{\mathrm{G}}(\mathbf{S}, \hat{\mathbf{S}}(\boldsymbol{\theta})) = \sum_{C=X,Y} \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} \left\{ \log \hat{\mathbf{S}}^{(\mathrm{std})}_{C,i,j} + \frac{(\hat{\mathbf{S}}^{(\mathrm{mean})}_{C,i,j} - \mathbf{S}_{C,i,j})^2}{2\left(\hat{\mathbf{S}}^{(\mathrm{std})}_{C,i,j}\right)^2} \right\}. \tag{7}$$

The neural network will learn to jointly optimize the two moments of the predicted Gaussian distribution, as we show in the schematic of Figure 2. Note that we also jointly train on both zonal and meridional components of the forcing, rather than having separate neural networks for each component, as in (Zanna & Bolton, 2020).

2.5 Neural Network Architecture

Our neural network is a Fully Convolutional Neural Network (Long et al., 2015) with a sequence of eight convolutional layers. The ReLU activation function is used for hidden layers. Given that the neural network is fully convolutional, it can adapt to varying sizes of the input subdomain. We remind the reader that the input consists of two channels, one per component of the velocity field, while the output consists of four channels, two for each of the two components of the subgrid momentum forcing, see Section 2.4. We do not use any padding in the implementation of our convolutional layers. Due to the lack of padding in our neural network structure, some *pixels* near the edges are lost in the application of convolutional layers. This results in the outputs predicted by our neural network having spatial extent $(n_x - p, n_y - p)$, where p is a non-negative integer that depends on the size of the kernels used in the convolutional layers.

The mean of the subgrid momentum forcing predicted by the neural network can take any real value, as such we do not use any activation function in the final layer for the first two channels. However, the output predicted for the standard deviations are required to be positive. To enforce this constraint, we use a softplus function, defined by,

$$\mathrm{softplus}(x) = \ln(1 + \exp x) > 0, \tag{8}$$

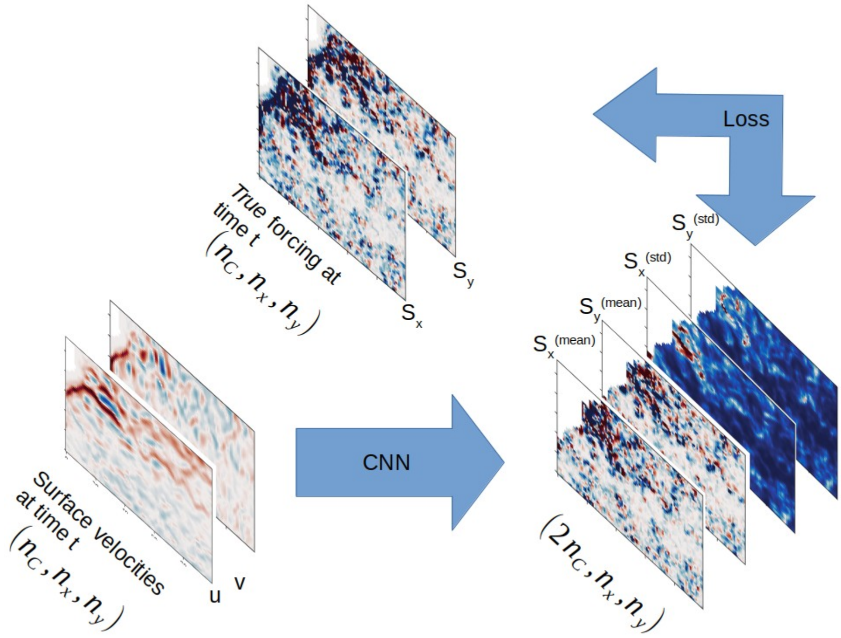


Figure 2: Our neural network outputs four maps: the two first maps are the maps of the means of the predicted forcing components, the last two maps are the standard deviation of the predicted forcing components.

as a final activation function for the two output channels associated with the standard deviations.

2.6 Training and Validation Procedure

We now describe our training procedure. The inputs are fed into the neural network in the form of mini-batches (i.e. small batches of several samples stacked along an extra dimension), rather than individually, such that the dimensions of our input tensors are $(n_{\text{batch}}, n_C, n_x, n_y)$, where n_{batch} is the number of samples per mini-batch, set to $n_{\text{batch}} = 4$ in our experiments. A common practice in the methodology of neural networks is to normalize inputs to be distributed within the interval $[-1, 1]$, to avoid vanishing and exploding gradients in the application of the back-propagation algorithm. Here, we multiply the surface velocities by a factor of 10. This same transformation is applied in testing.

The targets used to train and evaluate our neural network consist of the *true* subgrid momentum forcing computed in Equation 1 for a given subdomain. We train our neural network on data from the piControl simulation. We restrict the training data to a combination of four selected sub-domains of the oceans — shown as gray rectangles in Figure 1b, see also Table 1— that correspond to various dynamical regimes: the Gulf Stream extension, the Equatorial Atlantic, just south of the Equatorial Pacific, and in the South Pacific gyre. We leave further improvements through more advanced selection and weighting of the training subdomains for future work. We select the first 80% of the data (approximately spanning 16 years) as training data, and the final 15% (approximately spanning 3 years) are used for validation. We ignored 5% of the data (1 year) to avoid any correlation between the training data and the validation data, as it could cause validation metrics to become over-optimistic.

Table 1: Subdomains used for training and validation.

subdomain	latitude range	longitude range
A	35.0°, 50.0°	-50.0°, -20.0°
B	-40.0°, -25.0°	-180.0°, -162.0°
C	-20.0°, -5.0°	-110.0°, -92.0°
D	0.0°, 15.0°	-48.0°, -30.0°

In the training phase samples are entirely shuffled across the time dimension, as well as across subdomains. This allows to jointly train on data from all selected subdomains simultaneously. However, this requires the tensor inputs obtained from all the subdomains to have the same spatial sizes. We therefore crop the input tensors according to the smallest size across subdomains for both spatial dimensions, resulting in training samples of spatial extent $(n_x, n_y) = (38, 45)$.

We compute the average loss — defined in Section 2.4 — over the samples of a mini-batch and across the two components of the forcing, and across both longitude and latitudes. The average loss is then back-propagated to obtain the derivatives of the loss function with respect to the neural network's parameters. The neural network's parameters are then updated using the ADAM algorithm (Kingma & Ba, 2015). ADAM has become one of the go-to optimization algorithm in the Deep Learning community, which is in part due to its robustness to the choice of the learning rate and its quick convergence. After the neural network's parameters have been updated, we repeat the same process with a new mini-batch, and so on, until all the training data has been used, which corresponds to one epoch of training. At this point, we compute the average loss over the validation data, which was not used for optimization. We track this validation loss over the whole set of training epochs and repeat this process. We implement early stopping so that training stops once the validation loss has not improved for four consecutive epochs of training. More details about our final choice of hyperparameters, such as the learning rate, hand-picked through a validation procedure, can be found in Appendix A.

3 Offline Tests on a Global Scale

We test our stochastic deep learning parameterization on a global scale and demonstrate its generalization properties offline via test metrics for which notation is introduced in Section 3.2. In Section 3.3 we first carry out a test on piControl in order to assess the ability of our neural network model to generalize to subdomains and dynamical regimes not seen during the training phase. We then carry out a test on 1ptCO2 in Section 3.4, where the CO_2 levels in the atmosphere reach double those of the piControl simulation. Our results show that our stochastic deep learning parameterization performs well in this new climate, without requiring further training of our neural network. This is crucial if such parameterizations are to be used for climate projections (Rasp et al., 2018; O'Gorman & Dwyer, 2018).

3.1 Global Reconstruction for Offline Testing

We directly apply our trained neural network to the global coarse velocities for offline testing. When applying the neural network to global data, we extend the input velocities cyclically along the zonal dimension, thus ensuring the output covers all longitudes. This is not possible along the meridional dimension, thus resulting in the loss of $p = 10$ grid boxes (see Section 2.5) along the meridional dimension at both extreme latitudes.

Velocity snapshots are assembled to form small mini-batches with size 4 (equivalent to 4 days); the size is determined by the available GPU memory. Non-ocean points of the input grid are stored as $NaNs$. In our tests, we therefore ignore locations whose receptive field intersect with a continent and show them as greyed-out in the maps shown thereafter (note, the receptive field of a neuron within the neural network's output is the set of input neurons that impact its value). We leave the treatment of near-continent grid points for future work.

3.2 Metrics and Statistics for Offline Performance

To quantify the offline accuracy of our neural network's predictions of the subgrid momentum forcing, we define several metrics. We note $T = 7300$ the total number of days over which these metrics are computed.

We first define our notation for the standard Mean Square Error (MSE) and correlation. To make explicit the dimension along which the data is reduced to compute these two metrics, we write $\mathrm{MSE}_{C,i,j,-}$ for the time-mean MSE of the $C \in \{X, Y\}$ component of the forcing, where the reduction is carried out along the time axis, i.e.

$$\mathrm{MSE}_{C,i,j,-} = \frac{1}{T}\sum_{t=1}^{T}\left(\hat{\mathbf{S}}^{(\mathrm{mean})}_{C,i,j,t} - \mathbf{S}_{C,i,j,t}\right)^2, \quad i = 1,\ldots,n_x, \quad j = 1,\ldots,n_y. \tag{9}$$

The combined MSE, that encompasses both components X and Y, can be shown on a map, and is defined as

$$\mathrm{MSE}_{i,j,-} = \frac{1}{T}\sum_{t=1}^{T}\left\{\left(\hat{\mathbf{S}}^{(\mathrm{mean})}_{X,i,j,t} - \mathbf{S}_{X,i,j,t}\right)^2 + \left(\hat{\mathbf{S}}^{(\mathrm{mean})}_{Y,i,j,t} - \mathbf{S}_{Y,i,j,t}\right)^2\right\}. \tag{10}$$

We also define a scalar MSE according to,

$$\mathrm{MSE} = \frac{1}{n_x\ n_y\ T}\sum_{t=1}^{T}\sum_{i=1}^{n_x}\sum_{j=1}^{n_y}\left\{\left(\hat{\mathbf{S}}^{(\mathrm{mean})}_{X,i,j,t} - \mathbf{S}_{X,i,j,t}\right)^2 + \left(\hat{\mathbf{S}}^{(\mathrm{mean})}_{Y,i,j,t} - \mathbf{S}_{Y,i,j,t}\right)^2\right\}. \tag{11}$$

In addition to the standard MSE, we define an R^2 coefficient which is normalized by the value of the true subgrid forcing such that

$$\mathrm{R}^2_{C,i,j,-} = 1 - \frac{\sum_{t=1}^{T}\left(\hat{\mathbf{S}}^{(\mathrm{mean})}_{C,i,j,t} - \mathbf{S}_{C,i,j,t}\right)^2}{\sum_{t=1}^{T}\mathbf{S}^2_{C,i,j,t}}, \quad C = X, Y, \tag{12}$$

and its scalar version R^2_C, according to,

$$\mathrm{R}^2_{C} = 1 - \frac{\sum_{t=1,i,j}^{T}\left(\hat{\mathbf{S}}^{(\mathrm{mean})}_{C,i,j,t} - \mathbf{S}_{C,i,j,t}\right)^2}{\sum_{t=1}^{T}\mathbf{S}^2_{C,i,j,t}}, \quad C = X, Y. \tag{13}$$

We note that $\mathrm{R}^2_C \leq 1$ and if $\hat{\mathbf{S}}^{(\mathrm{mean})}_{C,i,j,t}$ is zero, R^2_C is 0. The advantage of this quantity is that it is easier to interpret when shown on a map — values close to 1 indicate that our predictions account for a large part of the average amplitude of the subgrid momentum forcing, while values close to 0 would indicate the opposite.

In order to verify that our model is not simply predicting the seasonal climatology of the subgrid momentum forcing, we define a modified version of this quantity, according to,

$$\mathrm{R}^{2,\mathrm{clim}}_{C,i,j,-} = 1 - \frac{\sum_{t=1}^{T}\left(\hat{\mathbf{S}}^{(\mathrm{mean})}_{C,i,j,t} - \mathbf{S}_{C,i,j,t}\right)^2}{\sum_{t=1}^{T}\left(\mathbf{S}^{clim}_{C,i,j,t} - \mathbf{S}_{C,i,j,t}\right)^2}, \quad C = X, Y, \tag{14}$$

where $\mathbf{S}^{clim}_{C,i,j,t}$ is the climatological $C-$component subgrid momentum forcing at location i, j and time t. This metric allows us to assess what percentage of the signal's variance we account for, after removing the inherent variability due to the seasonal climatology.

Another quantity of interest given our probabilistic representation of the subgrid momentum forcing parameterization is that of the standardized residuals, given by,

$$\mathbf{e}_{C,i,j,t} = \frac{\hat{\mathbf{S}}^{(\text{mean})}_{C,i,j,t} - \mathbf{S}_{C,i,j,t}}{\hat{\mathbf{S}}^{(\text{std})}_{C,i,j,t}}, \quad C = X, Y. \tag{15}$$

Under our idealized assumption, these normalized residuals are expected to follow a standard normal distribution.

We will also use confidence intervals, to quantify the uncertainty in the predicted subgrid forcing and evaluate it performance. Under the Gaussian assumption, a 95% confidence interval corresponds to,

$$\mathbf{S}^{(\text{mean})}_{C,i,j} \pm 1.96\ \mathbf{S}^{(\text{std})}_{C,i,j}, \quad C = X, Y. \tag{16}$$

3.3 Generalization & Subdomains — Test on piControl

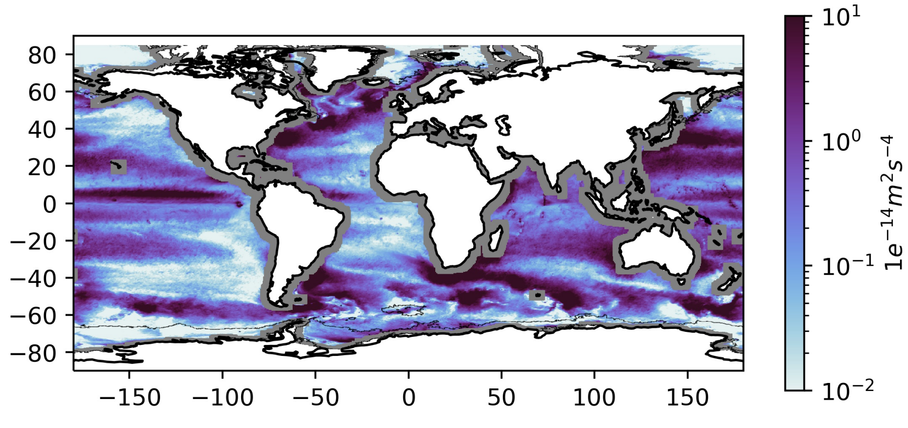
We carry out an offline test of our neural network on global scale data from piControl. There are large variations in subgrid eddy momentum in the piControl (Fig. 3a) across the oceans, with the largest amplitude occurring in eddy rich regions such as the Gulf Stream, Kuroshio, Southern Ocean and equatorial regions. There is a strong coherence between the pattern of the variance of the mean of the true subgrid forcing (Fig. 3a) and that of the predicted forcing (Fig. 3b). This coherence holds for the zonal and meridional component of the forcing, as shown for example in the correlation map between the true zonal forcing and the mean component of the predicted zonal forcing (Fig. C1).

The time-mean MSE over both components of the forcing (eqn. 11) can vary by several orders of magnitude from one region to another (Fig. 4a). However, these changes are largely due to the inherent spatial variability of the subgrid forcing, evident by comparing its spatial pattern (Fig. 3a) with the spatial pattern of the MSE (Fig. 4a). Therefore, the $R^{2,\text{clim}}_{i,j,-}$ coefficient (eqn. 14) is more informative of the neural network's performance (Fig. 4b).

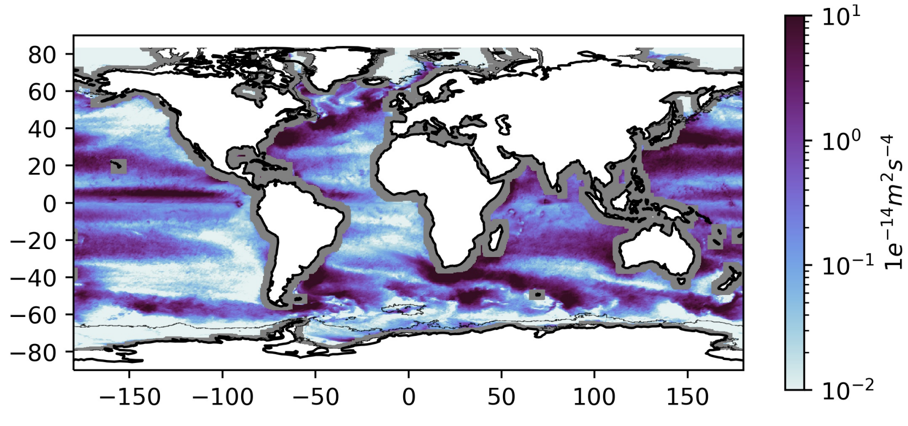
In most regions of the oceans, our neural network is able to account for more than 70% of the signal's variance, with performance nearing 90% in regions where the variance of the eddy momentum forcing is the highest, for instance in the Gulf Stream region and Southern Ocean (see the appendix for maps of the $R^{2,\text{clim}}$ computed for each component of the forcing – Fig. C2 – showing similar skill). These metrics indicate that the neural network generalizes well to most regions, despite being trained on only four small subdomains of the oceans. However, our neural network performs poorly in sea-ice covered regions, which is not surprising as the dynamics of these regions were not included in the training and varies widely from open ocean turbulence. Considering turbulence at the ocean-ice boundary will be left for future work, and will require numerical simulations that can adequately represent such processes.

The near-global $(60°S, 60°N)$ scalar R^2 value obtained is 0.869, while for $R^{2,\text{clim}}$ we obtain 0.855; the skill demonstrates the high performance of our neural network and further confirms that the neural network does not merely predict large variations due to the seasonal climatology. The global R^2 is higher than the average of R^2 values over the map due to the higher R^2 values in regions where the variance of the forcing is large (note that eqn. 13 is not the spatial average of eqn. 12).

To demonstrate some advantages of predicting the two moments of a Gaussian distribution, we focus on time series at two different locations. We compare the time se-

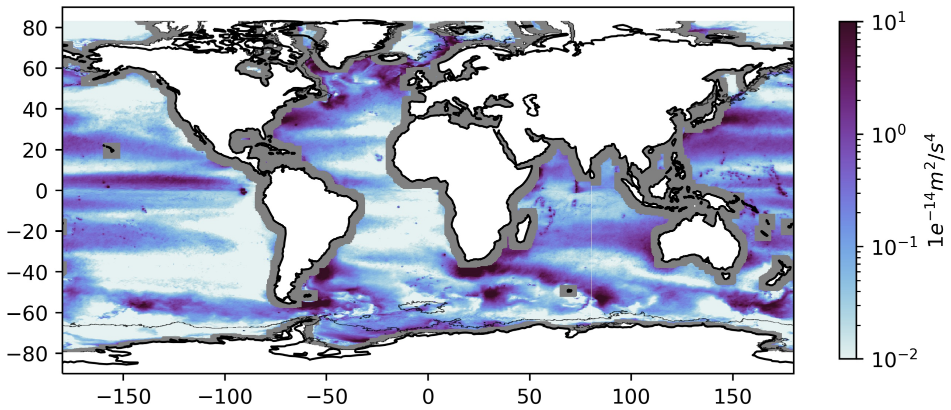


(a) True forcing

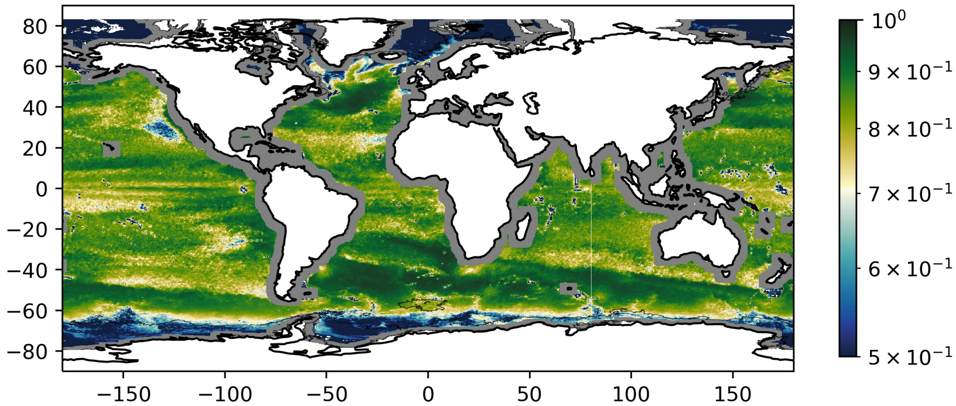


(b) Predicted mean

Figure 3: Time-mean variance of the norm of momentum forcing in piControl: (a) True forcing $||\mathbf{S}||$; (b) predicted mean, $||\hat{\mathbf{S}}^{mean}||$ in offline testing.



(a) Mean Square Error (piControl)



(b) $R^{2,\text{clim}}$ metric (piControl)

Figure 4: Time-mean (a) MSE (equation (10)) and (b) $R^{2,\text{clim}}$, defined using Equation (14) as $\mathrm{R}^{2,\text{clim}}_{X,i,j,-} + \mathrm{R}^{2,\text{clim}}_{Y,i,j,-}$, for piControl.

ries of the true and predicted zonal forcing at $30°N, 60°W$, which is located within the turbulent Gulf Stream region (Fig. 5a), and at $20°S, 104°W$, which corresponds to a more quiescent region with less mesoscale eddy activity (Fig. 5b). The true zonal forcing $S_{X,i,j,t}$ is shown along with the mean prediction $\hat{S}^{(\text{mean})}_{X,i,j,t}$ and the 95% confidence interval obtained from the predicted standard deviation $\hat{S}^{(\text{std})}_{X,i,j,t}$. The forcing is generally well approximated by the predicted mean forcing, except when extremes occur. However, the true forcing is, most of the time, within the 95% confidence interval. The predicted standard deviation $\hat{S}^{(\text{std})}_{X,i,j,t}$ varies greatly across the considered time window — indicating that the uncertainty of the forcing is not constant. Our neural network performs best in turbulent regions. This is in agreement with R^2 maps where higher values are observed in regions where the forcing is larger, and also with results from idealized ocean models (Bolton & Zanna, 2019; Zanna & Bolton, 2020). Finally, to investigate regions with a low R^2 score, we analyze the time series of the true and predicted meridional forcing at $29°N, 129°W$ (Fig. C3), which corresponds to a location near the West Coast of the United States where the R^2 score is 0.532. The time series indicates that the low R^2 occurs due to a few extreme events that are not well predicted.

To further analyze our predicted forcing from the piControl dataset, we study the global distribution of a stochastic simulation of subgrid momentum forcing generated using,

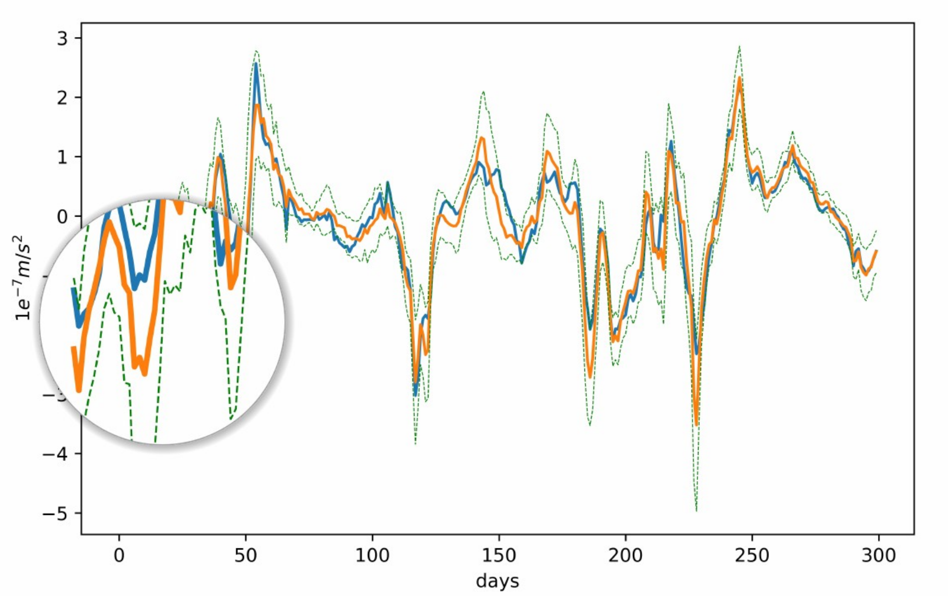
$$\tilde{\mathbf{S}}_{C,i,j} = \hat{\mathbf{S}}^{(\mathrm{mean})}_{C,i,j} + \epsilon_{C,i,j} \times \hat{\mathbf{S}}^{(\mathrm{std})}_{C,i,j}, \quad C = X, Y, \quad i = 1, \cdots, n_x, \quad j = 1, \cdots, n_y,$$

where the inputs to the neural network are the coarse surface velocities from piControl. The histograms of the global distribution of each component of the subgrid forcing for the true and simulated forcing show that the two distributions are very similar (Figure C4). However, the distribution of the true forcing has larger tails than that of the simulated forcing. This is partly due to our assumption that the distribution of the forcing, conditioned on the coarse surface velocity field, is Gaussian. We test this hypothesis by investigating the distribution of normalized residuals, defined by Equation 15. Figure C5a consists of the sample distribution of normalized residuals (blue), after subsampling one point out of ten along the time axis, and one point out of five along the spatial axes, shown together with the probability density function of the standard normal distribution (red. We also present a quantile-quantile (QQ)-plot of the sampled normalized residuals in Figure C5b, using the same subsampling procedure as in Figure C5a. The normalized residuals have much heavier tails than those of a standard normal. Hence, we could improve our model by using another distribution with heavier tails, or a multimodal distribution (Bishop, 1991). This approach will likely improve the offline prediction of extreme events which we have shown is problematic in our neural network. We leave this investigation for future work.

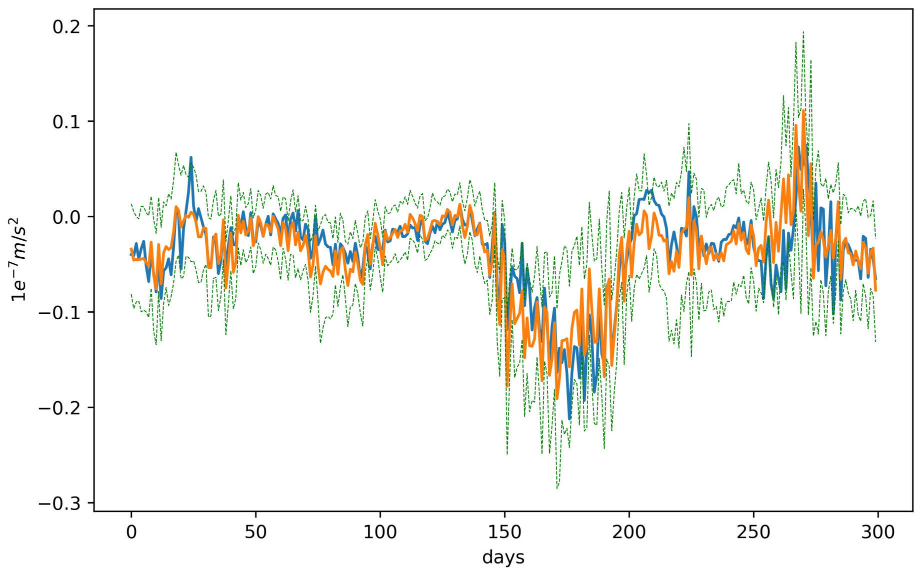
3.4 Generalization & Climate Change — Test on 1ptCO2

One key challenge for deep learning parameterizations in ocean and climate modeling is for them to be able to generalize to a new climate (O'Gorman & Dwyer, 2018). So far, we have only used data from the piControl simulations, both for training (Section 2.6) and testing (Section 3.3). Here, we test the trained neural network from section 2.6, without further tuning, using simulated data from 1ptCO2. The surface velocities, associated kinetic energy, and subgrid momentum forcing, are influenced by the CO_2 forcing. The time-mean standard deviation of the surface velocity between piControl and 1ptCO2 (Figure 6a) changes by up to 40% in some parts of the oceans. The majority of the changes are occurring in regions dominated by high kinetic energy in piControl such as the Gulf Stream region and its extension, the Kuroshio extension, or the Southern Ocean. Besides, we identify changes in the Indian Ocean and in the Arctic (ice-melt is likely related to changes in the latter). Similar changes in the subgrid momentum forcing are occurring as well (Figure 6b). The surface velocities used as inputs to the neural network and the target subgrid forcing to be predicted are therefore significantly different from those of piControl.

In order to compare performance of our neural network on piControl and 1ptCO2 we use the same metrics as in Section 3.3. The MSE and $R^{2,\mathrm{clim}}$ metrics computed over the 20 years of daily simulation data from 1ptCO2 are shown in Fig. 7. Our neural network performs as well for this new climate as it did for the climate it had been trained on (e.g., compare Fig. 7 with Fig. 4). The time-mean $R^{2,\mathrm{clim}}$ obtained on piControl and 1ptCO2 show little difference (Figure 7c), except in the North-East Atlantic and in certain polar regions which were partially ice-covered in piControl, where there is a slight decrease in performance (at most 0.1) as measured by the time-mean $R^{2,\mathrm{clim}}$. We compute scalar metrics of the performance of our neural network's performance over the piControl and 1ptCO2 simulation data, again limited to $60°S, 60°N$, and obtain 0.871 for R^2 and 0.858 for $R^{2,\mathrm{clim}}$, i.e. very similar to the values obtained for piControl. The neu-

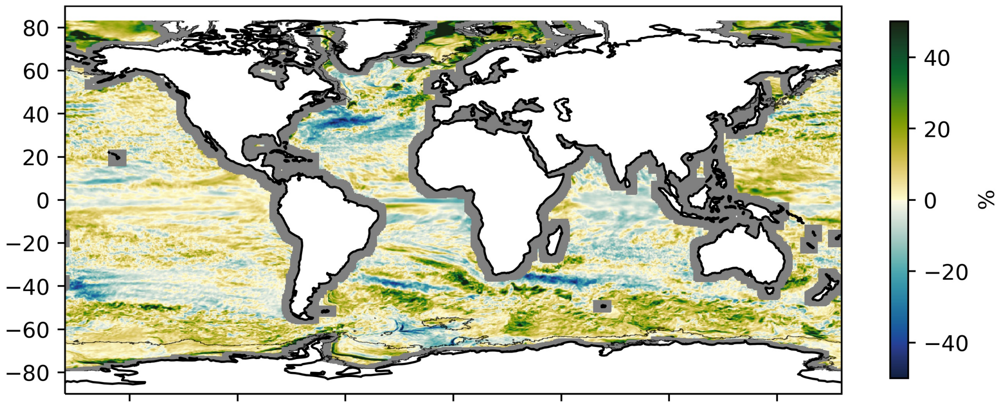


(a) turbulent

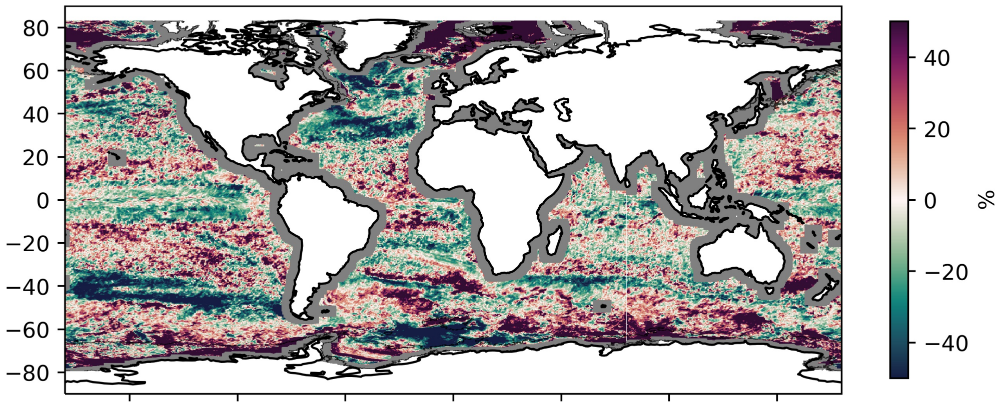


(b) quiescent

Figure 5: Time series of the zonal component of the subgrid momentum forcing at (a) $30°N, 60°W$, a location dominated by turbulent behavior and (b) $20°S, 104°W$, a more quiescent location for one year: true forcing (solid blue), mean of the predicted forcing (orange), and 95% confidence interval (green).



(a) Surface velocity relative change.



(b) subgrid momentum forcing relative change.

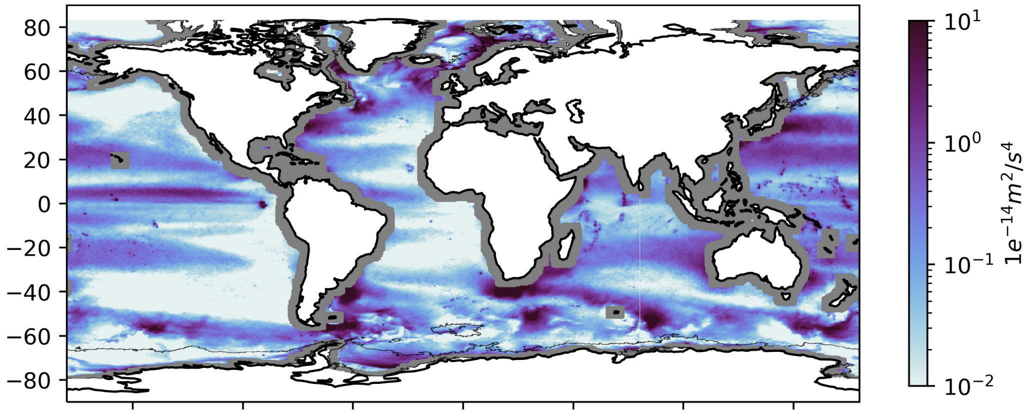
Figure 6: Relative difference between piControl and 1ptCO2 in the standard deviation of the (a) surface velocity norm and (b) subgrid forcing norm. Positive (negative) values indicate that the variance has increased (decreasing) in the 1ptCO2 compared to the piControl.

ral network for subgrid momentum forcing trained on piControl data generalizes well to an unseen warmer climate as simulated by a coupled high-resolution climate model.

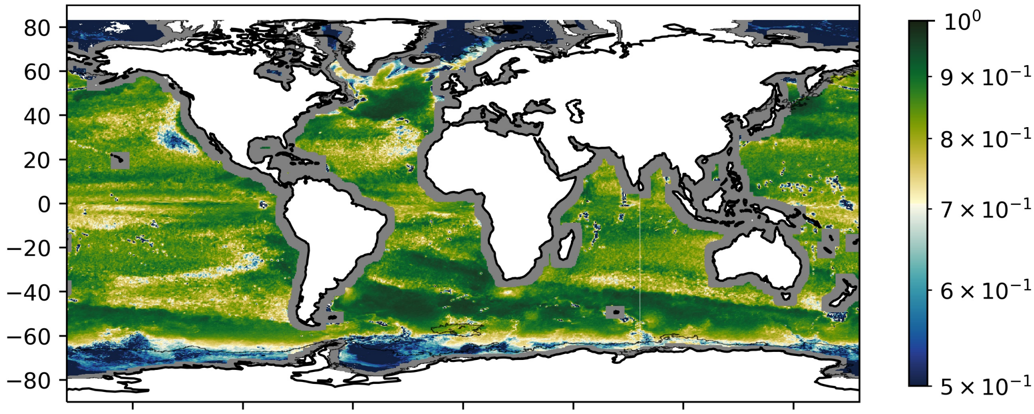
4 Online Implementation in an Idealized Model

Offline performance tests have not been good predictors for online performance, as shown for example in Zanna and Bolton (2020), at least not using current assessment metrics. The coupling between the machine learning (ML) parameterization and the prognostic model must satisfy the same numerical stability criteria and conservation properties as any physics-derived parameterization. Therefore, good offline performance is a necessary condition to the success of any ML parameterization, but is not a sufficient condition.

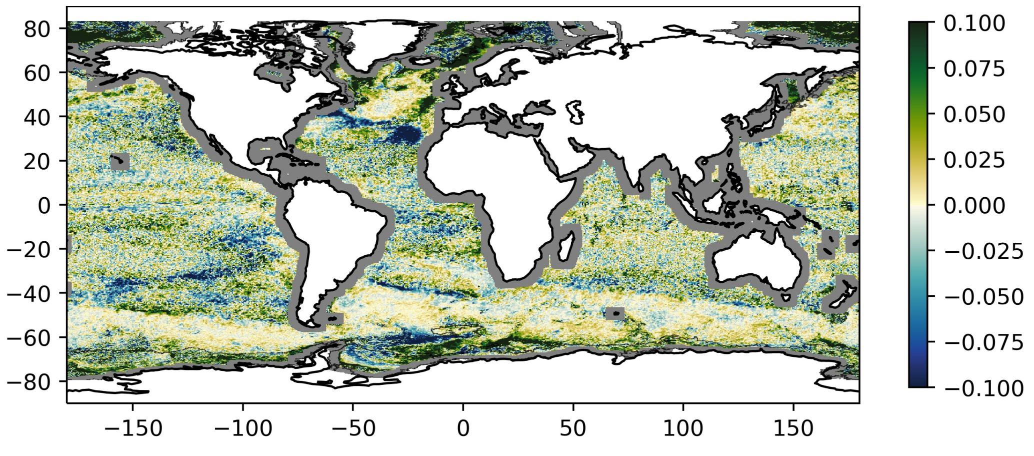
Zanna and Bolton (2020) implemented a convolutional neural network parameterization which, while physically constrained, led to too vigorous an inverse energy cascade. While the model was not numerically unstable, the behavior of the model was pushed



(a) Mean Square Error (1ptCO2)



(b) $R^{2,\text{clim}}$ metric (1ptCO2)



(c) Change in $R^{2,\text{clim}}$ between 1ptCO2 and piControl.

Figure 7: Performance of the trained neural network on 1ptCO2simulation: (1) Mean Square Error (1ptCO2) ; (b) $R^{2,\text{clim}}$ metric (1ptCO2); (c) Change in $R^{2,\text{clim}}$ between 1ptCO2 and piControl.

into a different dynamical regime in which the eddy mean-flow interactions dominated over the wind forcing. To ensure a reasonable dynamical behavior, the authors tuned down the parameterization by a spatially and temporally uniform multiplicative factor to reduce the magnitude of the forcing in an ad-hoc way.

The use of a stochastic parameterization has the potential to damp the eddy (and destabilizing) feedbacks seen in Zanna and Bolton (2020). Here, we use the same idealized barotropic shallow water model as in Zanna and Bolton (2020) (see their study for further details about the model, or our code) to implement the stochastic deep learning parameterization learned from complex CM2.6 data. The stochastic parameterization is implemented in a 40 km horizontal resolution run, and we compare the runs to a high-resolution model run at 10 km horizontal resolution, hence mimicking the change in resolution between CM2.6 simulation data and the coarse-grained data we generated to diagnose the momentum forcing.

Unlike CM2.6 which was on a B-grid, the shallow water model is discretized on an Arakawa C-grid. Therefore, at each time step of the integration, we first interpolate the two velocity components on tracer points and then pass them through our neural network. This produces, for each grid box and for each component of the forcing, a mean $\hat{\mathbf{S}}^{(\text{mean})}_{C,i,j}$ and a standard deviation $\hat{\mathbf{S}}^{(\text{std})}_{C,i,j}$. The stochastic subgrid momentum forcing $\tilde{\mathbf{S}}$ implemented in the shallow water model is then generated (see schematics in Fig. 10) according to,

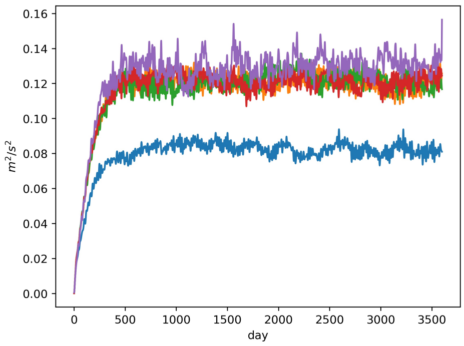
$$\tilde{\mathbf{S}}_{C,i,j} = \hat{\mathbf{S}}^{(\text{mean})}_{C,i,j} + \epsilon_{C,i,j} \times \hat{\mathbf{S}}^{(\text{std})}_{C,i,j}, \quad C = X, Y, \quad i = 1, \cdots, n_x, \quad j = 1, \cdots, n_y, \tag{17}$$

where the $\epsilon_{C,i,j}$ are sampled according to i.i.d. standard normal distributions. The field $\tilde{\mathbf{S}}$ is then interpolated back to the u and v grid for the X and Y components, respectively, and used as the value of the subgrid momentum forcing in the shallow water model.

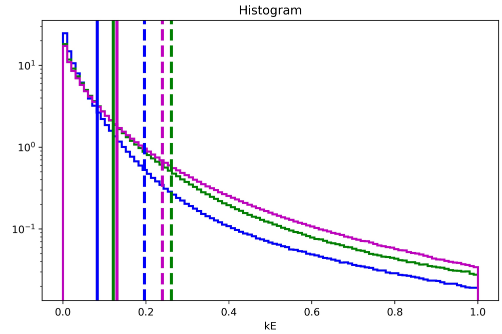
We ran the model for 10 years and produced 3 different ensemble members of the parameterized model. The parameterized simulations are stable and produced a physically-consistent state without any tuning or scaling factor. The kinetic energy of the flow is improved: both the mean and the standard deviation are very close to the high-resolution simulation (Fig. 8). Similarly to Zanna and Bolton (2020), the variance of the velocity fields (not shown) and sea surface height (Fig. 9) are vastly improved by the parameterization. However, changes in the mean velocity are rather small (not shown). We believe that the simplicity of the shallow water model used in the present study is at the core of the lack of substantial improvement in the mean flow and will be tested in a more complex model in future work. Unlike Zanna and Bolton (2020), no physical constraint was imposed when learning the neural network parameterization in our study; yet, we do not observe any drift in the model. Despite using zero-padding during the implementation, the solutions near the boundaries are not strongly impacted, as reported by Zanna and Bolton (2020). Overall, the coarse resolution stochastic simulations are 25% slower than the unparameterized runs but more than 40 times faster than a high-resolution simulation at 10 km on the same CPU. However, this statement is to be taken with care as the high-resolution simulation was not optimized.

5 Discussion

Current parameterizations of ocean and atmosphere processes remain a large source of bias and uncertainty in climate models. Therefore, harnessing state-of-the-art Deep Learning and statistical methods to improve parameterizations of subgrid processes has recently raised a lot of interest (Rasp et al., 2018; Bolton & Zanna, 2019; Yuval & O'Gorman, 2020; Zanna & Bolton, 2020). Here, we have demonstrated the potential of Deep Learning approaches for the problem of ocean momentum subgrid parameterizations using data generated by a realistic coupled climate model, as opposed to data from idealized ocean-



(a) Time series



(b) Histogram

Figure 8: Kinetic energy $[m^2/s^2]$ (a) time series, and (b) histogram for the low resolution unparametrized simulation at 30 km (blue), low resolution parameterized ensemble member simulations (green, orange, red), and filtered + coarse-grained high-resolution simulation (purple). In panel b: the solid lines indicate the mean and the dashed lines the standard deviation of the simulated kinetic energy; note that only one ensemble member is shown, but the other ensemble members produce similar statistics.

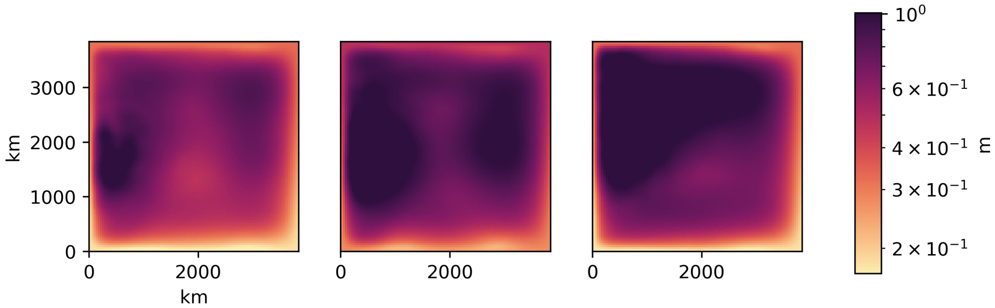


Figure 9: Standard deviation of sea surface height $[m]$ for the (left) low resolution simulation; (middle) one ensemble member from the parameterized versions; (right) the high resolution simulation.

only quasi-geostrophy or primitive equation simulations (Bolton & Zanna, 2019; Zanna & Bolton, 2020).

The use of data from realistic coupled climate models to train Deep Learning is non-trivial due to the size of the problem, the use of the tripolar irregular spherical grid, and the coupling between the ocean and the atmosphere. Here, we establish a filtering and coarse-graining procedure to diagnose the subgrid momentum forcing in a global model and show that using only a limited number of subdomains, we can train a neural network to skillfully predict the subgrid momentum forcing over the global ocean, and in a different climate with increased CO_2 levels. However, there are several remaining challenges. We have shown that the offline skill of the predictions is lower in regions where sea-ice is present. Therefore, to improve parameterizations of ocean mesoscale eddies in these regions, it might be necessary to acquire data that can faithfully represent these

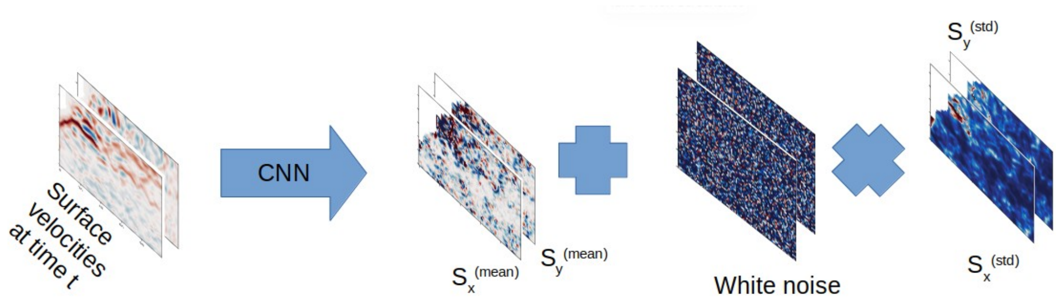


Figure 10: Procedure for generating the stochastic parameterization (eqn. 17) implemented in the coarse resolution idealized model, based on the trained neural network.

interactions. Another outstanding challenge is related to grid boxes located near continents, due the filtering and learning phases.

The neural network was trained to predict the parameters (mean and standard deviation) of a Gaussian probability distribution at each grid box, therefore providing a probabilistic approach to predicting the subgrid forcing with Deep Learning. This probabilistic approach attempts to account for both the uncertainty in the mapping between the coarse velocity field and the subgrid forcing and the uncertainty in the data itself. Stochasticity has been shown to improve model bias and produce more reliable ensemble predictions (Berner et al., 2017). Besides, while most current Deep Learning approaches to the parameterization of subgrid processes have been deterministic, a stochastic approach could be key when it comes to online implementations. Many Deep Learning implementations of parameterizations trained offline have resulted in poor stability properties or unrealistic flows in online simulations. Stochasticity could potentially solve this issue as shown in previous work (Palmer, 2012; Zanna et al., 2017). Using an idealized shallow water model, we showed that implementing our stochastic parameterization results in stable simulations and produces a realistic flow without any tuning. However, while the stochastic parameterization vastly improved some metrics (mean and variance of the kinetic energy), the impact on other metrics were only modest (e.g., zonal velocities).

The probabilistic approach presented here to learning the subgrid forcing remains simple and could be applied to parameterizing other processes. Yet it could benefit from more advanced probabilistic modeling. While we limited ourselves to conditionally i.i.d. Gaussian distributions, our analysis of residuals shows that representing higher moments could lead to a better representation of the distribution of subgrid forcing. In addition, we do not account for model uncertainty, i.e. uncertainty in the parameters of the neural network (Jospin et al., 2020). While Bayesian neural networks remain computationally more expensive, recent progress on that front could be an interesting avenue of investigation, and provide additional assurance compared to single outputs.

Finally, combining closed-form parameterizations with stochastic Deep Learning approaches could be another fruitful avenue. For instance, it would be possible to predict the mean forcing via a closed-form equation, such as done by Zanna and Bolton (2020) using equation-discovery methods, while representing higher-order moments via a probabilistic Deep Learning approach similar to that proposed in this manuscript. This approach could improve our understanding of missing processes and their representation in climate models. While the effects of Deep Learning subgrid parameterizations on climate projections remain to be ascertained, the benefits of Deep Learning could be greater if they are used to understand processes from a probabilistic perspective.

Appendix A Training Hyperparameters

The values of the hyperparameters used in our training procedure are provided in Table A1. The learning rate is decreased through the training procedure, hence we provide its initial value (epoch 0) and epochs at which it is decreased. The provided number of epochs corresponds to the maximum number of training epochs. In practice, training usually stops earlier due to our implementation of early stopping.

Table A1: Hyperparameter values for training

Hyperparameter		Value
Number of epochs		100
Learning rate	Epoch 0	$5e^{-4}$
	Epoch 10	$5e^{-5}$
	Epoch 20	$5e^{-6}$
Batch size		4
Filter sizes	Layers $1-2$	5
	Layers $3-8$	3
Padding	No	

Appendix B Generation of Low-Resolution Data and Estimates of the Missing Mesoscale Forcing

In this appendix we provide pseudo-code for the generation of the low-resolution data based on the CM2.6 high-resolution dataset. This algorithm makes use of two functions whose pseudo-code is also provided, *filter*, which applies a Gaussian filter to the passed data weighted by the cell areas, and *advections*, which computes the advection term of a discrete velocity field.

Algorithm 1: Filtering & Coarse-graining procedure

Data: $u_{i,j}, v_{i,j}, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n$

Result: $\overline{u}_{i,j}, \overline{v}_{i,j}, \overline{S}_{X,i,j}, \overline{S}_{Y,i,j}$

`/* Pre-processing: replace nans (i.e. land points) with zeros */`

if $u_{i,j} == NAN$ **then** $u_{i,j} \leftarrow 0$;

if $v_{i,j} == NAN$ **then** $v_{i,j} \leftarrow 0$;

`/* Compute the filtered high-rez surface velocities, for each component */`

$\overline{u}_{i,j} \leftarrow filter(u_{i,j})$; `/* filter function defined below */`

$\overline{v}_{i,j} \leftarrow filter(v_{i,j})$;

`/* Compute the advection term of the filtered surface velocities */`

$\psi_{X,i,j}, \psi_{Y,i,j} \leftarrow advection(\overline{u}_{i,j}, \overline{v}_{i,j})$; `/* advection function def. below */`

`/* Compute the filtered advection term from high-rez surface velocities */`

$\phi_{X,i,j}, \phi_{Y,i,j} \leftarrow advection(u_{i,j}, v_{i,j})$;

$\overline{\phi_{X,i,j}} \leftarrow filter(\phi_{X,i,j})$;

$\overline{\phi_{Y,i,j}} \leftarrow filter(\phi_{Y,i,j})$;

`/* Compute the components of the forcing term */`

$S_{X,i,j} \leftarrow \psi_{X,i,j} - \overline{\phi_{X,i,j}}$;

$S_{Y,i,j} \leftarrow \psi_{X,i,j} - \overline{\phi_{X,i,j}}$;

`/* Apply coarse-graining by factor` σ `*/`

$\overline{u}_{i,j} \leftarrow coarsen(\overline{u}_{i,j}, \sigma)$;

$\overline{v}_{i,j} \leftarrow coarsen(\overline{v}_{i,j}, \sigma)$;

$S_{X,i,j} \leftarrow coarsen(S_{X,i,j}, \sigma)$;

$S_{Y,i,j} \leftarrow coarsen(S_{Y,i,j}, \sigma)$;

Function `filter`**:**

Input: $u_{i,j}, dx_{i,j}, dy_{i,j}, \sigma$

Output: $\overline{u}_{i,j}$

$A_{i,j} = dx_{i,j} \times dy_{i,j}$; `/* Area of the` U`-cell */`

$$\overline{u}_{i,j} = \frac{\sum_{i',j'=-2\sigma}^{2\sigma} u_{i',j'} * A_{i',j'} \exp\left\{-\frac{(i'-i)^2+(j'-j)^2}{2\left(\frac{\sigma}{2}\right)^2}\right\}}{\sum_{i',j'=-2\sigma}^{2\sigma} A_{i',j'} \exp\left\{-\frac{(i'-i)^2+(j'-j)^2}{2\left(\frac{\sigma}{2}\right)^2}\right\}};$$

return $\overline{u}_{i,j}$;

Function `Advection`**:**

Input: $u_{i,j}, v_{i,j}, dx_{i,j}, dy_{i,j}$

Output: $\phi_{X,i,j}, \phi_{Y,i,j}$

$\partial_x u_{i,j} \leftarrow \frac{u_{i,j} - u_{i-1,j}}{dx_{i,j}}$;

$\partial_y u_{i,j} = \frac{u_{i,j} - u_{i,j-1}}{dy_{i,j}}$;

$\partial_x v_{i,j} = \frac{v_{i,j} - v_{i-1,j}}{dx_{i,j}}$;

$\partial_y v_{i,j} = \frac{v_{i,j} - v_{i,j-1}}{dy_{i,j}}$;

`/* Note here that the 4 quantities defined above need to be interpolated back on the` U`-grid before the two lines below */`

$\phi_{X,i,j} = u_{i,j}\partial_x u_{i,j} + v_{i,j}\partial_y u_{i,j}$;

$\phi_{Y,i,j} = u_{i,j}\partial_x v_{i,j} + v_{i,j}\partial_y v_{i,j}$;

return $\phi_{X,i,j}, \phi_{Y,i,j}$;

Remark: the high-resolution velocities are not defined on continents. To still be able to apply the Gaussian filtering used in the above procedure, we define the surface velocities at those points as zero.

642 ## Appendix C Complementary Figures

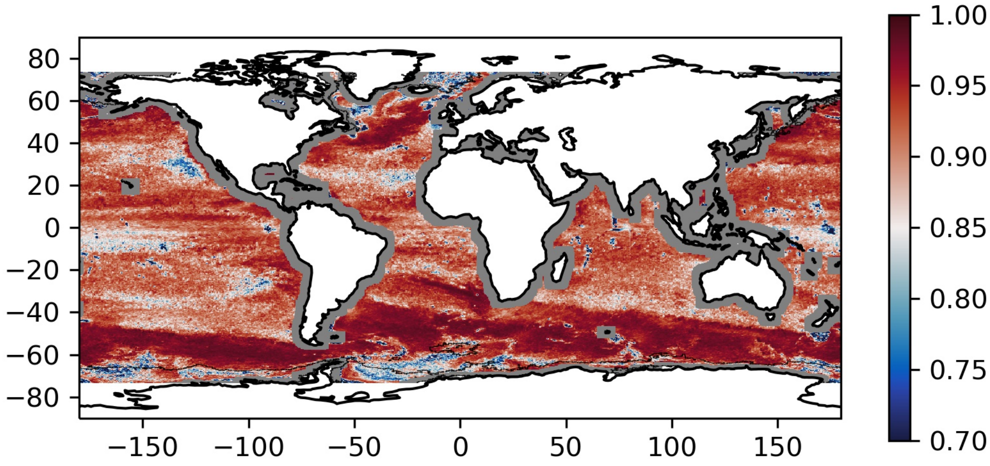
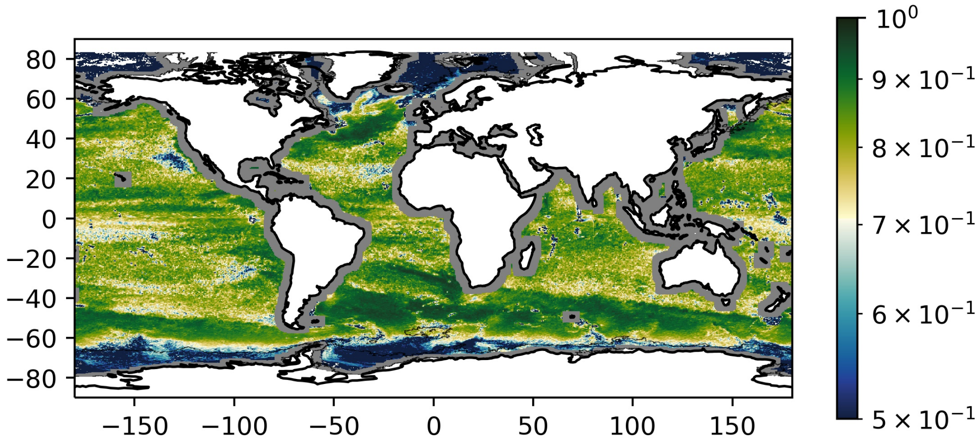
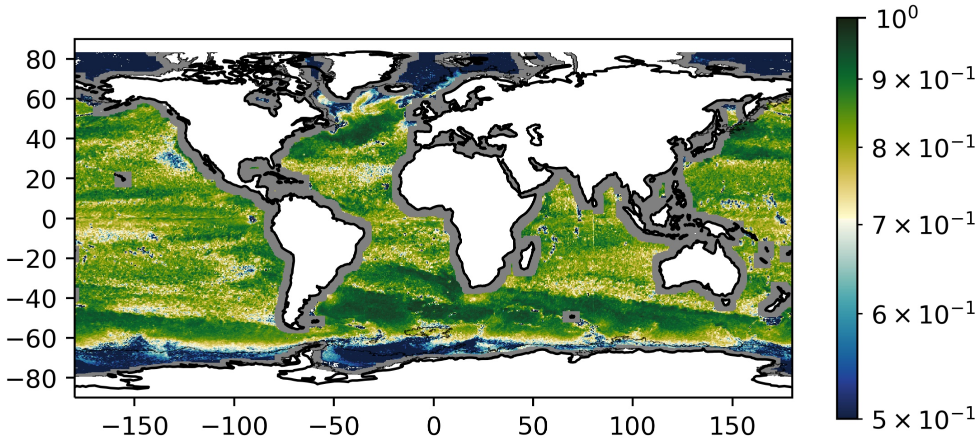


Figure C1: Correlation between S_X and $\hat{S}_X^{(mean)}$



(a) $R^{2,\text{clim}}$ metric (piControl) for the zonal component



(b) $R^{2,\text{clim}}$ metric (piControl) for the meridional component

Figure C2: Map of time-mean $R^{2,\text{clim}}$ metric in piControl for (a) the zonal component (b) the meridional component. The performances for both components are similar.

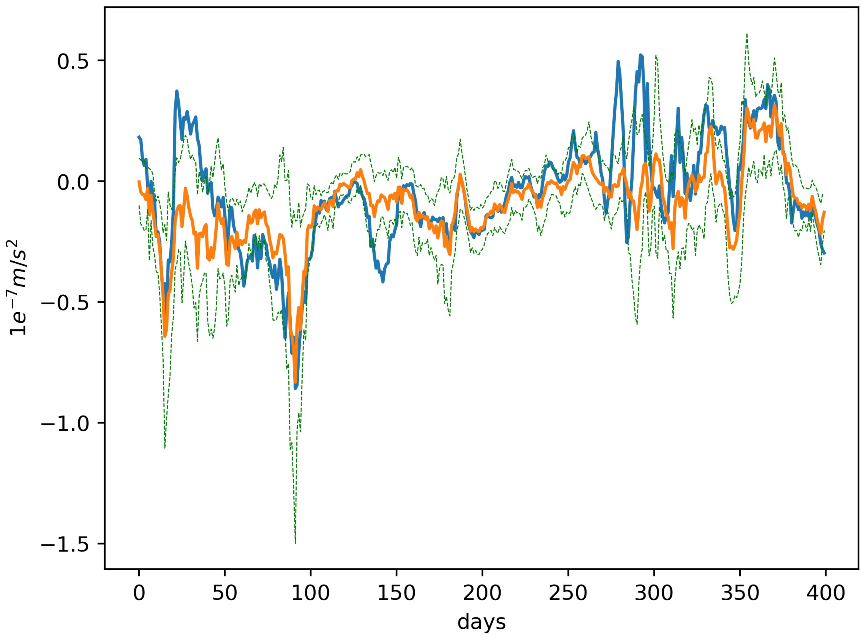


Figure C3: Time series of the true (solid blue) zonal component of the subgrid momentum forcing, mean zonal component of our neural network (orange), and 95% confidence interval (green), at $29°N, 129°W$. This location was selected within the region on the West coast of the United States where the R^2 is lower; this appears to be due to a few extreme events that are not accurately predicted, rather than a consistent ill-performance.

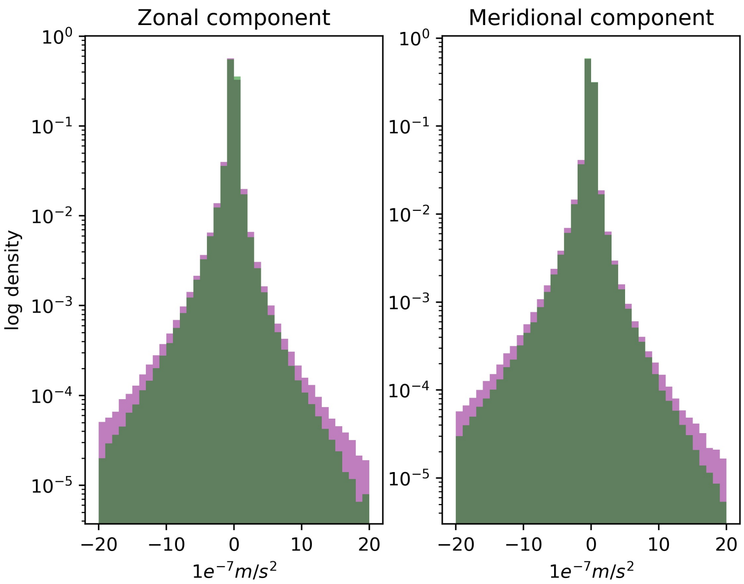
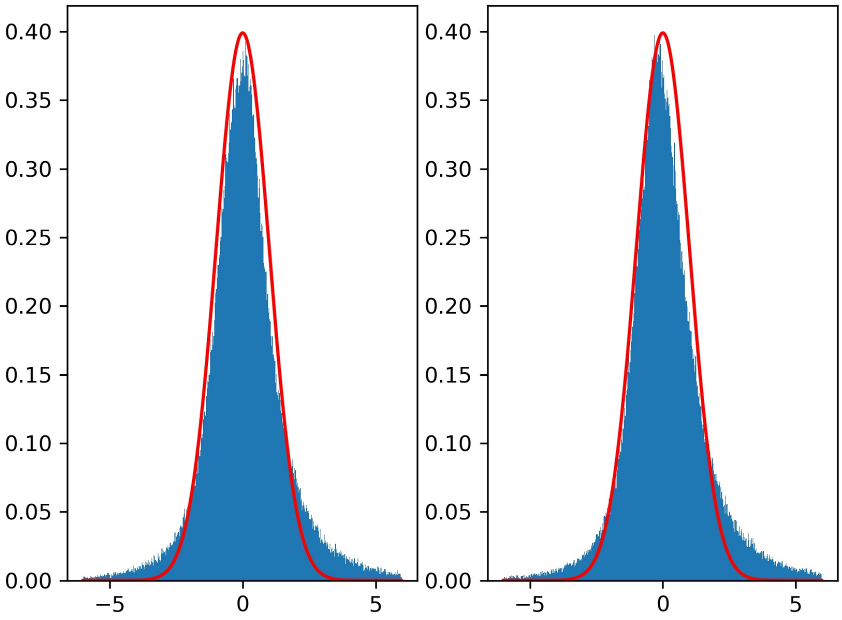
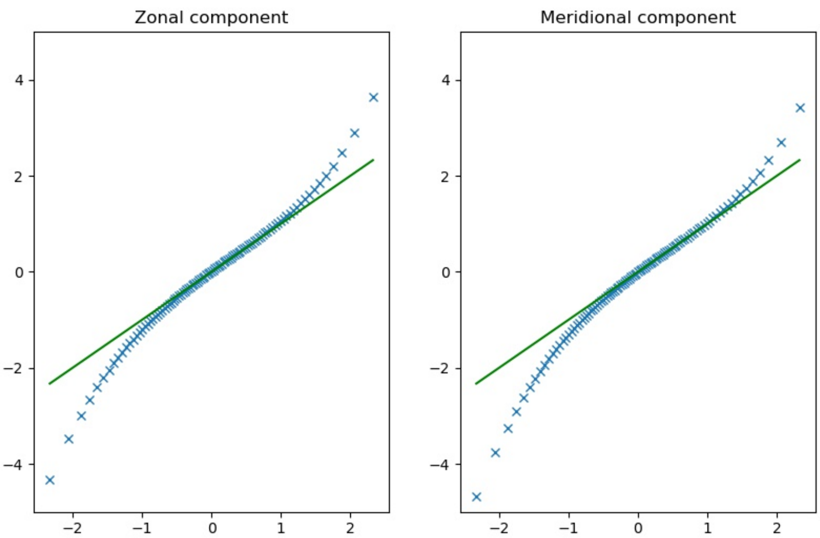


Figure C4: Sample log-probability distribution of true (purple) and stochastically simulated forcing (green) in the control simulation, for both components — zonal (left) and meridional (right). The histograms are shown on a log scale due to the hyperbolic-type distribution of the forcing.



(a) Sample distribution



(b) QQ plot

Figure C5: Distribution analysis of normalized residuals (eqn. 15) of subgrid momentum forcing in the control simulation. (a) Sample distribution (blue) along with the probability density function of the standard normal distribution (red), (b) QQ plot (blue) of normalized residuals against the standard normal distribution, and line (green) defined by $y = x$.

Acknowledgments

The work was supported in part by NSF-GEO 1912357, NOAA CVP NA19OAR4310364, and through the NYU IT High Performance Computing resources, services, and staff expertise. We thank Alistair Adcroft, Ryan Abernathey, Steve Griffies and V. Balaji for extremely valuable inputs at different stages of this work.

Data availability statement

We downloaded the simulation's ocean surface velocities from a Pangeo data catalog at https://raw.githubusercontent.com/pangeo-data/pangeo-datastore/master/intake-catalogs/ocean/GFDL_CM2.6.yaml made publicly available by the Geophysical Fluid Laboratory. The code used in this study can be accessed from two repositories: doi 10.5281/zenodo.4573438 for the data processing, neural network training and its tests; doi 10.5281/zenodo.4573448 for its implementation in a shallow water model.

References

Abernathey, R., Augspurger, T., Banihirwe, A., Blackmon-Luca, C., Crone, T., Gentemann, C., ... Signell, R. (2021). Cloud-native repositories for Big Scientific Data. *Computing in Science & Engineering*(01), 1-1. doi: 10.1109/MCSE.2021.3059437

Bachman, S. D. (2019). The GM+E closure: A framework for coupling backscatter with the Gent and McWilliams parameterization. *Ocean Modelling*, *136*, 85-106. doi: 10.1016/j.ocemod.2019.02.006

Balaji, V. (2021). Climbing down charney's ladder: machine learning and the post-Dennard era of computational climate science. *Philosiphical Transactions of the Royal Society A*, *379*(2194). doi: 10.1098/rsta.2020.0085

Berner, J., Achatz, U., Batté, L., Bengtsson, L., De La Cámara, A., Christensen, H. M., ... Yano, J. I. (2017). Stochastic parameterization: toward a new view of weather and climate models. *Bulletin of the American Meteorological Society*, *98*(3), 565–587. doi: 10.1175/BAMS-D-15-00268.1

Bishop, C. (1991). *Mixture density networks.* Birmingham.

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. doi: 10.1029/2018MS001472

Brankart, J.-M. (2013). Impact of uncertainties in the horizontal density gradient upon low resolution global ocean modelling. *Ocean Modelling*, *66*, 64-76. doi: 10.1016/j.ocemod.2013.02.004

Brenowitz, N. D., Beucler, T., Pritchard, M., & Bretherton, C. S. (2020). Interpreting and stabilizing machine-learning parametrizations of convection. *Journal of the Atmospheric Sciences*, *77*(12), 4357-4375. doi: 10.1175/JAS-D-20-0082.1

Davison, A. C. (2003). *Statistical models.* Cambridge University Press.

Delworth, T. L., Rosati, A., Anderson, W., Adcroft, A. J., Balaji, V., Benson, R., ... Zhang, R. (2012). Simulated climate and climate change in the gfdl cm2.5 high-resolution coupled climate model. *Journal of Climate*, *25*(8), 2755-2781. doi: 10.1175/JCLI-D-11-00316.1

Dunne, J. P., Horowitz, L. W., Adcroft, A. J., Ginoux, P., Held, I. M., John, J. G., ... Zhao, M. (2020). The GFDL Earth System Model Version 4.1 (GFDL-ESM 4.1): Overall coupled model description and simulation characteristics. *Journal of Advances in Modeling Earth Systems*, *12*(11), e2019MS002015. doi: 10.1029/2019MS002015

Ferrari, R., & Wunsch, C. (2009). Ocean circulation kinetic energy: reservoirs, sources, and sinks. *Annual Review of Fluid Mechanics*, *41*(1), 253-282. doi: 10.1146/annurev.fluid.40.111406.102139

Gagne II, D. J., Christensen, H. M., Subramanian, A. C., & Monahan, A. H. (2020). Machine learning for stochastic parameterization: Generative adversarial networks in the Lorenz '96 model. *Journal of Advances in Modeling Earth Systems*, *12*(3), e2019MS001896. doi: 10.1029/2019MS001896

Gent, P. R., & McWilliams, J. C. (1989). Isopycnal mixing in ocean circulation models. *Journal of Physical Oceanography*, *20*, 150–155. doi: 10.1175/1520-0485(1990)020⟨0150:IMIOCM⟩2.0.CO;2

Gerard, L. (2007). An integrated package for subgrid convection, clouds, and precipitation compatible with meso-gamma scales. *Quarterly Journal of the Royal Meteorological Society*, *133*, 711–730. doi: 10.1002/qj.58

Griffies, S. M. (2015). *A handbook for the GFDL CM2-O model suite.* `https://mom-ocean.github.io/assets/pdfs/CM2_O_suite_1990_tutorial_and_analysis.pdf`.

Griffies, S. M., Winton, M., Anderson, W. G., Benson, R., Delworth, T. L., Dufour, C. O., . . . Zhang, R. (2015, 02). Impacts on ocean heat from transient mesoscale eddies in a hierarchy of climate models. *Journal of Climate*, *28*(3), 952-977. doi: 10.1175/JCLI-D-14-00353.1

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

IPCC. (2013). *Climate change 2013: The physical science basis. contribution of working group I to the fifth assessment report of the intergovernmental panel on climate change* [Book]. Cambridge, United Kingdom and New York, NY, USA: Cambridge University Press. doi: 10.1017/CBO9781107415324

Jansen, M. F., Adcroft, A., Khani, S., & Kong, H. (2019). Toward an energetically consistent, resolution aware parameterization of ocean mesoscale eddies. *Journal of Advances in Modeling Earth Systems*, *11*(8), 2844-2860. doi: 10.1029/2019MS001750

Jones, T. R., Randall, D. A., & Branson, M. D. (2019). Multiple-instance superparameterization: 2. the effects of stochastic convection on the simulated climate. *Journal of Advances in Modeling Earth Systems*, *11*(11), 3521-3544. doi: 10.1029/2019MS001611

Jospin, L. V., Buntine, W., Boussaid, F., Laga, H., & Bennamoun, M. (2020). *Hands-on bayesian neural networks – a tutorial for deep learning users.*

Juricke, T. N., S. T. N. Palmer, & Zanna, L. (2017). Stochastic perturbations to sub-grid scale ocean mixing: Impacts on low frequency variability. *Journal of Climate*, *30*(13), 4997-5019. doi: 10.1175/JCLI-D-16-0539.1

Kingma, D. P., & Ba, J. L. (2015). Adam: a method for stochastic optimization. *arXiv preprint:1412.6980v9*, 1–15.

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (p. 3431-3440).

Mason, P. J., & Thomson, D. J. (1992). Stochastic backscatter in large-eddy simulations of boundary layers. *Journal of Fluid Mechanics*, *242*(28), 51–78. doi: 10.1017/S0022112092002271

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. doi: 10.1029/2018MS001351

Palmer, T. N. (2012). Towards the probabilistic earth-system simulator: a vision for the future of climate and weather prediction. *Quarterly Journal of the Royal Meteorological Society*, *138*(665), 841-861. doi: 10.1002/qj.1923

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

Sirignano, J., MacArt, J. F., & Freund, J. B. (2020). DPM: A deep learning PDE augmentation method with application to large-eddy simulation. *Journal of*

Computational Physics, *423*, 109811. doi: 10.1016/j.jcp.2020.109811

Stanley, Z., Grooms, I., Kleiber, W., Bachman, S. D., Castruccio, F., & Adcroft, A. (2020). Parameterizing the impact of unresolved temperature variability on the large-scale density field: Part 1. theory. *Journal of Advances in Modeling Earth Systems*, *12*(12), e2020MS002185. doi: 10.1029/2020MS002185

Treguier, A. M., Held, I. M., & Larichev, V. D. (1997). Parameterization of quasi-geostrophic eddies in primitive equation ocean models. *Journal of Physical Oceanography*, 567–580. doi: 10.1175/1520-0485(1997)027⟨0567:POQEIP⟩2.0.CO;2

Waterman, S., & Jayne, S. R. (2011). Eddy-mean flow interactions in the along-stream development of a western boundary current jet: An idealized model study. *Journal of Physical Oceanography*, *41*(4), 682-707. doi: 10.1175/2010JPO4477.1

Williams, P. D., Howe, N. J., Gregory, J. M., Smith, R. S., & Joshi, M. M. (2016). Improved climate simulations through a stochastic parameterization of ocean eddies. *Journal of Climate*, *29*(24), 8763-8781. doi: 10.1175/JCLI-D-15-0746.1

Yuval, J., & O'Gorman, P. A. (2020). Stable machine-learning parameterization of subgrid processes for climate modeling at a range of resolutions. *Nature Communications*, *11*(1), 1–10. doi: 10.1038/s41467-020-17142-3

Zanna, L., Bachman, S., & Jansen, M. (2020). Energizing turbulence closures in ocean models. *CLIVAR Exchanges/US CLIVAR Variations*, *18*(1), 3–8. doi: 10.5065/g8w0-fy32.

Zanna, L., & Bolton, T. (2020). Data-driven equation discovery of ocean mesoscale closures. *Geophysical Research Letters*, e2020GL088376. doi: 10.1029/2020GL088376

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

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