Benchmarking of machine learning ocean subgrid parameterizations in an idealized model

Andrew Ross[1], Ziwei Li[1], Pavel Perezhogin[1], Carlos Fernandez-Granda[1,2], Laure Zanna[1]

[1]Courant Institute of Mathematical Sciences, New York University, New York, NY, USA
[2]Center for Data Science, New York University, New York, NY, USA

Key Points:

- We introduce an open-source framework for training, testing, and evaluating data-driven subgrid parameterizations in an ocean model.
- Neural network parameterizations are stable and perform well online for some filtering operators, across a range of quantifiable metrics.
- A parameterization discovered with our new combined genetic programming and linear regression algorithm outperforms 147 other closures.

Corresponding author: Laure Zanna, laure.zanna@nyu.edu

Abstract

Recently, a growing number of studies have used machine learning (ML) models to parameterize computationally intensive subgrid-scale processes in ocean models. Such studies typically train ML models with filtered and coarse-grained high-resolution data and evaluate their predictive performance offline, before implementing them in a coarse resolution model and assessing their online performance. In this work, we provide a framework for systematically benchmarking the online performance of such models, their generalization to domains not encountered during training, and their sensitivity to dataset design choices. We apply this proposed framework to compare a large number of physical and neural network (NN)-based parameterizations. We find that the choice of filtering and coarse-graining operator is particularly critical and this choice should be guided by the application. We also show that our all of our physics-constrained NNs are stable when implemented online, but that performance across metrics can vary drastically. In addition, to test generalization and help with interpretability of data-driven parameterizations, we propose a novel equation-discovery approach combining linear regression and genetic programming with spatial derivatives. We find this approach performs on par with neural networks on the training domain but generalizes better beyond it. We release code and data to reproduce our results and provide the research community with easy-to-use resources to develop and evaluate additional parameterizations.

Plain Language Summary

Accurately predicting climate change requires running intensive computer simulations called climate models. Climate models divide the world into grid cells, solving an approximation of continuous equations that model the true dynamics. For accurate predictions, these cells must be small, or equivalently models must be high-resolution. However, even with modern supercomputers, running many high-resolution simulations is prohibitively expensive.

One solution is to run climate models at coarser resolution, but include "subgrid parameterizations," which account for physical processes occurring at finer scales to correct bias. Parameterizations are usually developed by analyzing the continuous equations and empirically determining formulae to predict unresolved effects. However, recent studies have applied machine learning (ML) methods to learn parameterizations automatically from limited high-resolution data.

This approach has shown promise, but also introduced new challenges with dataset preparation, evaluation, interpretability, and implementation. We provide an open-source framework for learning and evaluating parameterizations in a simplified model of the ocean. We use this framework to evaluate numerous ML methods and analyze how best to prepare datasets. We also develop a method of learning equation-based parameterizations which can be more easily interpreted and implemented. Our approach performs comparably to the best ML parameterizations, but generalizes better to oceanic conditions unseen during training.

1 Introduction

Current state-of-the-art climate models solve geophysical fluid equations on grids of size 25km and coarser. Models at this level of resolution are not able to accurately and sufficiently resolve processes with physical length scales smaller than the model grid, for example, convection in the atmosphere and mesoscale eddies in the ocean. Since increases in computational power will likely not enable climate models to resolve these processes before the effects of climate change ensue (Fox-Kemper et al., 2014; Schneider et al., 2017), we must represent subgrid-scale (SGS) processes with closure models (also known as parameterizations). Yet, these SGS models are some of the largest sources of bias and

uncertainties in climate projections: e.g., insufficient representations of transient eddies cause biases in modeled currents and sea surface temperature in the ocean (Griffies et al., 2015; Hewitt et al., 2020), and the precipitation pattern is strongly sensitive to the different subgrid cloud closures, thereby causing significant errors in climate projections (Stevens & Bony, 2013). Therefore, developing robust parameterizations remains an important task towards reliable climate projections.

In ocean circulation models, stratified turbulent processes are one of the primary targets of SGS closures (Pope, 2000; Vallis, 2017). Classic SGS models are typically designed with specific goals in mind; for example, to dissipate small-scale enstrophy (Smagorinsky, 1963), to reinject energy at larger scales via backscattering (Jansen & Held, 2014; Jansen et al., 2015), or to improve the representation of heat and tracer transport in the ocean interior (Redi, 1982; Gent & Mcwilliams, 1990; Gent et al., 1995). However, human choices in the design, formulation, and tuning of these SGS models sometimes lead to poor correlation between parameterized SGS forcing and true SGS forcing as diagnosed from high resolution simulations (Khani & Porté-Agel, 2017). This can result in unrealistic large-scale simulations despite strong recent progress in the representation of resolved processes (Griffies et al., 2009; Fox-Kemper et al., 2019). These shortcomings call for complementary, more systematic and data-driven approaches.

Recently, an increase in high-resolution observations and simulations combined with advances in machine-learning (ML) methods has propelled a surge in the development of data-driven SGS parameterizations in climate models (Krasnopolsky et al., 2010; Bolton & Zanna, 2019; Rasp et al., 2018; O'Gorman & Dwyer, 2018; Guillaumin & Zanna, 2021; Zanna & Bolton, 2021; Beucler et al., 2021; Guan et al., 2022; Yuval & O'Gorman, 2021; Frezat et al., 2022; Subel et al., 2022). Directly learning from data, ML methods automatically extract relevant information from observations and high-resolution simulations to improve coarse-resolution models at a reduced computational cost. Despite their universal approximation properties (Hornik et al., 1989), popular ML models such as neural networks are often opaque to interpretation and can extrapolate poorly to conditions unseen during training (Recht et al., 2018; O'Gorman & Dwyer, 2018; Bolton & Zanna, 2019; Subel et al., 2022).

The performance of data-driven approaches is greatly influenced by choices that must be made in dataset preparation. The formulation of the subgrid forcing term, either in terms of tendency or subgrid-scale fluxes, can change the stability of parameterized models (Yuval et al., 2021). Different filtering schemes also have significant effects on the online performance of subgrid parameterizations (Piomelli et al., 1988; Zhou et al., 2019; Frezat et al., 2022).

There is currently a vast number of possible choices in terms of ML models, training target formulation, and filtering and coarse-graining methods. However, few studies offer a direct and adequate comparison between data-driven ML methods and physical-based parameterizations. Moreover, well-defined quantitative (rather than qualitative) online metrics are lacking. In this paper, we introduce a family of datasets (Sections 2 and 3) and quantitative metrics (Section 4) for learning and evaluating ocean eddy subgrid parameterizations, both offline and online, using a quasi-geostrophic (QG) simulation (datasets and code are available open-source; see Appendix D). Our metrics quantify to what extent the time-averaged spectral and distributional properties of parameterized simulations match those of ground-truth high-resolution simulations, as well as whether they improve the accuracy of more predictable short-term dynamics. These metrics make it possible to comprehensively compare numerous parameterizations, and the effects of dataset design choices on their performance.

In Section 5, we perform such a study for fully convolutional neural network parameterizations, evaluating how offline and online performance change with different designs of inputs (i.e. types of feature variables – velocity or potential vorticity) and out-

puts (i.e. formulations of subgrid-scale forcing). Even for the best-performing neural networks, we find poor generalization to flow regimes unseen during training, consistent with previous literature (Recht et al., 2018; O'Gorman & Dwyer, 2018; Bolton & Zanna, 2019; Subel et al., 2021; Guan et al., 2022; Subel et al., 2022).

Motivated by these generalization issues, as well as the lack of interpretability of neural networks, there has been increasing interest in the physical sciences community in symbolic regression (also known as equation discovery). In symbolic regression, instead of an opaque model, the final output of training is a transparent equation, which often generalizes better (Rudy et al., 2017; Zhang & Lin, 2018; Champion et al., 2019; Zanna & Bolton, 2020; Mojgani et al., 2021). Many of these studies perform symbolic regression using sparse linear regression on top of a manually constructed basis of terms representing various operations (e.g. derivatives or multiples) of base features. Although powerful for small numbers of terms, this approach quickly becomes prohibitive because the space and time requirements grow exponentially if we consider higher-order operations.

To address these challenges, in Section 6 we introduce a novel algorithm for equation discovery based on genetic programming, an alternative form of symbolic regression that is stochastic but can more efficiently explore higher-order operations (Koza, 1994; Schmidt & Lipson, 2009; Xing et al., 2022). We adapt this algorithm to search over spatial differential operators, and combine it with linear regression and residual-fitting to more efficiently and accurately fit constants. We find that the discovered expression of symbolic parameterization includes features discovered in prior works, is superior to traditional physics-informed turbulence SGS closures, has similar performance to neural networks in both offline and online metrics, and generalizes better to unseen flow regimes than neural networks and baseline physical parameterizations.

2 Numerical Simulations

This section describes the simulations we use to generate our datasets, which are based on `pyqg` (Abernathey et al., 2022), a Python library that models quasi-geostrophic (QG) systems using pseudo-spectral methods. QG systems are able to capture the generation of ocean mesoscale eddies, the key process we parameterize in this study, and are often used to develop and test physic-based parameterizations (P. S. Berloff, 2005; Porta Mana & Zanna, 2014; Jansen & Held, 2014). In addition, QG systems are a reasonable approximation to the equations of motion in more realistic ocean models in the limit of strong stratification and rotation. Importantly for this study, which tests numerous parameterizations online, they can be simulated much more efficiently than full-fledged ocean models or GCMs.

2.1 Idealized two-layer QG model

We use a two-layer version of the QG model from `pyqg`. The model's prognostic variable is potential vorticity (PV), denoted as q_1 in the upper and q_2 in the lower layer:

$$q_m = \nabla^2 \psi_m + (-1)^m \frac{f_0^2}{g' H_m} \Delta\psi, \qquad m \in \{1, 2\}, \tag{1}$$

where ψ_m is the streamfunction with depth H_m, $\Delta\psi = (\psi_1 - \psi_2)$, and ∇ is the horizontal gradient operator. Zonal and meridional velocities are obtained from the streamfunction by the relations $u_m = -\partial_y \psi_m$ and $v_m = \partial_x \psi_m$, for each layer with $m \in \{1, 2\}$. We express the horizontal velocity as a single vector $\mathbf{u}_m = \langle u_m, v_m \rangle$. We use the beta-plane approximation, such that the Coriolis acceleration is a linear function of latitude (y) with slope β, such that $f = f_0 + \beta y$, and g' is the reduced gravity.

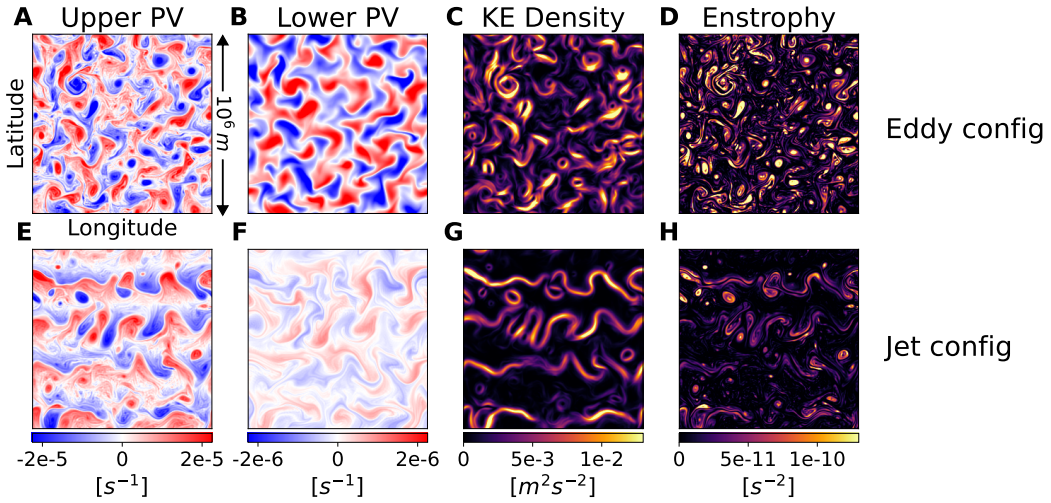

Figure 1. Snapshots of upper (A,E) and lower (B,F) potential vorticity (PV), barotropic kinetic energy (C,G), and barotropic enstrophy (D,H) for simulations run in eddy (A-D) and jet (E-H) configurations over a square, doubly-periodic domain of length $10^6\,m$. Eddy configuration results in an approximately isotropic distribution of vortices, while jet configuration results in the formation of stable, long-lived jets with more coherent latitudinal structure.

The prognostic equations, solved in spectral space, are:

$$\frac{\partial \hat{q}_m}{\partial t} = -\hat{\mathsf{J}}\left(\psi_m, q_m\right) - \mathrm{i}k\beta_m\hat{\psi}_m - \mathrm{i}kU_m\hat{q}_m - \delta_{m,2}r_{ek}\kappa^2\hat{\psi}_2 + \widehat{\mathrm{ssd}}, \tag{2}$$

where ∂_t is the Eulerian time derivative, $\widehat{(\ )}$ denotes taking the Fourier transform, and $\kappa = \sqrt{k^2 + l^2}$ is the radial wavenumber, where k and l are zonal and meridional wavenumbers, respectively. $\mathsf{J}\left(A\,,B\right) = A_xB_y - A_yB_x$ is the horizontal Jacobian. The mean PV gradient in each layer is $\beta_m = \beta + (-1)^{m+1}\frac{f_0^2}{g'H_m}\Delta U$, where $\Delta U = U_1 - U_2$ is a fixed mean zonal velocity shear between the two fluid layers. The Dirac delta function, $\delta_{m,2}$, indicates that the bottom drag with coefficient r_{ek} is only applied to the second layer by the ocean floor. $\hat{q}$ and $\hat{\psi}$ are related to each other via

$$\left(\mathbf{M} - \kappa^2\mathbf{I}\right)\cdot\begin{bmatrix}\hat{\psi}_1\\ \hat{\psi}_2\end{bmatrix} = \begin{bmatrix}\hat{q}_1\\ \hat{q}_2\end{bmatrix}, \text{where } \mathbf{M} = \begin{bmatrix}-\frac{f_0^2}{g'H_1} & \frac{f_0^2}{g'H_1}\\ \frac{f_0^2}{g'H_2} & -\frac{f_0^2}{g'H_2}\end{bmatrix}, \tag{3}$$

such that either q or ψ can independently identify the state of the system.

The model is solved pseudospectrally (Fox & Orszag, 1973) through inverting the velocity field and PV to real space, calculating the Jacobian using real-space PV fluxes, and transforming back to spectral space. The scale-selective dissipation (ssd) takes the form of an exponential filter which attenuates the last 1/3 of the spatial frequencies of all terms on the right-hand side of Eq. 2 (although for convenience it's written as an additive term). Similar to the 2/3 dealiasing rule (Orszag, 1971), this filtering scheme reduces aliasing errors in the same range of scales, but additionally provides numerical dissipation. The energetic contribution from the ssd term is relatively small, which is important for simulations of quasi-2D turbulence (Thuburn et al., 2014).

2.2 Model setup

We configure the model with a doubly-periodic square domain with a size of $L = 10^6\,m$, a flat topography, and a total depth of $H = H_1 + H_2 = 2500\,m$. There is a fixed mean zonal velocity shear of $\Delta U = U_1 - U_2 = 0.025\,m/s$ ($U_1 = 0.025\,m/s$, $U_2 = 0$) between the upper and lower layers. We set the deformation radius to $r_d = 15000\,m$, and r_d is related to f_0^2/g' via $r_d^2 = \frac{g'}{f_0^2}\frac{H_1 H_2}{H}$.

We select the model's grid size, Δx, in relation to the deformation radius, which is the characteristic scale for baroclinic instability and mesoscale turbulence. To resolve mesoscale eddies, one needs to ensure that $r_d/\Delta x$ is greater than 2 (Hallberg, 2013). With $r_d = 15000\,m$, if we choose a 256×256 grid where $\Delta x_{hires} = L/256 = 3906.25\,m$, then $r_d/\Delta x_{hires} = 3.84$, and mesoscale turbulence should be well-resolved; if we choose $\Delta x_{lores} = L/64 = 15625\,m$ such that $r_d/\Delta x_{lores} = 0.96$, we expect that the simulation is unrealistic. In such configuration, we would need to find a parameterization that acts at that resolution to replace the missing turbulent physics. We hereby refer simulations with a grid of 256×256 as "Highres", and simulations with a grid of 64×64 as "Lores". All high-resolution simulations are on a 256×256 grid with numerical timestep $\Delta t = 1\,h$, and all low-resolution simulations are on a 64×64 grid with $\Delta t = 2\,h$.

We consider two distinguishable flow regimes on which generalization properties of parameterizations can be tested:

- **Eddy configuration**: $H_1/H_2 = 0.25$, $r_{ek} = 5.787\times10^{-7}\,s^{-1}$, $\beta = 1.5\times10^{-11}\,(ms)^{-1}$,
- **Jet configuration**: $H_1/H_2 = 0.1$, $r_{ek} = 7\times10^{-8}\,s^{-1}$, $\beta = 10^{-11}\,(ms)^{-1}$

These configurations result in qualitatively different flow structure after running the simulation for 10 years, shown in Figure 1.

2.3 Diagnostics

The physical characteristics of QG systems can be qualitatively represented by various diagnostics such as energy and enstrophy spectra, total kinetic energy and enstrophy, and a spectral energy budget. Further, we also use these diagnostics quantitatively to define difference and similarity metrics and compare the performance across different SGS models in section 4.

Resolution has a strong impact on these diagnostics. In Figure 2, we show quasi-steady state statistics and spectra from simulations run at multiple resolutions (48×48, 64×64, 128×128, and 256×256 grids). The two higher-resolution simulations ($L/\Delta x \geq 128$) show similar behavior, indicating near-convergence of the statistical characteristics over the wavenumber band containing most of the kinetic energy of mesoscale eddies. The two lower-resolution simulations ($L/\Delta x \leq 64$) show significant differences due to insufficiently-resolved turbulent features which affect the flow at all scales.

To identify the energy pathway of the flow, we show spectral fluxes of different terms in the two-layer QG system. Let $E(k,l)$ denote the total spectral energy density of the two-layer system, we have

$$\begin{aligned}\frac{\partial E(k,l)}{\partial t} = &\sum_{m=1}^{2}\frac{H_m}{H}\mathbb{R}\left[\hat{\psi}_m^*\hat{\mathsf{J}}(\psi_m,\nabla^2\psi_m)\right] + \frac{f_0^2}{g'H}\mathbb{R}\left[(\hat{\psi}_1^*-\hat{\psi}_2^*)\hat{\mathsf{J}}(\psi_1,\psi_2)\right]\\ &+\frac{f_0^2}{g'H}k\Delta U\mathbb{R}\left[j\hat{\psi}_1^*\hat{\psi}_2\right] - \frac{H_2}{H}r_{ek}\kappa^2|\hat{\psi}_2|^2,\end{aligned} \tag{4}$$

where * denotes complex conjugate, $\mathbb{R}$ denotes real part, j is the imaginary unit, and the terms on the right-hand side are the spectral contributions from kinetic energy flux (KE

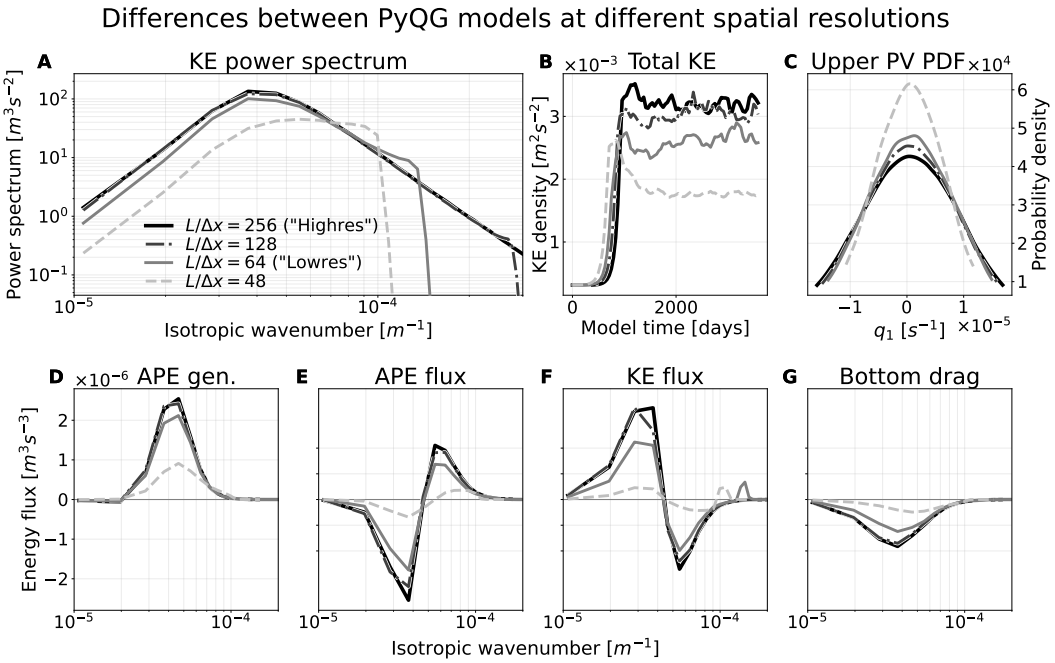

Figure 2. Comparison of time-averaged kinetic energy power spectra summed over fluid layers (A), time-series of total kinetic energy, (B), spatially flattened probability distribution of upper layer PV (C), and spectral energy flux terms (D-G) for eddy configuration simulations at multiple horizontal resolutions: $L/\Delta x$=256, $L/\Delta x$=128, $L/\Delta x$=64, and $L/\Delta x$=48. Higher-resolution simulations ($L/\Delta x \geq 128$) converge, while lower-resolution simulations ($L/\Delta x \leq 64$) differ from each other and from the higher-resolution simulations.

flux), available potential energy flux (APE flux), available potential energy generation (APE gen), and bottom drag, respectively.

The lower row of Figure 2 demonstrates the typical energy cycle in QG turbulence (Salmon, 1980; Vallis, 2017): the potential energy of fluctuations is extracted from the prescribed mean flow (APE gen) and cascades toward small scales up to the deformation radius (APE flux) where it is transferred to kinetic energy due to baroclinic instability (not shown). The kinetic energy then flows back to large scales following the inverse energy cascade (KE flux), where it is ultimately dissipated by friction (bottom drag).

The coarse resolution models ($L/\Delta x \leq 64$) poorly resolve the formation of mesoscale eddies due to baroclinic instability and their enlargement due to the inverse energy cascade (Zanna et al., 2020), leading to underestimated extraction of energy from the mean flow and a breakdown in the energy cycle.

A promising approach to avoid this breakdown is to supplement the resolved kinetic energy flux with a so-called "backscatter" parameterization (Jansen & Held, 2014; Porta Mana & Zanna, 2014) which energize eddies in large scales. We believe that efficient subgrid parameterizations should simulate backscatter at eddy permitting resolution, but also other processes that may matter, such as dissipation. In this paper we do not study precisely what physics are parameterized with data-driven subgrid models, but instead quantify how they influence the resolved energy cycle.

3 Diagnosing Subgrid Forcing

The goal of our work is to learn models that, given only low-resolution inputs, can predict the subgrid forcing missing from a low-resolution QG model. To do that, we need to first quantify subgrid forcing, which is generally done by filtering and coarse-graining high-resolution simulations (sometimes under the implicit assumption that coarsened high-resolution data will have a similar enough distribution to low-resolution data that the same data-driven parameterizations will work for both). We use $\overline{(\,)}$ to denote a generic filtering and coarse-graining operator. However, the choice of filtering and coarse-graining and the subgrid forcing terms are not uniquely defined, so we describe different options here.

In Sections 3.1 and 3.2, we present several options for how to define subgrid terms in their continuous forms, and in Section 3.3 we discuss their contribution to the energy budget. In Section 3.4, we describe the different filtering and coarse-graining options applied in this work.

3.1 Subgrid forcing of potential vorticity

We consider three different definitions of subgrid PV forcing for each fluid layer of the QG model: a total tendency, which is computed online as the residual between the low-res and high-res simulation (e.g., P. S. Berloff (2005) and Brenowitz and Bretherton (2018)); the subgrid forcing due to nonlinear advection; and the subgrid flux divergence forcing.

3.1.1 Total tendency (nonlinear advection and numerical dissipation)

Let ∂_t^H and ∂_t^L denote tendency functions from the high- and low-resolution models, respectively (dropping subscripts referring to the model layer for simplicity). For any given high-resolution q, we can express its total subgrid forcing (Porta Mana & Zanna, 2014; Kent et al., 2016; P. Berloff et al., 2021; Shevchenko & Berloff, 2021) due to the differences between the high- and low-resolution models with respect to $\overline{(\,)}$ as

$$S_{q_{tot}} = \overline{\partial_t^H q} - \partial_t^L \overline{q}. \tag{5}$$

We compute this quantity by setting the initial conditions of the high- and low-resolution models to be q and $\overline{q}$, respectively, taking a single step forward with equal Δt, and subtract the tendency of the low-resolution model from the filtered and coarse-grained tendency of the high-resolution model.

3.1.2 Subgrid tendency due to nonlinear advection

Another commonly used definition of subgrid forcing considers the unresolved nonlinear advection (Bolton & Zanna, 2019; Beck et al., 2019; Maulik et al., 2019; Xie et al., 2020; Zanna & Bolton, 2020; Guillaumin & Zanna, 2021; Guan et al., 2022), which can be expressed as

$$\begin{aligned} S_q &= \overline{(\mathbf{u}\cdot\nabla)q} - (\overline{\mathbf{u}}\cdot\overline{\nabla})\overline{q}, \\ &= \overline{\left(u\frac{\partial q}{\partial x} + v\frac{\partial q}{\partial y}\right)} - \left(\overline{u}\frac{\partial \overline{q}}{\partial x} + \overline{v}\frac{\partial \overline{q}}{\partial y}\right), \end{aligned} \tag{6}$$

where $(\overline{\mathbf{u}}\cdot\overline{\nabla})$ denotes the advection operator defined on the coarse grid. Following Grooms et al. (2013) and Porta Mana and Zanna (2014), we define the filtered and coarsened velocity $\overline{\mathbf{u}}$ by inverting the filtered and coarsened potential vorticity $\hat{\overline{q}}$ to $\hat{\overline{\psi}}$ using Eq. 3, multiplying $\hat{\overline{\psi}}$ by ik and il, and applying an inverse Fast Fourier Transform (FFT) to obtain $\hat{\overline{u}}$ and $\hat{\overline{v}}$, respectively.

3.1.3 Flux divergence subgrid tendency.

One difficulty in parameterizing subgrid forcing is that naive ML parameterizations may not obey conservation laws, e.g. for momentum and vorticity. Many physical parameterizations are formulated as divergences of fluxes to satisfy conservation laws by the divergence theorem. Ideally, it should be possible to learn ML parameterizations which behave similarly. One approach is to train ML models to predict subgrid forcing (e.g. S_q) but incorporate a numerical divergence operation into their architectures (e.g. as the final layer of a neural network, see Zanna and Bolton (2020)). Another is to diagnose a different quantity whose divergence equals the subgrid forcing (Pawar et al., 2020; Stoffer et al., 2021; Yuval et al., 2021), train ML models to predict this quantity directly, and compute divergences outside the model as part of the parameterization.

To enable experimentation with this second approach, we include an approximate version of such a quantity, which we call a "subgrid flux"

$$\boldsymbol{\phi}_q = \overline{\mathbf{u}q} - \bar{\mathbf{u}}\bar{q}. \tag{7}$$

Under the assumption that the flow is incompressible (i.e. that $\nabla \cdot \mathbf{u} \approx \overline{\nabla} \cdot \bar{\mathbf{u}} \approx 0$) and that differentiation commutes with filtering and coarsening,

$$\begin{aligned}
\overline{\nabla} \cdot \boldsymbol{\phi}_q &= \overline{\nabla} \cdot (\overline{\mathbf{u}q} - (\bar{\mathbf{u}})(\bar{q})) \\
&\approx \overline{\nabla \cdot (\mathbf{u}q)} - \overline{\nabla} \cdot (\bar{\mathbf{u}}\bar{q}) \\
&= \overline{(\nabla \cdot \mathbf{u})q + (\mathbf{u} \cdot \nabla)q} - (\overline{\nabla} \cdot \bar{\mathbf{u}})\bar{q} - (\bar{\mathbf{u}} \cdot \overline{\nabla})\bar{q} \\
&\approx \overline{(\mathbf{u} \cdot \nabla)q} - (\bar{\mathbf{u}} \cdot \overline{\nabla})\bar{q} = S_q
\end{aligned}$$

In practice, these three formulations (S_q, $S_{q_{tot}}$, and $\overline{\nabla} \cdot \phi_q$) are always highly correlated and often nearly identical, but the exact value of this correlation (especially for $\overline{\nabla} \cdot \phi_q$ vs. the others) can range from $>1-10^{-14}$ to <0.75 depending on the layer, timestep, configuration, and especially the filtering and coarse-graining operator (Section 3.4).

3.2 Subgrid forcing of velocity

Realistic ocean models use velocity as their prognostic variable with temperature and salinity, rather than PV. Here, we define momentum flux which is later related to the subgrid PV forcing.

The advection-based subgrid momentum forcing is given by

$$\mathbf{S}_\mathbf{u} = \overline{(\mathbf{u} \cdot \nabla)\mathbf{u}} - (\bar{\mathbf{u}} \cdot \overline{\nabla})\bar{\mathbf{u}}. \tag{8}$$

Analogous to Equation 7, We also define momentum subgrid flux terms as

$$\begin{aligned}
\boldsymbol{\phi}_u &= \overline{\mathbf{u}u} - \bar{\mathbf{u}}\,\bar{u}, \\
\boldsymbol{\phi}_v &= \overline{\mathbf{u}v} - \bar{\mathbf{u}}\,\bar{v}.
\end{aligned} \tag{9}$$

Note that the y-component of $\boldsymbol{\phi}_u$ is equal to the x-component of $\boldsymbol{\phi}_v$. We use $\boldsymbol{\Phi}_\mathbf{u} = (\boldsymbol{\phi}_u, \boldsymbol{\phi}_v)$ to denote the matrix of all four terms.

Given that the PV flux is composed of two parts: the relative vorticity flux and the buoyancy (or thickness) flux (Vallis, 2017; Killworth, 1997), we note that the relative vorticity flux is related to the momentum flux via the curl operator (Vallis, 2017). In our simulations, we update the PV tendency with the curl of subgrid momentum forcing, i.e. $\mathrm{curl}(S_u, S_v) = \partial_x S_v - \partial_y S_u$, which serves as a momentum parameterization in QG equations. Note that $\mathrm{curl}(S_u, S_v)$ is different from S_q when obtained from the respective coarse-grained fluxes: the two terms are sometimes uncorrelated or even anti-correlated, and the correlations range from +0.2 to -0.4 depending on the filtering and coarse-graining operator.

3.3 Contribution of forcing to diagnostics

Similar to Eq. 4, we derive the spectral contribution of subgrid-scale forcing towards total energy

$$\left(\frac{\partial E(k,l)}{\partial t}\right)^{\text{sub}} = -\frac{1}{H}\sum_{m=1}^{2} H_m \mathbb{R}\left[\hat{\psi}_m^* \hat{S}_{qm}\right], \tag{10}$$

where $\hat{S}_{qm}$ denotes the spectral PV tendency induced by the SGS model (the curl of velocity forcing if parameterizing momentum tendencies) in the mth layer. This equation states that the total tendency induced by the subgrid term can be written as the projection of subgrid tendency onto the streamfunction in each layer.

3.4 Coarse-graining and filtering

We are using a combination of filtering and coarse-graining to diagnose the subgrid forcing. There are a number of possible ways to filter and coarse-grain simulations. Filtering can be identified by various convolutional kernels (top-hat, Gaussian, e.g., Sagaut (2006)), which can be approximated on a given mesh with quadrature rules (Xie et al., 2020; Guillaumin & Zanna, 2021), polynomials based on Laplacian operator (Sagaut & Grohens, 1999; Grooms et al., 2021) or applied in spectral space (Guan et al., 2022). Coarse-graining methods include spectral truncation (Thuburn et al., 2014), averages over boxes (Porta Mana & Zanna, 2014; Beck & Kurz, 2021) or subsampling (Xie et al., 2020, i.e. selection of every $K's$ point). Here, rather than focusing on one method for filtering and coarse-graining, we examine the sensitivity of our results to three different operators (referred to as "Operator 1", "Operator 2", and "Operator 3") for diagnosing the subgrid forcing in our simulations: two different filters in spectral space, and one filter in real space.

Given that `pyqg` is a pseudo-spectral model, it is natural to use spectral methods to perform coarse-graining and filtering, and for data generation, we first coarse-grain and then filter. Coarse-graining is done by coarse-graining the simulation by a factor of K, or more precisely, truncating the set of spatial modes of $\hat{q}$ by only keeping the first $1/K$. For example, in the case of going from resolution 256×256 to 64×64, we start with a $\hat{q}$ with 128 modes and only keep the first 32.

Spectral filtering generally consists of applying selective decay that reduces the strength of the highest frequencies, whereas low-frequency components are mostly retained after truncation. Here, we use two different filtering methods.

3.4.1 Operator 1: spectral truncation, sharp filter

The first option is implemented by applying a sharp filter which leaves small wavenumbers unchanged but attenuates wavenumbers above a cutoff threshold $\kappa^c \equiv 2/3$ of the low-resolution model's Nyquist frequency, using a quadruple exponential:

$$\hat{\overline{q}}_\kappa = \begin{cases} \hat{q}_\kappa, & \kappa < \kappa^c \\ \hat{q}_\kappa * e^{-23.6(\kappa-\kappa^c)^4 \Delta x_{\text{lores}}^4}, & \kappa \geq \kappa^c. \end{cases} \tag{11}$$

This filter is used internally by `pyqg` to implement numerical dissipation, so in some sense this is the most conservative choice of filter possible (i.e. closest to not filtering at all). We use "Operator 1" to refer to spectral truncation followed by the application of this filter.

3.4.2 Operator 2: spectral truncation, softer Gaussian filter

The second spectral filtering option considered ("Operator 2") is to instead apply the following Gaussian filter to all remaining modes:

$$\hat{\bar{q}}_\kappa = \hat{q}_\kappa * e^{-\kappa^2(2\Delta x_{\text{lores}})^2/24} \tag{12}$$

This choice of filter is based on Guan et al. (2022) and Pope (2000). According to the definition of the filter width given by Lund (1997), this filter is twice as large as the grid size of the coarse model.

3.4.3 Operator 3: diffusion-based filtering, real-space coarsening

Finally, to mimic the procedure necessary for ocean models which are not run in spectral space, we convert our output to a Cartesian grid and apply `GCM-Filters` (Grooms et al., 2021; Loose et al., 2022), a recent filtering method which approximates the spectral transfer function of Gaussian filter with polynomials based on the Laplacian diffusion operator. We then coarse-grain the output in real space. To reduce the resolution by a factor of K, we average the input field over non-overlapping boxes of $K \times K$ points. We call this procedure "Operator 3."

A comparison of the effects of these different filtering and coarse-graining operators on potential vorticity and its subgrid forcing is shown in Figures 3 and 4.

3.5 Comments regarding notation

We use superscripts (1), (2), and (3) (for Operators 1, 2, and 3, respectively) to describe subgrid forcing computed with each operator. For example, $S_q^{(1)}$ signifies the subgrid tendency due to nonlinear advection diagnosed by Equation 6 and computed with the operator from Section 3.4.1, while $\mathbf{\Phi}_{\mathbf{u}}^{(3)}$ signifies the tensor of velocity subgrid fluxes diagnosed by Equation 9 and computed with the operator from Section 3.4.3.

In addition, we use $\overline{(\,)}$ to refer to all low-resolution variables, whether coarse-grained from a high-resolution simulation or natively from a low-resolution simulation. The reason is that we evaluate parameterizations offline over filtered and coarse-grained high resolution variables, but evaluate parameterizations online over low-resolution variables. Although these variables can be different in various respects (e.g., may be differently distributed), when learning data-driven parameterizations from subgrid forcing data collected offline, we necessarily assume that one will generalize to the other. Though this is an assumption we explicitly test in online evaluation.

In the remaining text, we also simplify $\overline{\nabla}$ to just ∇ for conciseness. It should be treated as $\overline{\nabla}$ when applied to a coarsened or low-resolution variable.

4 Metrics

In the sections that follow, we evaluate a large number of parameterizations on data generated with different operators and forcing formulations, as well as different inputs and architectures. Given that the models are too many to manually inspect. Instead, we define a rich set of metrics to quantify their performance. We start by defining metrics which can be evaluated offline, i.e. on held-out testing sets of subgrid forcing data, and then define a large number of online metrics that quantify the performance of a parameterization implemented into a free-running simulation.

4.1 Offline

Offline metrics quantify the parameterization's skill at predicting its intended target. For each fluid layer, we consider:

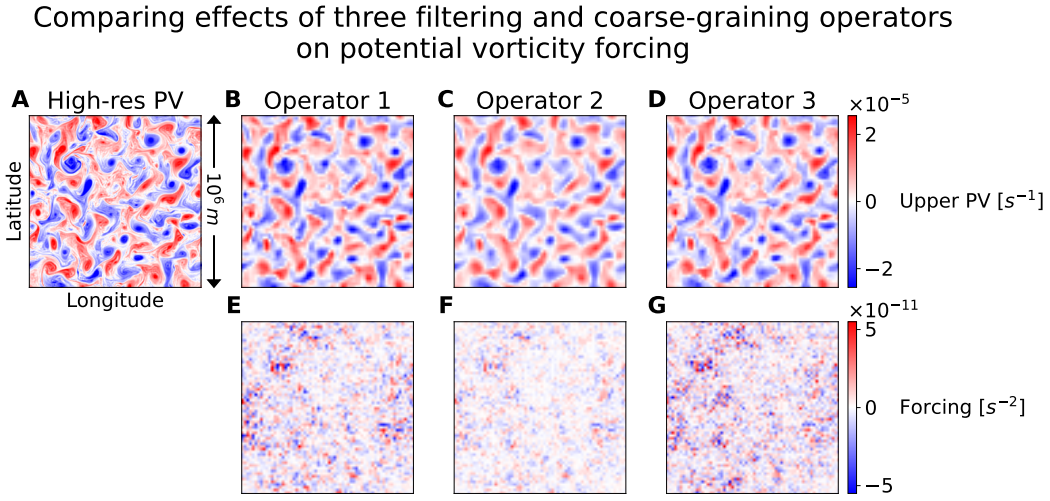

Figure 3. Comparison of the effects of three different methods of filtering and coarse-graining 256×256 eddy configuration initial states (A) to 64×64 (B-D), along with resulting forcing terms S_q, defined in Eq. 6 (E-G). Operators 1 (B,E) and 2 (C,F) truncate Fourier modes and apply sharp and soft spectral filters, respectively, while Operator 3 (D,G) applies diffusion-based filtering and averaging in real space. See Section 3.4 for operator definitions and Figure 4 for comparisons of associated spectral properties.

1. The coefficient of determination (R^2), $1 - \frac{\mathbb{E}\left[(S-\hat{S})^2\right]}{\mathbb{E}\left[(S-\mathbb{E}[S])^2\right]}$, which is 1 when predictions are perfect, 0 when predictions are no better than than always predicting the mean, and negative when worse than always predicting the mean.
2. Pearson correlation (ρ), $\frac{\mathrm{Cov}(S,\hat{S})}{\sigma_S \sigma_{\hat{S}}}$, where σ denotes the empirical standard deviation of a quantity over the dataset. This quantity is between -1 and 1 and can remain high even when R^2 is negative, e.g. if predictions are wrong by a large but consistent scaling factor.

These metrics are evaluated on held-out datasets of filtered and coarse-grained high-resolution simulations from both eddy and jet configurations. They can either be aggregated over time and space or expressed as functions of time or space. In addition, we visualize the power and energy redistribution spectra of the predicted subgrid forcing and compare them to the corresponding quantities for the ground-truth forcing.

4.2 Online

In contrast to offline metrics, we evaluate online metrics by initializing a new QG simulation at low resolution and, at every time step, passing its state to the parameterization and adding the parameterization's output to the PV tendency. The distribution of these low-resolution states may therefore be different, but by analyzing the ultimate results and testing for various forms of consistency with high-resolution results (i.e. online metrics), we can evaluate whether the parameterization is effective.

To compute such online metrics, we first run 5 parameterized low-resolution simulations for 10 years in both eddy and jet configurations from different random initial conditions, saving all state variables and diagnostics described in Section 2.3. We then compute distance metrics between the (statistical and spectral) distributions of these variables and those from 5 simulations run at high resolution in corresponding configura-

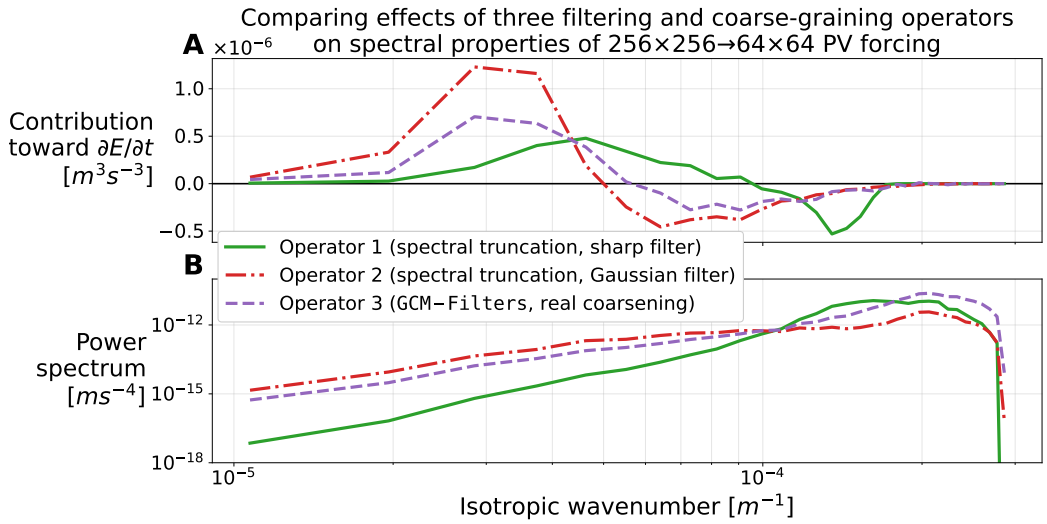

Figure 4. Energy redistribution, Eq. 10 (A) and power spectra (B) of S_q by filtering and coarse-graining operator (computed on eddy configuration data, averaged over time, and summed across layers). Each operator produces forcing which redistributes energy differently across scales with different spatial spectra. See Figure 3 for comparisons of forcing snapshots.

tions. Finally, we normalize these distances by the corresponding metrics for unparameterized low-resolution simulations to obtain more interpretable similarity scores.

4.2.1 Differences between time-averaged power spectra and fluxes

Some of the most important characteristics of simulations are how energy and enstrophy distribute and move across scales, which we measure using power spectra and the spectral flux diagnostics described in Section 2.3. Ideally, a parameterized simulation should match a high-resolution simulation with respect to all such quantities.

For both power spectra and fluxes, we compute a total root mean squared difference between curves f:

$$\texttt{spectral\_diff}(\texttt{sim1}, \texttt{sim2}; f) \equiv \sqrt{\frac{1}{|\mathcal{K}|} \sum_{k \in \mathcal{K}} \left(f_{\texttt{sim1}}(k) - f_{\texttt{sim2}}(k)\right)^2} \tag{13}$$

where $\mathcal{K}$ is a suitably chosen set of isotropic wavenumbers common to both simulations. In our case, $\mathcal{K}$ is evenly distributed in log space and is up to 2/3 of the Nyquist frequency of the low-resolution simulation ($\approx 1.07 \times 10^{-5} m^{-1}$). We compute this metric for the energy and enstrophy power spectra in each layer and for the spectral energy fluxes (KE flux, APE flux, APE generation, and bottom drag), yielding a total of 8 metrics. The contribution of parameterizations towards total energy (Eq. 10) is added onto the KE flux term for parameterized low-resolution simulations. An illustration of this kind of distance metric is shown in Figure 5A.

4.2.2 Differences between spatially flattened probability distributions

We also consider differences between the empirical distributions of various quantities in different simulations at the end of the simulation, which we measure with earth

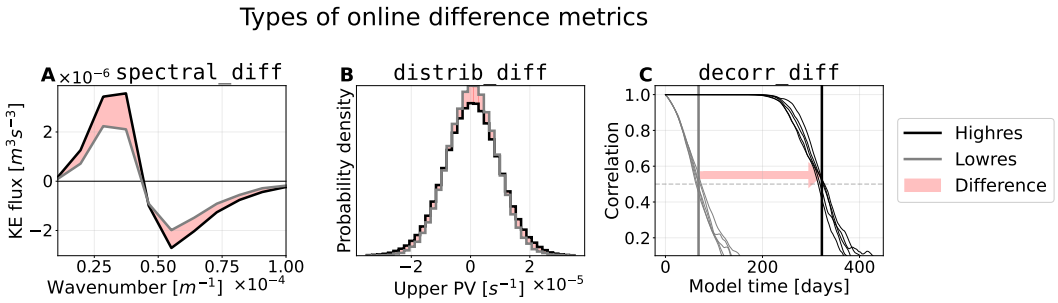

Figure 5. Illustration of the three types of difference metrics defined in Section 4.2. `spectral_diff`s (A) compute the RMSE between different quantities summed over isotropic wavenumber κ. `distrib_diff`s (B) compute the earth mover's distance between the marginal distributions of variables at the end of the simulation. `decorr_diff` (C) estimates how much faster a given simulation diverges from a high-resolution simulation when starting from the same random initial state.

mover's distance (or Wasserstein distance with $p = 1$):

$$\texttt{distrib\_diff}(\texttt{sim1}, \texttt{sim2}; f) \equiv \int_{-\infty}^{\infty} \left| P_{\texttt{sim1}}(f \leq x) - P_{\texttt{sim2}}(f \leq x) \right| \mathrm{d}x, \tag{14}$$

where $P_{\texttt{sim}}(f \leq x)$ is a cumulative distribution function of quantity f in a given `sim`ulation. If we imagine the two probability density functions as mounds of earth, this metric corresponds to the minimum amount of work required to move all the mass from one mound to the other. For 1-dimensional distributions, it reduces to the integral of the difference in each cumulative distribution function (which we approximate empirically). We compute these differences for the quasi-steady-state distributions (marginalized over space and at the final timestep) of u, v, q, the kinetic energy density $(u^2{+}v^2)/2$, and enstrophy $\mathrm{curl}^2(\mathbf{u})/2$ at each layer (giving us 10 total metrics).

Note that when comparing low-resolution to high-resolution metrics, we are comparing the distributions of, e.g., u and $\overline{u}$, so histograms are appropriately normalized. An illustration of this kind of comparison is shown in Figure 5B, though for brevity we show only the difference in the integrals of PDFs rather than the integrals of the corresponding CDFs (which gives the exact value of `distrib_diff`).

4.2.3 Differences in decorrelation times

The previous metrics consider whether *aggregate* simulation statistics match those of high resolution simulations. However, we could maximize these metrics with a trivial parameterization that samples $\overline{q}_{t+1}$ randomly from a previously-run filtered and coarsened high-resolution simulation. Such a parameterization would be completely unrealistic, since subsequent states $\overline{q}_t$ and $\overline{q}_{t+1}$ would be completely unrelated. Arguably, parameterizations should improve the realism not just of long-term statistics, but also of short-term trajectories.

We attempt to measure the short-term realism by defining a "decorrelation time" metric, i.e. minimum time t to achieve correlation δ from above, averaged over ensemble of initial conditions q_0 and their perturbations ϵ

$$\texttt{decorr\_time}(\texttt{sim1}, \texttt{sim2}) \equiv \mathbb{E}_{q_0,\epsilon}\left[\min_t \left\{ t : \mathrm{Corr}\left(q^{(t)}_{\texttt{sim1}}(q_0),\, q^{(t)}_{\texttt{sim2}}(q_0{+}\epsilon)\right) \leq \delta \right\}\right] \tag{15}$$

where each $q^{(t)}_{\texttt{sim}}(q_0)$ denotes the potential vorticity for the given `sim`ulation integrated for time t starting from an initial condition q_0 (sampled from the quasi-steady state),

ϵ is a small independent Gaussian perturbation with standard deviation 10^{-10}, and $\delta = 0.5$. When `sim1` and `sim2` have the same dimensionality, Corr denotes the simple Pearson correlation, but when they have different dimensionalities (i.e. if `sim1` is higher resolution), we compute the correlation after filtering and coarse-graining the higher-resolution simulation to the resolution of the other using the sharp spectral filter from Equation 11. We approximate expectations $\mathbb{E}_{q_0,\epsilon}$ using empirical averages over 5 random samples of q_0, ϵ, and we use the same random high-resolution q_0s for all low-resolution models so that correlation trends for different low-resolution models can be paired. Note that many of these choices are arbitrary, and could be modified for future studies.

With decorrelation time now defined, we can then compare the expected time it takes for one type of simulation to fall out of sync with another, vs. the expected time it takes for one simulation to fall out of sync with a perturbed version of itself:

$$\texttt{decorr\_diff}(\texttt{sim1}, \texttt{sim2}) \equiv \texttt{decorr\_time}(\texttt{sim1}, \texttt{sim1}) - \texttt{decorr\_time}(\texttt{sim1}, \texttt{sim2}), \quad (16)$$

In our study, `sim1` is a high-resolution simulation, which stays correlated with a perturbed version of itself for a relatively long time, while `sim2` will be a low-resolution simulation. With the eddy configuration, the correlation of high-resolution simulations stay above 0.5 for about 1 year, whereas unparameterized low-resolution simulations remain >0.5 correlated for about 2 months, leading to a `decorr_diff` of 10 months. For a parameterized low-resolution simulation, we hope that `decorr_diff` is lower (e.g., 8 months), indicating that its short-term evolution is more consistent with that of the high-resolution model. An illustration of `decorr_diff` is shown in Figure 5C.

4.2.4 From difference to similarity

One issue with defining such a variety of distance metrics is that they become difficult to compare especially when they have different units. However, for any particular metric, what we care about is not its actual value but whether it is smaller for parameterized simulations (vis-à-vis Highres simulations) than for low-res simulations. To that end, we re-express our distance metrics as similarity scores that quantify how much closer parameterized models are to high-res than to low-res:

$$\text{Similarity}(\text{param}, \text{high-res}; \texttt{diff}) \equiv 1 - \frac{\texttt{diff}(\text{param}, \text{high-res})}{\texttt{diff}(\text{low-res}, \text{high-res})} \quad (17)$$

This similarity score is ≈1 if the parameterized model's distance to high-res is much smaller than that of the low-res model (and exactly 1 for the high-res model), ≈0 if this distance is approximately equal to that of the low-res model (and exactly 0 for the low-res model), and <0 if the distance is larger. An example is shown in Figure 6. We also include a validation of the consistency of these scores with respect to high- and low-resolution simulations generated with different random initial conditions in Figure E10. In general, we evaluate our online results using similarity scores.

4.3 Experimental Setup

With our datasets and metrics now defined, we now describe our experiments to learn and evaluate parameterizations. In total, we test 148 parameterizations—105 fully convolutional neural networks (FCNNs) trained with different dataset design decisions (described in Section 5), a hybrid linear and symbolic regression method using genetic programming (described in Section 6), and 42 different parameter settings spread over three baseline physical parameterizations: symbolic regression from Zanna and Bolton (2020), backscatter from Jansen and Held (2014), and Smagorinsky (1963) (described in Appendix A).

The trained parameterizations are evaluated offline but are also implemented into the coarse resolution simulation with 64×64 horizontal resolution for the online eval-

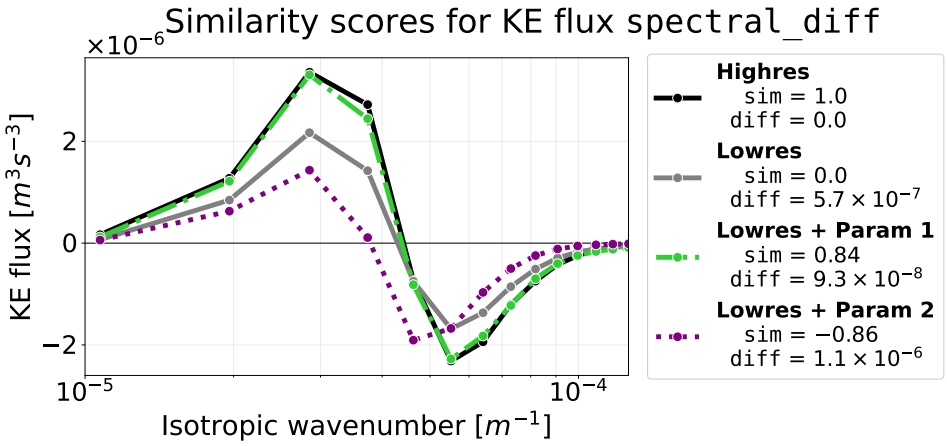

Figure 6. Example online similarity scores for two parameterizations corresponding to the `spectral_diff` of their KE flux terms with respect to high-res (as compared to low-res). In this example, the first parameterized model's KE flux curve is much closer to that of the high-res model than the low-res model, so its similarity is positive and close to 1 (though slightly lower than it might seem from visual inspection due to the logarithmic x-scale). The second parameterized model, on the other hand, is further away than low-res, so it receives a negative score.

uation. To simplify the discussion, we begin by describing these categories of parameterizations individually, along with some of the experimental results specific to those categories. We then compare performance across parameterization categories in Section 7.1.

Because we have multiple categories of parameterization (FCNN, genetic programming, and baselines) and multiple categories of online metric (spectral, distributional, and decorrelation time) with numerous individual parameterizations (148) and metrics (19) within each category, we will often simplify as follows. For each category of online metric, we summarize individual parameterization's scores by taking means (or medians and percentiles if visualizing variation). For each category of parameterization, we either show a distribution of these mean scores, or select individual parameterizations to highlight from the Pareto frontier of mean scores within each category, i.e. the set of parameterizations which maximize some linear combination of mean scores.

5 Convolutional Neural Network Parameterizations

We consider parameterizations implemented as fully convolutional neural networks (FCNNs) which output predictions for all x, y points simultaneously. Models receive input data at all points x, y, z in both layers (though we train separate models for each fluid layer to reduce memory cost during training), which allows them to be maximally flexible, and therefore useful for studying the effects of changing attributes of the dataset on best-case performance.

5.1 Dataset design choices

For our FCNN experiments, we are interested in how the structure of the dataset affects the ultimate performance. We train FCNNs to predict subgrid forcing diagnosed with each of the five forcing formulations (S_q, $S_{q_{tot}}$, ϕ_q, $S_{\mathbf{u}}$, $\mathbf{\Phi}_{\mathbf{u}}$, Section 3), and for each forcing formulation, we generate three FCNNs trained on datasets generated by each filtering and coarse-graining operator (Section 3.4). Finally, we also investigate the effect

of the choice of input variables we pass to the FCNN by testing every non-empty element of the power set of $\{\overline{q}, \overline{\mathbf{u}}, \nabla\overline{\mathbf{u}}\}$ (7 options in total, where $\nabla\overline{\mathbf{u}} = (\partial_x\overline{u}, \partial_x\overline{v}, \partial_y\overline{u}, \partial_y\overline{v})$). This gives us $5 \times 3 \times 7 = 105$ total options for constructing FCNN parameterizations.

Notation-wise, we refer to models trained on each option as, e.g., FCNN($\overline{q}, \overline{\mathbf{u}} \to S^{(2)}_{\mathbf{u}}$), which signifies an FCNN trained on the values of potential vorticity and velocity to estimate subgrid momentum forcing (Eq. 8) and computed with Operator 2 (spectral truncation + Gaussian filter, Section 3.4.2) .

5.2 Architectural details and constraints

Following Guillaumin and Zanna (2021), we train FCNNs with 8 fully convolutional layers (128 and 64 filters for the first two layers, and 32 thereafter), ReLU activations, batch normalization after all intermediate layers, and circular padding due to the periodicity of the domain. Each input variable at each fluid layer is passed in a separate input channel. The loss function is mean squared error (MSE), defined as $\mathbb{E}\left[(S - \hat{S})^2\right]$, where $\mathbb{E}$ denotes the expected value over a dataset and S is a generic prediction target. The FCNNs are trained for 50 epochs on a MSE loss evaluated over minibatches of 64 samples.

In preliminary experiments, we found that constraining the FCNNs' final output layers to have zero spatial mean when predicting S_q and $S_{\mathbf{u}}$ significantly improved stability. This is done within the FCNN architecture and not as a post-processing step. The constraint ensures that at each timestep, parameterizations redistribute but not increase or decrease the total potential vorticity. However, when predicting ϕ_q and $\mathbf{\Phi_u}$, we leave FCNNs unconstrained because we only apply predictions after taking their divergence.

Although the chosen architecture could be improved (e.g., by adopting the U-Net model of Ronneberger et al. (2015)), our goal is not to maximize the performance but to study its relationship with dataset design choices.

5.3 Sensitivity of FCNN performance to dataset design

We now present FCNN-specific results of how online performance varies with the dataset design choices described in Section 5.1. For each design choice, we constructed the corresponding eddy configuration training data, trained an FCNN parameterization, and evaluated it in both eddy and jet configuration, both offline and online. In each case, all simulations were numerically stable (the Courant-Friedrichs-Lewy condition was not violated). The stability is likely due to our architectural constraints (and perhaps the spectral numerical dissipation scheme of `pyqg`). However, performance in terms of similarity metrics varied greatly.

To visualize this variation, in Figure 7 we plot kernel density estimates (Rosenblatt, 1956) of conditional probability distributions of the mean `spectral_diff` similarity score (substituting Eq. 13 into Eq. 17) for different dataset design choices of filtering and coarse-graining operator, forcing formulation, and input variables. Specifically, these plots show the distribution of the average similarity score across KE power spectra, enstrophy power spectra, and energy budget terms over isotropic wavenumber, conditioned on different choices of filter and coarse-graining (Fig. 7A), targeted forcing formulation (Fig. 7B), and input variables (Fig. 7C). Probability density closer to 1 indicates better performance. Overall, we see higher spectral similarity scores for FCNNs trained on data generated with Operator 1 (spectral truncation with sharp filter) (Fig. 7A) and predicting forcing in terms of velocity rather than PV (Fig. 7B). The choice of input has a weaker impact on these scores (Fig. 7C), though simpler terms ($\overline{\mathbf{u}}$, $\nabla\overline{\mathbf{u}}$, or $\overline{q}$ alone) do slightly better. The same results hold for `distrib_diff` similarity (not shown), which is strongly correlated with `spectral_diff` (Figs 8).

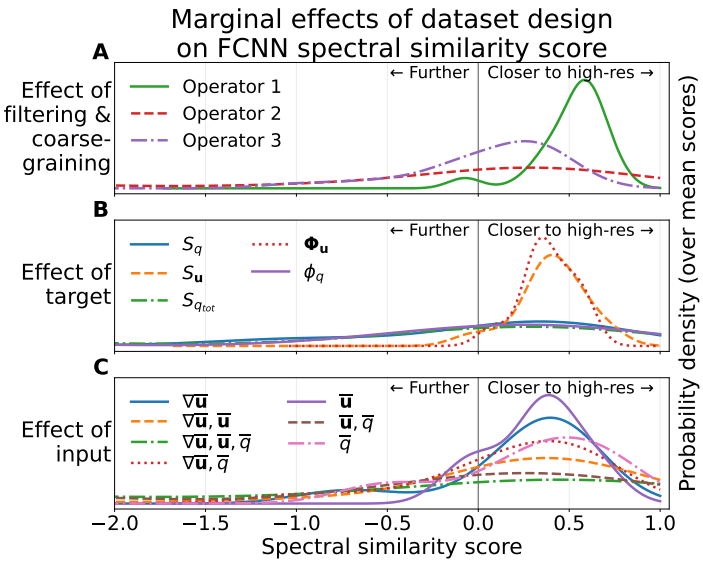

Figure 7. Visualizing the effects of different dataset design choices: A) filtering and coarse-graining operator, B) forcing formulation, C) input. We use the probability distributions of mean spectral similarity scores, conditioned on each design choice, and smoothed using kernel density estimation for visual clarity. Similarity score probability mass further to the right (past the 0 line, and towards 1) indicate that the corresponding difference metric was low compared to low-res, therefore indicating good online performance. The results suggest that marginally, similarity was highest along most metrics for parameterizations trained to use velocity (Panel C) to predict velocity-based subgrid forcing (Panel B) calculated with a sharp spectral filter (Panel A).

In addition, we can gain insights through analyzing specific models. If we look at the Pareto frontier of eddy-configuration distributional and spectral similarity across all our experiments (Fig. 8), we find that the only Pareto-optimal FCNN predicts $S^{(1)}_{q_{tot}}$, which is computed with Operator 1 but formulated in terms of PV rather than velocity. If we compare this FCNN to others which are identical except for the filtering and coarse-graining operator (Fig. E4) or forcing formulation (Fig. E5), we find again that the choice of operator continues to matter, but that the forcing formulation has much less effect as FCNNs predicting other forcing formulations with the same filtering and coarse-graining operator all have near-identical effects. Combining these individual results with the aggregate results of Fig. 7, our overall interpretation is that a) the choice of operator is the most important for online performance, and b) predicting velocity forcing ($S_{\mathbf{u}}$ or $\mathbf{\Phi_u}$) rather than PV forcing (S_q, $S_{q_{tot}}$, or ϕ_q) is not necessary for optimal performance, but may be more robust to variations in other suboptimal design choices (e.g. picking Operators 2 or 3). Operator 1 is more faithful to the numerics of the coarse-resolution model that we are using in the online evalaution (this is further supported by the lack of backscatter generated using ZB2020, see conclusions).

5.4 Relationship between offline and online FCNN metrics

Offline performance, measured using R^2, is strong for all design choices (see Fig. 9). If we condition the relationship between offline and online performance on the filtering and coarse-graining operator, we more positive relationships between offline and online performance when using Operator 1 (Fig. 10A, D), compared to low or negative correlations for Operators 2 and 3, where improved offline performance was associated with

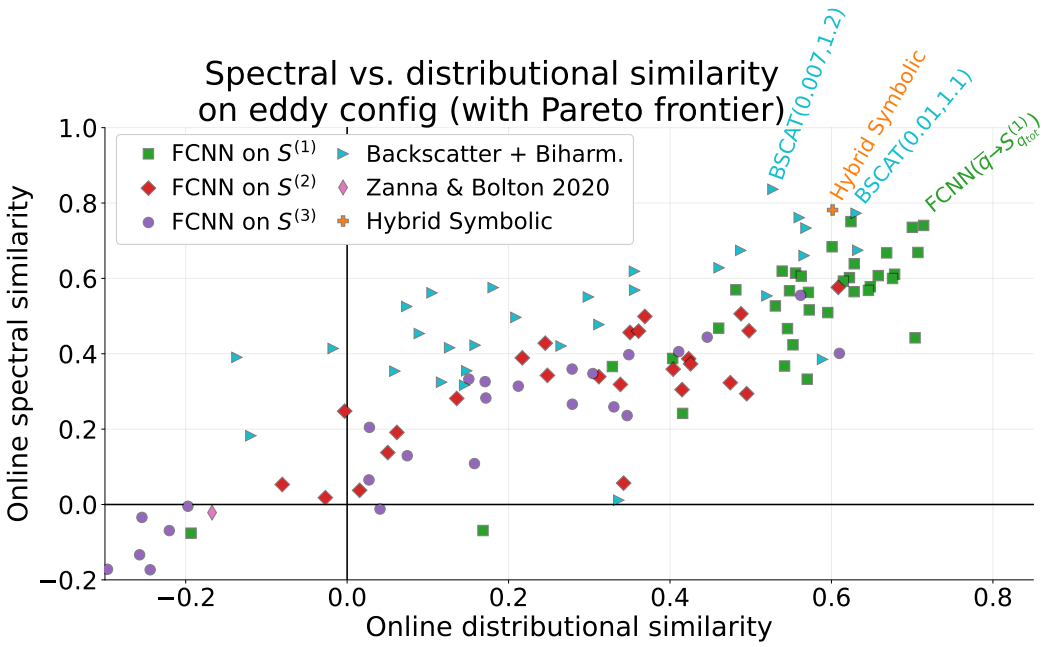

Figure 8. Mean eddy-configuration distributional and spectral similarity scores for many of the 148 parameterizations tested, with those defining the Pareto frontier shown with text (the runs with remaining parameterizations, including all Smagorinsky runs, have scores to the lower-left of the plot range.)

worse rather than better online performance. This result underscores the importance of not focusing too much on improving the offline performance of subgrid parameterizations without first demonstrating that such improvements lead to improvements in physical realism online.

5.5 Varying the evaluation target

Some studies measure the online performance of parameterized low-resolution models with respect to the filtered and coarse-grained version of high-resolution data (Beck et al., 2019; Xie et al., 2020; Guan et al., 2022), rather than the high-resolution simulation. Calculating similarity scores for coarse-resolution parametrized models relative to a coarsened and filtered high-resolution simulation increases the scores of top-performing FCNNs, if the parameterization and the target high-resolution models using the same operator (Figure 11). However, even when the target is coarsened with Operator 2 or 3, the actual scores of these models are significantly lower than in the case where the FCNN is trained on data generated by Operator 1. The FCNN trained on data generated by Operator 1 has the best overall spectral similarity score whether we perform the evaluation using the original high-resolution data or data coarsened with Operator 1. This result suggests that Operator 1 is more appropriate for computing subgrid forcing in this dataset in an absolute sense.

5.6 Feature importance for FCNNs

To explore the importance of individual features to our FCNN predictions, we look at snapshots of input gradients, or the partial derivatives of the model's output with respect to its inputs (Baehrens et al., 2010). Note that although there are many proposed

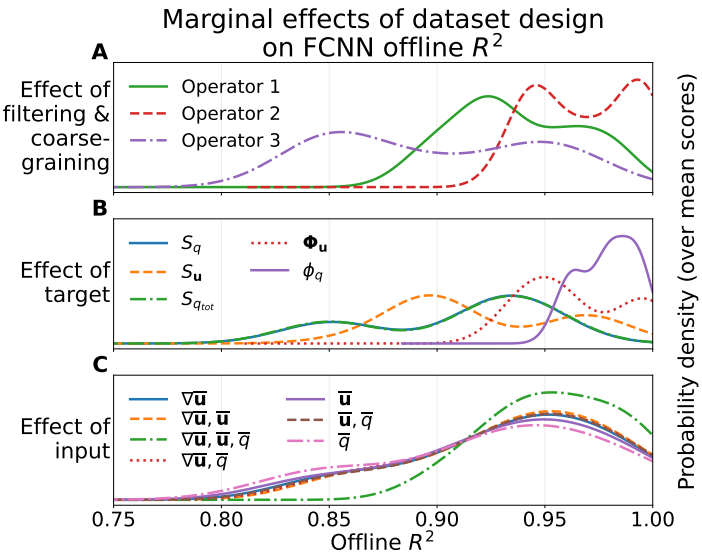

Figure 9. Offline R^2 scores by dataset design choice as in Figure 7, almost all of which achieve an R^2 of above 0.8 regardless of condition. The best models by offline R^2 are different from those in Figure 7.

methods for quantifying neural network input saliency (Springenberg et al., 2014; Bach et al., 2015), input gradients consistently pass sanity checks that have been developed to validate these methods, while many alternatives do not (Adebayo et al., 2018; Kindermans et al., 2019).

In Fig. 12, we show a snapshot of input gradients for the FCNN($\overline{q}\rightarrow S^{(1)}_{q_{tot}}$) at the center of the domain, which quantifies the sensitivity of its predictions at this particular location to its inputs. Although our FCNN architecture allows changes in the upper layer $\overline{q}$ to influence the lower layer prediction and vice-versa, this particular FCNN's input gradients are only large in magnitude for $\overline{q}$ in the same layer as the output, suggesting that it largely operates layer-wise.

Additionally, gradients were largest in magnitude around the spatial location of the output, suggesting that the model operates locally in the horizontal plane. However, we find that a radius of 5 pixels (4th-order operations) is needed to explain 50% of the gradients, and a radius of 9 pixels (8th-order operations) is needed to reach 95% (Figure 12 I). This suggests that symbolic parameterizations may need to be fairly non-local and high-order to mimic the behavior of FCNNs. We explore this in the next section.

6 Hybrid Linear and Symbolic Regression and Genetic Programming

In addition to opaque models such as neural networks and random forests, it is also possible to learn equations from data directly with symbolic regression (Koza, 1994).

Symbolic-regression based on running sparse linear regression on top of a manually constructed feature library has become popular and achieved impressive results in a number of applications (Brunton et al., 2016; Li et al., 2021). Zanna and Bolton (2020) (ZB2020 hereafter) learned an expression for the subgrid momentum forcing $\mathbf{S_u}$ with sparse Bayesian regression (see Eq. A7 in Appendix). They used data generated from an idealized primitive equation model, with Gaussian filtering. Here, using data from `pyqg` and a Gaussian filter (Operator 2) to calculate the same basis features as in ZB2020 (i.e., di-

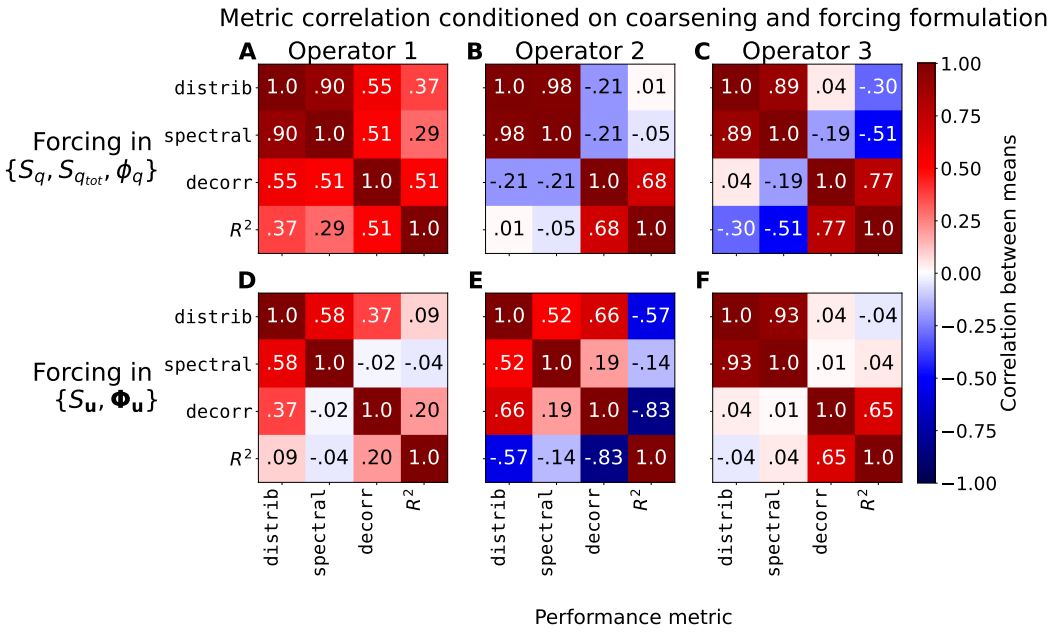

Figure 10. Correlation between FCNNs' mean scores in each metric group conditioned on the filtering and coarse graining operator (columns) and forcing formulation (rows) used to generate their training data. In most cases, distributional and spectral similarity are closely correlated. Correlations with offline R^2 tend to be negative or small, except for FCNNs trained to predict PV forcing variants computed with Operator 1 (A).

vergence, vorticity, stretching and deformation, their x- and y-derivatives, and all cross-multiples), we are able to re-discover Eq. A7 with a simple sparse linear regression algorithm.

However, sparse linear regression entails trade-offs between the size and expressiveness of the feature library and the complexity and cost of sparse regression, as discussed in Zanna and Bolton (2020). In the example above, our feature library has the initial basis features (4 elements), their first spatial derivatives (8 elements), and all cross-multiples of those initial features (144 elements). If we want to expand this library to consider successively higher-order derivatives (or more than just linear and quadratic multiples), then the number of different expressions we must evaluate for the whole dataset will grow exponentially. Additionally, many expressions will be highly correlated, which can prevent many sparse regression algorithms from converging (Hastie et al., 2015).

6.1 Hybrid genetic programming (GP)

An alternative approach for symbolic regression is genetic programming (GP), a classic approach in AI (Turing, 1950; Koza, 1994). In contrast to sparse regression, GP algorithms do not require an explicit feature library, simply a set of atomic features and a set of operations for combining them ensue. The GP algorithm then constructs arbitrarily deep expressions by successively applying operators to combine atomic and/or composite features in a randomized fashion, using evolutionary principles to guide a parallel search for an expression that parsimoniously fits the data.

More concretely, GP algorithms begin with a "population" of initially short and randomly-constructed programs. At each iteration ("generation"), programs are randomly

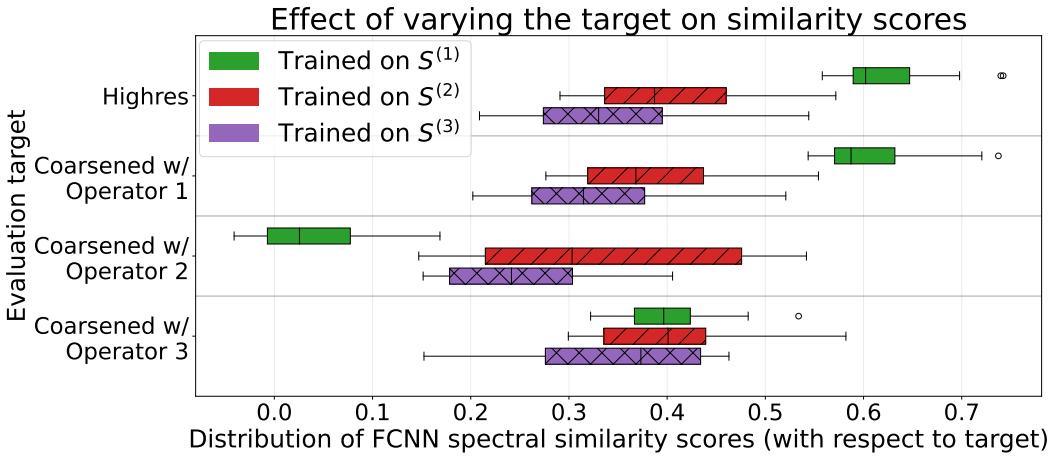

Figure 11. Boxplots showing distribution of `spectral_diff` similarity scores (across all FCNNs trained with each filtering and coarse-graining operator, e.g. $S^{(1)}$ for Operator 1, with any forcing formulation) when the definition of similarity is changed to be relative to a filtered and coarsened version of the high-resolution simulation (bottom three rows), rather than the original high resolution simulation (top row). Center line shows medians, colored bars show the interquartile range (middle 50% of the data), whiskers show positions of nearest points outside twice the interquartile range, and dots show outliers. In general, the relative performance of FCNNs improves when evaluating them against simulations coarsened with the same operator used in their training data. However, absolute performance is only high for FCNNs trained on data from Operator 1 (spectral truncation + maximally sharp filter).

culled, with probability inversely related to their relative performance on a "fitness" metric (see Algorithm 1). Programs that survive can then be randomly modified ("mutated") in a variety of ways, which can either lengthen or shorten them. This procedure is repeated for a configurable number of generations, after which the GP algorithm returns the best-performing program.

To implement genetic programming, we used the `gplearn` Python library (Stephens, 2019). We ran into several difficulties with its default implementation, primarily in its difficulty discovering linear combinations of terms with different orders of magnitude in the weights (constant ranges must be chosen beforehand, and are sampled randomly rather than optimized), as well as the lack of built-in support for spatial differential operators in program evolution. We defined custom `gplearn` functions for differential operators ($\partial/\partial x_i, \nabla^2$, and $\overline{\mathbf{u}}\cdot\nabla$)) and combined genetic programming and linear regression in an iterative, residual-fitting procedure described in Algorithm 1. Crucially, in each genetic programming step, we define fitness in terms of correlation rather than absolute error, making fitting the outermost constants unnecessary. We run genetic programming with $\overline{q}$, $\overline{u}$, and $\overline{v}$ as our base features. Arbitrary powers or cross-multiples of these features can be discovered since the operator set includes multiplication. This approach allows us to discover all the same terms which appear in the feature library used for ZB2020, but is not limited to them.

Based on results obtained from the fully convolutional networks (described in Section 5.4), we chose to run our method on subgrid forcing computed with Operator 1 (Section 3.4.1), diagnosed using S_q to simplify learning. Running Algorithm 1 without any manual experimenter intervention leads to a formula of the forcing with an expression for each iteration given in Fig. 13. We saw offline performance increase significantly, with

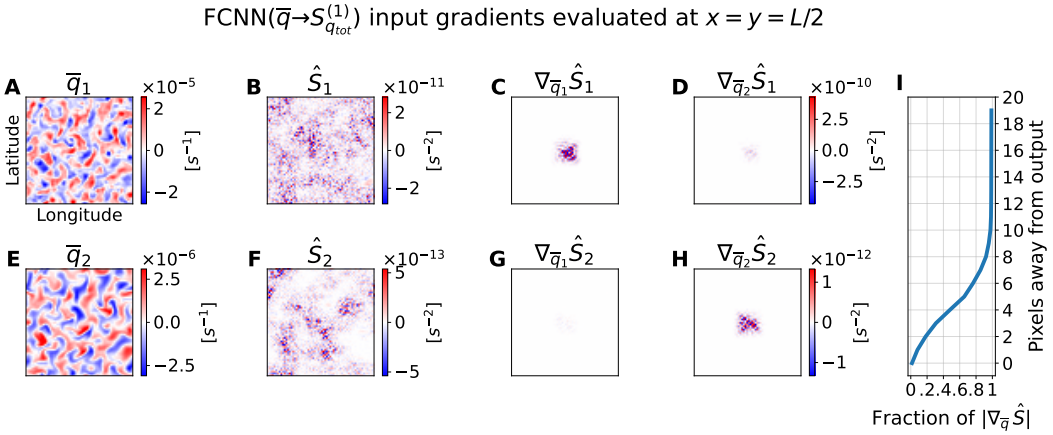

Figure 12. Input gradients of an FCNN mapping $\overline{q}$ (A,E) to $S^{(1)}_{q_{tot}}$ (B,F), evaluated at the center of the domain. Gradient magnitudes are largest around the x, y, z-position corresponding to the prediction (C,H). However, they still extend relatively far horizontally, needing 9 pixels to reach >95% of their full magnitude (I).

many of the discovered features seemingly physically-relevant, based on previous published parameterizations. In particular, we note that $\nabla^2(\overline{\mathbf{u}}\cdot\nabla)\overline{q}$ and $\nabla^2\overline{v}$ in the upper layer, together approximate a parameterization proposed by Porta Mana and Zanna (2014), though missing a Euleurian time derivative of PV which is not provided to the algorithm. However, terms discovered *after* the first two iterations tend to vary significantly on random restarts. Terms after the fourth iteration are also significantly harder to interpret (see right end of Figure 13). More importantly, we find that some combinations of terms include additions of terms with different units (e.g., q and u). In addition, parameterizations became unstable after the sixth iteration, and although online performance did improve over low-resolution models with the first 4-6 terms from the symbolic parameterizations, there were still significant differences with respect to many high-resolution diagnostics. To address these issues, we added a human-in-the-loop guidance step described below.

6.2 Human-in-the-loop guidance

Some manual intervention can be introduced during the learning procedure to improve interpretability and stability. We added a human-in-the-loop guidance step in each iteration (gray lines in Algorithm 1), where we edited or removed terms that seemed unphysical and sometimes added what seemed like natural extensions of existing terms. In our final OptionalUserEdits step, we attempted to prune the set of terms as much as possible by removing those whose removal did not worsen online performance or adding some that may improve it. We provide an account of our specific actions in Appendix C.

This procedure left us with a final parameterization of the form:

$$\begin{aligned} S_q^{\mathrm{GP}} &= (w_1\nabla^2 + w_2\nabla^4 + w_3\nabla^6)(\overline{\mathbf{u}}\cdot\nabla)\overline{q} \\ &+ (w_4\nabla^4 + w_5\nabla^6)\overline{q} \\ &+ (\overline{\mathbf{u}}\cdot\nabla)^2\nabla^2(w_6\overline{v}_x + w_7\overline{u}_y). \end{aligned} \tag{18}$$

Here w signify the linear weights. Evaluating this parameterization against FCNNs and traditional physics-based models, we find its performance competitive with neural networks in the eddy configuration (Figs. 16 and 17) and near-dominant in the jet config-

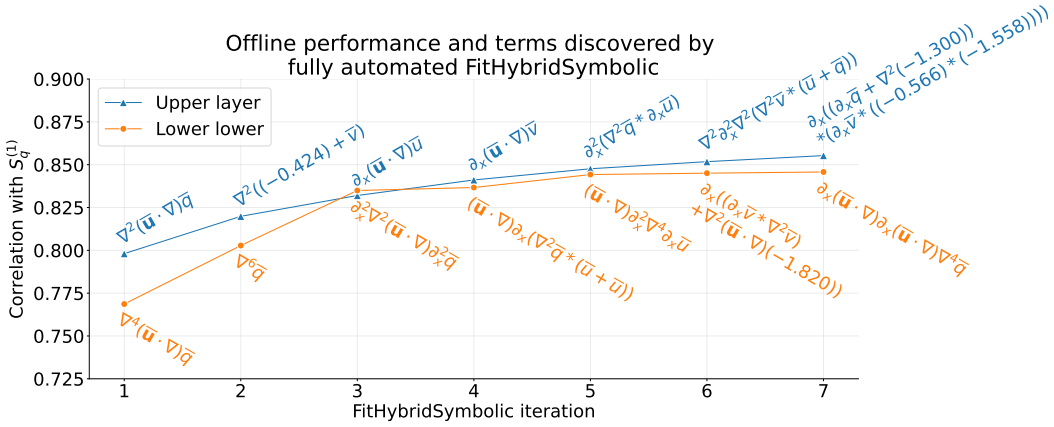

Figure 13. Offline correlation and sequence of terms discovered by Algorithm 1 without any human-in-the-loop intervention (terms learned for upper/lower layers in blue/orange respectively). Terms learned in initial iterations tended to be physically meaningful, relatively simple, and related to parameterizations in the literature, while terms learned in later iterations tended to be complex or unphysical (e.g. adding $\overline{u}$ and $\overline{q}$ despite incompatible units in iteration 6).

uration (18 and 19). We discuss its performance further in Section 7.1 where we compare and contrast different categories of parameterizations.

6.3 Symbolic regression feature importance

As in Section 5.6 for FCNNs, it is useful to quantify the relative importance of the different symbolic terms. One way to do this is by examining the weights w. These are visualized in Fig. 14 in two ways: (1) as raw values (on a log scale), and (2) normalized after dividing by the standard deviations of the corresponding features (on a linear scale), which makes them directly comparable despite each w_i having different units.

In normalized form, the largest coefficient in both layers is for $\nabla^4(\overline{\mathbf{u}}\cdot\nabla)\overline{q}$, though the second-largest upper-layer coefficient, $\nabla^6(\overline{\mathbf{u}}\cdot\nabla)\overline{q}$, is zero in the lower-layer, where Laplacians of PV are weighted much more significantly (though their weights are almost equal in absolute magnitude to the corresponding weights in the upper layer). The final two terms, $(\overline{\mathbf{u}}\cdot\nabla)^2\nabla^2\overline{v}_x$ and $(\overline{\mathbf{u}}\cdot\nabla)^2\nabla^2\overline{u}_y$, receive relatively little (normalized) weight in either layer. Although they can improve the performance of some online simulations (not shown), in our ablation study with more samples shown in Figure 15, the improvement is not significant.

Another way to estimate feature importance is by removing each term, re-fitting the linear regression coefficients, and re-evaluating online performance (Fig. 15). If we consider the performance decrease after removal of each feature as a measure of its importance, we reach similar conclusions: the ∇^4 and ∇^6 terms (for both $\overline{q}$ and $(\overline{\mathbf{u}}\cdot\nabla)\overline{q}$) are most important, the $\nabla^2(\overline{\mathbf{u}}\cdot\nabla)\overline{q}$ term is somewhat important, and the $(\overline{\mathbf{u}}\cdot\nabla)^2\nabla^2$ terms are relatively unimportant.

6.4 Interpretation of the learned expression

Note that the goal of the paper is not to focus on interpretability but to introduce methods for learning and evaluating parameterizations from data. Therefore, we are not claiming that this parameterization is more physical than anti-viscosity backscatter (Jansen

Algorithm 1 "Hybrid" linear and genetic programming-based symbolic regression (with optional human-in-the-loop interventions in light gray).

procedure FitGeneticProgram(x, y)
    Run `gplearn` (Stephens, 2019) with operators $\{\partial_x, \partial_y, \nabla^2, (\mathbf{u}\cdot\nabla), *, +\}$, and

$$\text{Fitness}(\texttt{term}) = |\text{Corr}(\texttt{term}(x), y)| - 0.001 * \text{Length}(\texttt{term})$$

end procedure

procedure FitLinearRegression(x, y)
    Find w to minimize $||w \cdot x - y||_2^2$
end procedure

procedure FitHybridSymbolic(x, y)
    `terms` $\leftarrow \emptyset$ ▷ set of symbolic expressions
    $w \leftarrow \emptyset$ ▷ weights of those expressions
    $\tilde{y} \leftarrow$ y ▷ residual forcing to predict

    repeat
        for all layers z **do**
            `terms` $\leftarrow$ `terms` $\cup$ FitGeneticProgram($x_z, \tilde{y}_z$) ▷ learn the next term
        end for
        `terms` $\leftarrow$ OptionalUserEdits(`terms`)
        for all layers z **do**
            $w_z \leftarrow$ FitLinearRegression(`terms`$(x_z), y_z$) ▷ reweight terms
            $\tilde{y}_z \leftarrow w_z \cdot$ `terms`$(x_z) - y_z$ ▷ update residuals
        end for
    until convergence or user decision

    return `terms`, w
end procedure

& Held, 2014) or deformation-based parameterizations (Anstey & Zanna, 2017). Yet, we will discuss briefly how the discovered terms compared to other subgrid parameterizations.

The components of the proposed model were discovered in the following order. In the first few iterations, quadratic expressions (proportional to $(\overline{\mathbf{u}}\cdot\nabla)\overline{q}$) were discovered. Quadratic models are often found to be highly-correlated with subgrid forcing (Porta Mana & Zanna, 2014; Anstey & Zanna, 2017; Meneveau & Katz, 2000; Layton & Rebholz, 2012), but often cannot be used as standalone parameterizations. The next few iterations led to eddy-viscosity models, $\nabla^4\overline{q}$ and $\nabla^6\overline{q}$. Particularly, both weights w_4 and w_5 being positive implies that there is dissipation of energy in small scales and redistribution to larger scales, i.e. backscattering (Jansen & Held, 2014). The final terms discovered are cubic in model variables and contains double-advection operator, $(\overline{\mathbf{u}}\cdot\nabla)^2$. The terms resemble the anticipated potential vorticity method from Vallis and Hua (1988). This method allows to preserve properties inherent to geostrophic turbulence such as conservation of energy and dissipation of enstrophy (Marshall & Adcroft, 2010), but it suffers from inaccurate representation of spectral fluxes (Thuburn et al., 2014). In summary, our discovered closure contains elements of existing subgrid parameterizations, which have pros and cons when used as standalone ones.

This symbolic parameterization includes up to the seventh spatial derivative of $\overline{q}$, which may be unrealistic to implement into a climate model. However, it might be more

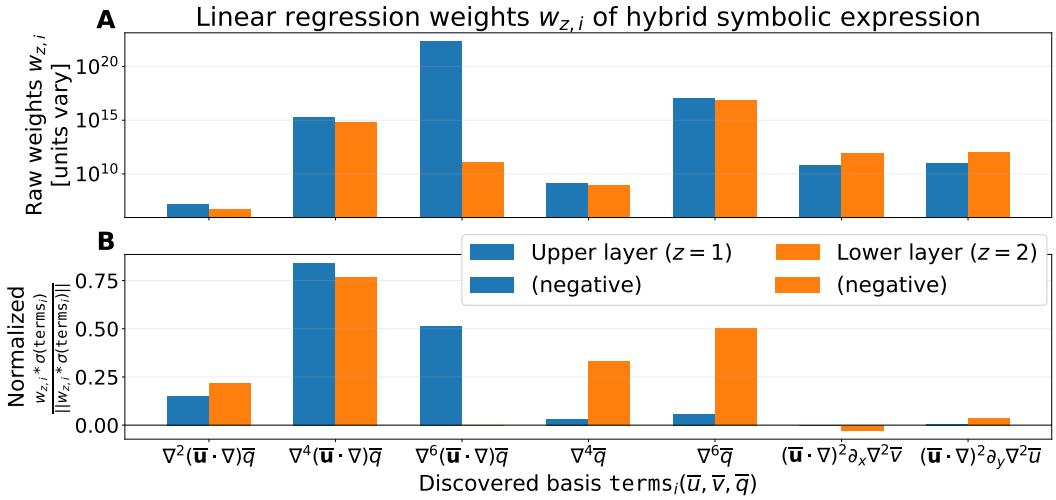

Figure 14. Linear regression-derived weights w for the human-in-the-loop genetic programming-derived basis terms of Equation 18, both as raw values (A, negative values shown with hatching) and normalized (B) after multiplying by the standard deviations of the terms over the training set (giving them consistent units). The absolute magnitudes of many terms are somewhat similar across layers, but their effective contributions to the output differ.

realistic than a fully non-local approach such as the convolutional neural network parameterizations considered in Section 5.

7 Discussion and Conclusion

We will finally compare our top parameterizations and then summarize our key findings in this section.

7.1 Comparing top parameterizations

To conclude our analysis, we focus on the top-performing models of different categories. The Pareto frontier of distributional and spectral similarity (Fig. 8) conveniently includes one FCNN, our symbolic parameterization, and two backscatter parameterizations (we select the one with higher spectral similarity). Note that the Smagorinsky parameterizations have very poor performance online (not surprisingly since they are dissipative) and we strongly encourage the community to choose better physics baselines when evaluating the performance of data-driven parameterization.

Offline on eddy configuration (Fig. 16), FCNN performance (A-E) is strongest overall, though power spectra diverge slightly at large scales (E). The symbolic regression model (F-J) performs slightly worse offline than the FCNN, but matches the power spectrum at all scales reasonably well. The backscatter model (K-O) performs much worse offline than the data-driven models, using R^2 as a metric. However, all three selected models perform well online (Fig. 17), with the FCNNs showing better distributional performance than the other models(Fig. 17C). However, the FCNN models seem to spin up the large scale faster than the other models (Fig. 17B).

On jet configuration, the offline performance remains similar for all models, except for the R^2 of the FCNN in the lower layer which is significantly lower than for the eddy configuration (Fig. 18B). However, online FCNN's performance degrades to significantly

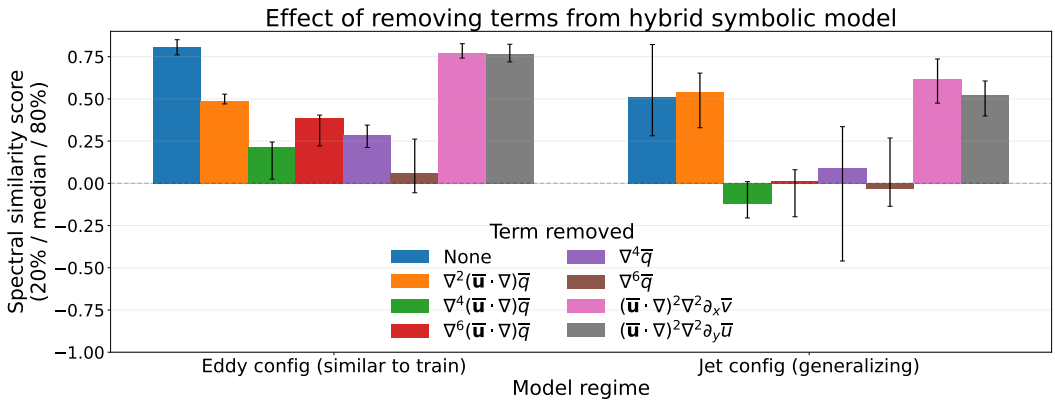

Figure 15. Effect of removing individual terms from the symbolic expression of Equation 18 on spectral similarity (median scores within groups, with errorbars showing the 20th and 80th percentiles). From left to right, removing $\nabla^2(\overline{\mathbf{u}}\cdot\nabla)\overline{q}$ reduced performance in eddy configuration, but not jet configuration. Removing ∇^4 and ∇^6 terms (for both $\overline{q}$ and $(\overline{\mathbf{u}}\cdot\nabla)\overline{q}$) drastically reduced performance in both configurations, which suggests these terms are crucial. Removing the $(\overline{\mathbf{u}}\cdot\nabla)^2\nabla^2$ terms had small effects, suggesting they could be dropped for future experiments.

worse than the low-resolution without parameterization (Figs. 19 and 20). In addition, the backscatter model does not have a significant impact on the low-resolution simulation, though this depends on which metric we consider (e.g., Fig. 19). On the other hand, the symbolic model remains fairly robust - without retraining or tuning in this new configuration.

FCNNs with different forcing formulations degraded slightly less when transferring to jet configuration (Fig E6). However, their average similarity scores were still low compared to the hybrid symbolic model (Fig. E9), and they disrupted the characteristic jet features, causing the flow to more closely resemble the eddy configuration on which they were trained (Fig. E3).

Even in the eddy configuration, `decorr_time`s for all models are only modestly closer to those from the high-resolution compared to those of the low-resolution simulation. In the case of the FCNN, the decorrelation times are actually worse (Fig. 21) than the low resolution. Using the decorrelation metric, Smagorinsky parameterizations actually performed best (slightly ahead of certain backscatter settings), even though they performed near the worst by all other metrics (see also Fig. E8). As expected, the data-driven parameterizations are doing well at representing the averaged statistics at coarse resolution (i.e., the climate) but do not improve the short term trajectories (i.e., the "weather").

7.2 Conclusion

We introduced a framework and a set of datasets for learning and evaluating ocean subgrid forcing parameterizations in a quasi-geostrophic setup, with a focus on a set of well-defined quantitative offline and online metrics. We used this framework to train and test physics-based and data-driven parameterizations under a variety of conditions, namely the different training datasets and definitions of subgrid forcing.

Several conclusions stand out as particularly relevant for developing subgrid parameterizations from high-resolution simulations for climate models (though note that the parameterizations developed here cannot easily be implemented in models due, for example, to the degree of non-locality needed). We summarize our key points as follows

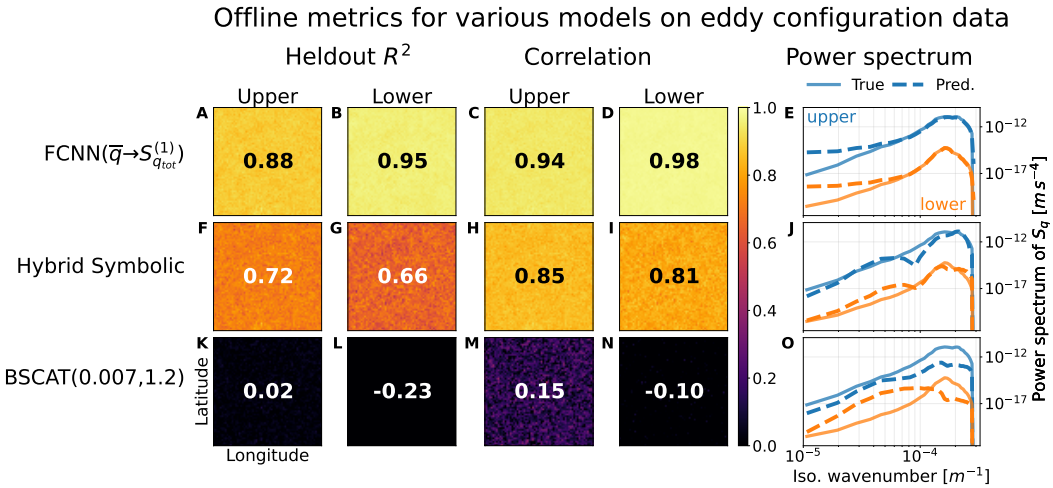

Figure 16. Offline performance for selected subgrid parameterizations on a heldout eddy configuration dataset computed with Operator 1, with means shown in spatial plots. FCNN performance (A-E) is strongest overall, though subgrid power spectra diverge slightly at large scales (E). The symbolic regression (F-J) model performs slightly worse, but matches the power spectrum at all scales reasonably well. The backscatter model (K-O) perform much worse offline (though all three perform well online, Fig. 17).

- **Metrics**: performance offline and online needs to be rigorously evaluated, rather than eyeballing improvement over a few selected diagnostics, to determine the accuracy and reliability of a given parameterization or simulation. Our open-source framework (Appendix D) will hopefully encourage the research community to find easy-to-use resources for such evaluation and facilitate the development of new parameterizations that more faithfully capture the effects of subgrid-scale processes.
- **Data design choice:** the filtering and coarse-graining operator is key, as hinted at in Zanna and Bolton (2021). The online results, for a given FCNN architecture, are highly sensitive to filtering choice. Therefore, we encourage testing multiple operators for data preparation guided by the target target application rather than varying hyperparameters or neural network architectures.
- **Stability**: Our architecturally-constrained FCNNs remained numerically stable in any configuration (as shown in Guillaumin and Zanna (2021) for different model configurations), which is also aided by the spectral truncation of high-frequency modes in `pyqg`. Moreover, in eddy configuration the FCNNs were robust enough to handle the distribution shift from coarsened high-resolution to low-resolution data. Therefore, online learning is not required for stable and robust online performance and might actually be a hindrance to generalization.
- **Generalization**: symbolic expressions, found using a new algorithm that we developed, generalized better than neural networks, which are infamously sensitive to even minor distributional shifts (Recht et al., 2018). With sufficiently diverse training data, FCNNs might perform much better in practice (Bolton & Zanna, 2019; O'Gorman & Dwyer, 2018).

There are many possible directions to improve the performance of the NN for generalization including training on multiple datasets as mentioned above, nondimensionalizing input variables (Beucler et al., 2021), finding a better latent space for our input (and eliminating spurious correlation with causal inference). In addition, finding better

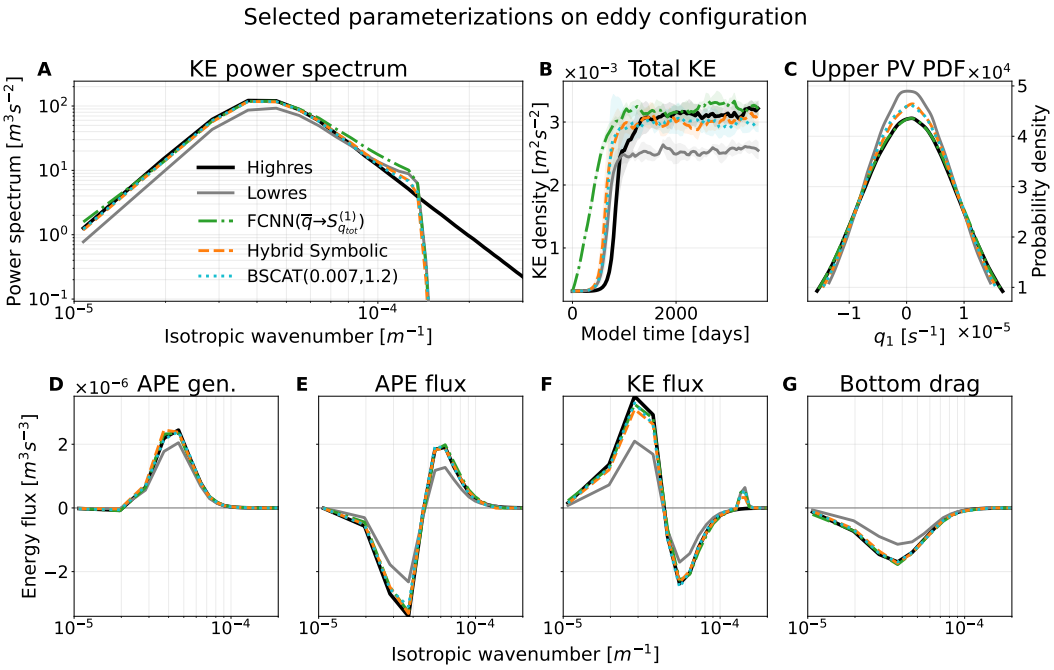

Figure 17. Sample of online performance diagnostics for symbolic regression and best FCNN/backscatter parameterizations by eddy-config `spectral_diff` (taken from the Pareto frontier of top models by spectral and distributional similarity, and averaged across five independent runs). Shading in KE time-series shows standard deviation over runs. All parameterizations improve significantly over the low-resolution model.

metrics for offline learning or testing might help ensure more robust results for online implementation, in particular in existing legacy climate models.

Appendix A Baseline Local Physical Parameterizations

A1 Smagorinsky

A common baseline for physical parameterizations was proposed by Smagorinsky (1963) as scale-selective dissipation. Given the strain-rate tensor, T,

$$T = \begin{pmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 2\overline{u}_x & \overline{u}_y + \overline{v}_x \\ \overline{u}_y + \overline{v}_x & 2\overline{v}_y \end{pmatrix}, \tag{A1}$$

the Smagorinsky parameterization predicts the subgrid forcing of u and v, denoted as S_{smag}, such that

$$S_{\text{smag}} = \begin{pmatrix} S_{u,\text{smag}} \\ S_{v,\text{smag}} \end{pmatrix} = 2\begin{pmatrix} (\nu_{\text{smag}} T_{11})_x + (\nu_{\text{smag}} T_{12})_y \\ (\nu_{\text{smag}} T_{21})_x + (\nu_{\text{smag}} T_{22})_y \end{pmatrix}, \tag{A2}$$

where the short-hands $()_{x,y} \equiv \frac{\partial}{\partial x,y}$ are used for low-resolution spatial derivatives,

$$\nu_{\text{smag}} = (C_S \Delta x)^2 \sqrt{T_{11}^2 + T_{12}^2 + T_{21}^2 + T_{22}^2}, \tag{A3}$$

and C_S is a tunable parameter. Here we will use $C_S \in \{0.075, 0.15, 0.3\}$.

Smagorinsky is a parameterization of small-scale dissipation, which can correct the tendency of low-resolution models to concentrate too much energy at small scales. However, the parameterization does not redistribute this energy back up to larger scales via

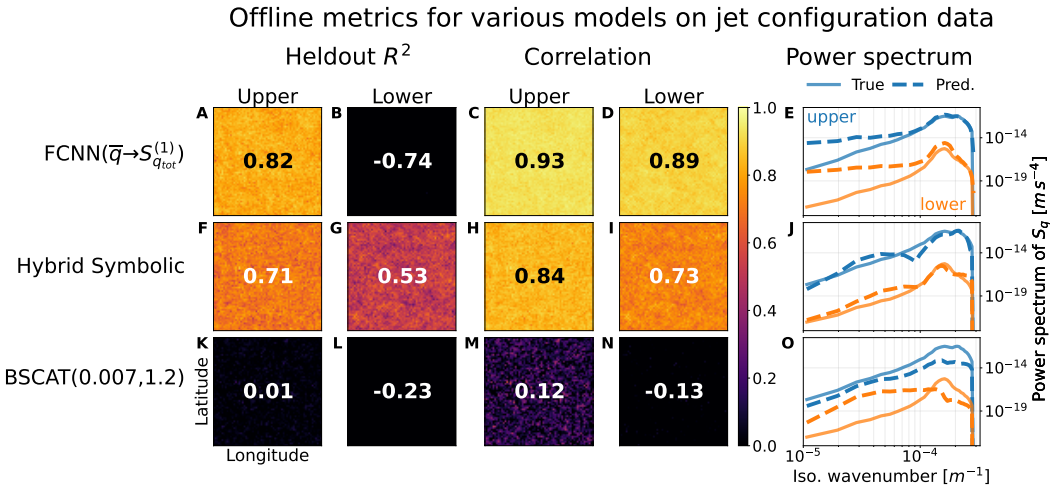

Figure 18. Offline performance as in Figure 18, but testing for generalization to jet configuration. For FCNNs (A-E), R^2 is lower in the upper layer and actually negative in the lower layer. However, correlation remains fairly high, suggesting that performance might improve with rescaling. For our symbolic regression model (F-J) and backscatter (K-O), offline performance remains similar to eddy configuration, though only the hybrid symbolic model generalizes online (Figure 19).

backscatter, as show in theoretical analysis and simulations of quasi-2D turbulence (Kraichnan, 1976; Thuburn et al., 2014; Natale & Cotter, 2017).

A2 Backscatter and Biharmonic Dissipation

Different parameterizations that can potentially address backscatter include the parameterization suggested by Jansen and Held (2014); Jansen et al. (2015), which consists of scale-selective dissipative operator and an additional negative viscosity part reinjecting energy at larger scales. The magnitude of the negative viscosity part is chosen such that resulting model approximately conserves energy.

We adapt this parameterizations for use in `pyqg`. The small-scale dissipation of enstrophy is parameterized with biharmonic Smagorinsky model (see Eq. A3)

$$F_{\text{smag}} = -\nabla^2 \left[\nu_{\text{smag}} \nabla^4 \overline{\psi}\right]. \tag{A4}$$

The negative viscosity backscatter is parameterized with less scale-selective Laplacian viscosity operator:

$$F_{\text{bscat}} = -\nu_{\text{bscat}} \nabla^4 \overline{\psi}, \tag{A5}$$

and total contribution to PV equation is given as $S_{\text{bscat}} = F_{\text{smag}} + F_{\text{bscat}}$. The negative viscosity backscatter re-injects the C_B fraction of the total energy dissipated by the biharmonic model. As such, the negative viscosity coefficient is given by:

$$\nu_{\text{bscat}} = C_B \frac{\sum_{i=1}^{2} H_i \iint \overline{\psi}_i F_{\text{smag},i} d\overline{x}\, d\overline{y}}{\sum_{i=1}^{2} H_i \iint \overline{\psi}_i \nabla^4 \overline{\psi}_i d\overline{x}\, d\overline{y}} \tag{A6}$$

where $F_{\text{smag},i}$ is the value of Eq. A4 at a particular layer. We run this parameterization at 36 parameter settings corresponding to every combination of $C_B \in \{.7, .8, .9, 1.0, 1.1, 1.2\}$ and $C_S^2 \in \{.003, .005, .007, .01, .02, .04\}$ (the use of C_S^2 is for convenience).

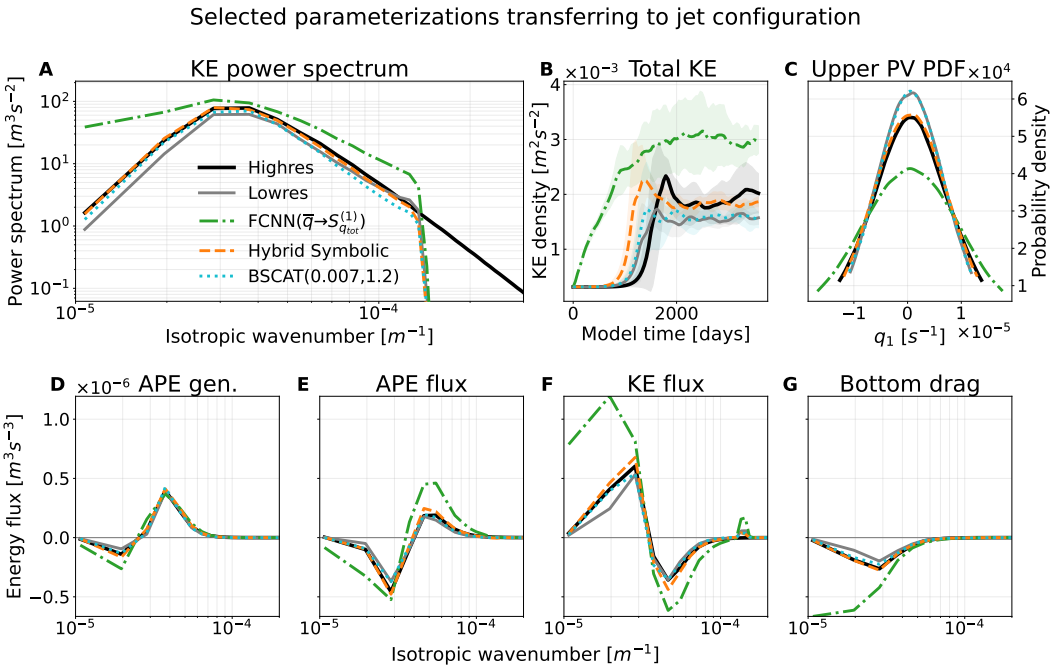

Figure 19. Similar to Figure 17, but evaluated on jet rather than eddy configuration (without retuning). The hybrid symbolic parameterization still improves significantly over low-resolution model, while backscatter has no discernible effect and FCNNs degrade significantly.

A3 Zanna Bolton Data-Driven Equation-Discovery parameterization

Using data from an idealized primitive equation model and relevance vector machine, Zanna and Bolton (2020) learned an expression for the subgrid momentum forcing. They use both barotropic and baroclinic simulated data, and apply Gaussian filtering with coarse-graining to diagnose the subgrid forcing. The form of the parameterization is given by

$$\hat{\mathbf{S}}_{\mathbf{u}}^{\mathrm{ZB2020}} \approx \kappa^{\mathrm{ZB2020}} \nabla \cdot \begin{pmatrix} -\zeta D & \zeta \tilde{D} \\ \zeta \tilde{D} & \zeta D \end{pmatrix} + \mathbf{I} \frac{1}{2} \kappa^{\mathrm{ZB2020}} \nabla (\zeta^2 + D^2 + \tilde{D}^2), \tag{A7}$$

for each vertical layer, with

$$\zeta = \overline{v}_x - \overline{u}_y, \qquad \sigma = \overline{u}_x + \overline{v}_y, \tag{A8a}$$

$$D = \overline{u}_y + \overline{v}_x, \qquad \tilde{D} = \overline{u}_x - \overline{v}_y, \tag{A8b}$$

where ζ is the relative vorticity, σ is the divergence, and D and $\tilde{D}$ are the shearing and stretching deformation of the low-resolution flow field, respectively.

For online tests, rather than using the value of κ^{ZB2020} diagnosed in Zanna and Bolton (2020), we fit the parameter empirically to achieve maximal offline R^2 on the training set (equivalent to that generated using Operator 2). For online simulations, we also test at $\kappa^{\mathrm{ZB2020}} = 2$ and $1/2$ times the empirically fit value.

Appendix B Decomposition of subgrid forcing

We can further decompose subgrid contribution into the contribution towards kinetic energy and the contribution towards potential energy. Let S_ψ be the tendency in

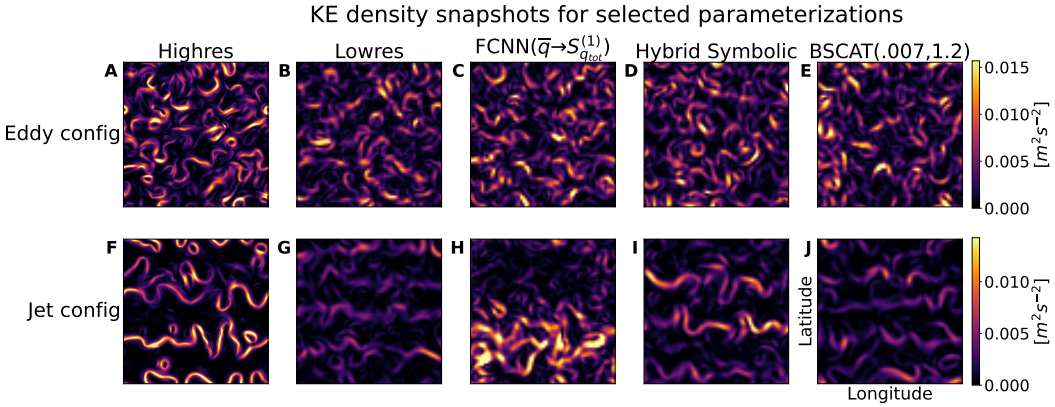

Figure 20. Randomly chosen snapshots of kinetic energy density for selected parameterizations on eddy (A-E) and jet configuration (F-J). On jet configuration, the symbolic parameterization (J) matches high-resolution (F) reasonably well, while the FCNN (I) deviates significantly and backscatter (H) does not appear to have any effect or modify the low-resolution (G). See Figures E1 and E2 for more.

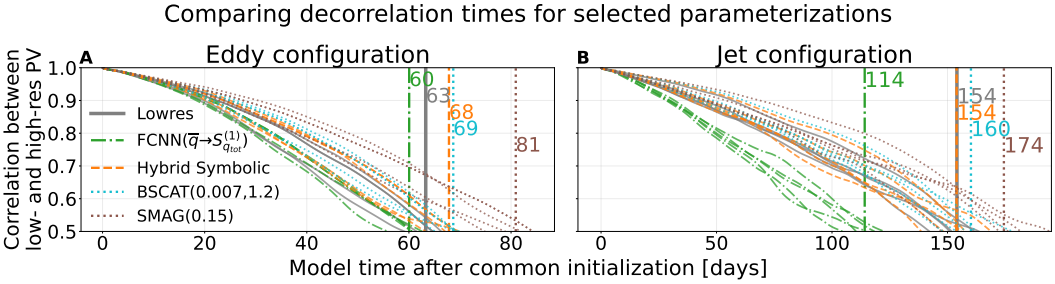

Figure 21. `decorr_time` results for selected parameterizations on eddy (A) and jet (B) configuration. FCNNs diverged from high resolution models faster than unparameterized models, while backscatter and hybrid symbolic parameterizations stayed correlated for similar durations. Smagorinsky parameterizations stayed correlated significantly longer.

the streamfunction induced by subgrid forcing, we use Eq. 3 to rewrite Eq. 10 as

$$\left(\frac{\partial E(k,l)}{\partial t}\right)^{\text{sub}} = -\frac{1}{H}\sum_{m=1}^{2} H_m \mathbb{R}\left[\hat{\psi}_m^*[(-\kappa^2\mathbf{I}+\mathbf{M})\hat{\mathbf{S}}_\psi]\right]$$
$$= \frac{1}{H}\kappa^2\sum_{m=1}^{2} H_m \mathbb{R}\left[\hat{\psi}_m^*\left(\mathbf{A}_\kappa\hat{\mathbf{S}}_q\right)_m\right] - \frac{1}{H}\sum_{m=1}^{2} H_m \mathbb{R}\left[\hat{\psi}_m^*\left(\mathbf{M}\mathbf{A}_\kappa\hat{\mathbf{S}}_q\right)_m\right], \tag{B1}$$

where $\mathbf{A}_\kappa = (-\kappa^2\mathbf{I}+\mathbf{M})^{-1}$. On the right-hand side of Eq. B1, the first term matches the definition of the contribution towards kinetic energy, and we regard the second term as the contribution towards potential energy. This decomposition is used in calculating the spectral similarity scores.

Appendix C Human-in-the-Loop Symbolic Regression Steps

In this section, we describe the specific "OptionalUserEdits" steps we took in applying Algorithm 1 to obtain Equation 18.

In the first `gplearn` step, we discovered $\nabla^2(\overline{\mathbf{u}}\cdot\nabla)\overline{q}$ (in the upper layer) and $\nabla^4(\overline{\mathbf{u}}\cdot\nabla)\overline{q}$ (in the lower layer), which gave us training set correlations of 0.80 (upper) and 0.77 (lower) after fitting models with both terms to each layer. To this, we added $\nabla^6(\overline{\mathbf{u}}\cdot\nabla)\overline{q}$ to extend the pattern, which brought the same correlations to 0.84 and 0.82. We then ran the next `gplearn` step, which outputted $\nabla^4\overline{q}$ (upper) and $\nabla^6\overline{q}$ (lower). This brought correlations up to 0.845 (upper) and 0.836 (lower). We kept both these terms, and experimented with adding $\nabla^8\overline{q}$, but correlations actually decreased in the lower layer. We then ran the next `gplearn` step, which returned $(\overline{\mathbf{u}}\cdot\nabla)^2\nabla^2\partial_x\overline{v}$ and $\partial_x\nabla^8\overline{q}$. This nudged correlations to 0.846 (upper) and 0.838 (lower), which nudged very slightly higher to 0.846 and 0.840 when further adding the counterparts of these terms obtained by switching x and y, $(\overline{\mathbf{u}}\cdot\nabla)^2\nabla^2\partial_y\overline{u}$ and $\partial_y\nabla^8\overline{q}$.

From this set of terms (which includes all terms in Equation 18 with the addition of two ninth-order $\partial_i\nabla^8\overline{q}$ terms), we began a final OptionalUserEdits step using online performance as a guide (removing each term individually, but pairing up the removals of the terms with natural x and y counterparts). In this step, we found that the $\partial_i\nabla^8\overline{q}$ terms were actually hampering online performance (i.e. performance rose without them), while the others all appeared to help (i.e. performance fell without them)—though our results in Figure 15 later showed that the slight improvement we saw from the $(\overline{\mathbf{u}}\cdot\nabla)^2\nabla^2\partial_i\overline{\mathbf{u}}_i$ terms was not significant. We then accepted the expression of Equation 18 as our final output, saving its weights (learned with respect to eddy-config $S_q^{(1)}$).

Note that because the genetic programming steps are stochastic, re-running this procedure with a different random seed might produce different results. For example, in Figure 13, we discovered a $\nabla^2\overline{v}$ term in the second step, but in this case such a term was never learned (though this could be alternately explained by the manual addition of $\nabla^6(\overline{\mathbf{u}}\cdot\nabla)\overline{q}$, which may have accounted for its contribution).

Appendix D Open Research

Version 1.0.1 of the Python repository used for training and evaluating parameterizations is preserved at `https://doi.org/10.5281/zenodo.6656313`, available via the MIT license and developed openly at `https://github.com/m2lines/pyqg_parameterization_benchmarks` (Ross et al., 2022).

The baseline high- and low-resolution datasets used for evaluating parameterizations, as well as the subgrid forcing datasets used for training them, are available at Zenodo via `https://doi.org/10.5281/zenodo.6609034` under a Creative Commons Attribution 4.0 International license (Ross, 2022).

Appendix E Supplementary Figures

This section includes additional result figures.

Acknowledgments

This research is supported by the generosity of Eric and Wendy Schmidt by recommendation of Schmidt Futures, as part of its Virtual Earth System Research Institute (VESRI). C.F.G. was partially supported by the NSF DMS grant 2009752. This research was also supported in part through the NYU IT High Performance Computing resources, services, and staff expertise. Finally, the authors would like to thank Elizabeth Yankovsky, Ryan Abernathey, Adam Subel, and Tom Beucler for helpful conversations and suggestions.

References

Abernathey, R., Rocha, C. B., Ross, A., Jansen, M., Li, Z., Poulin, F. J., ...

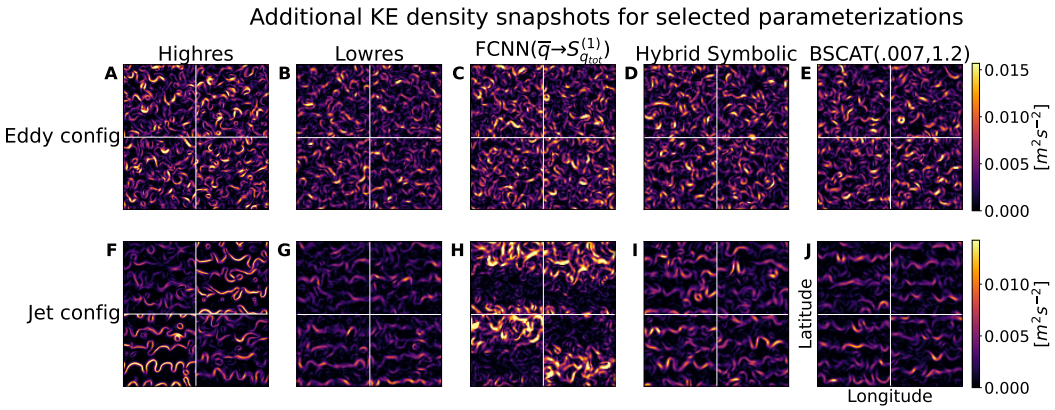

Figure E1. Like Figure 20, but showing additional randomly chosen snapshots.

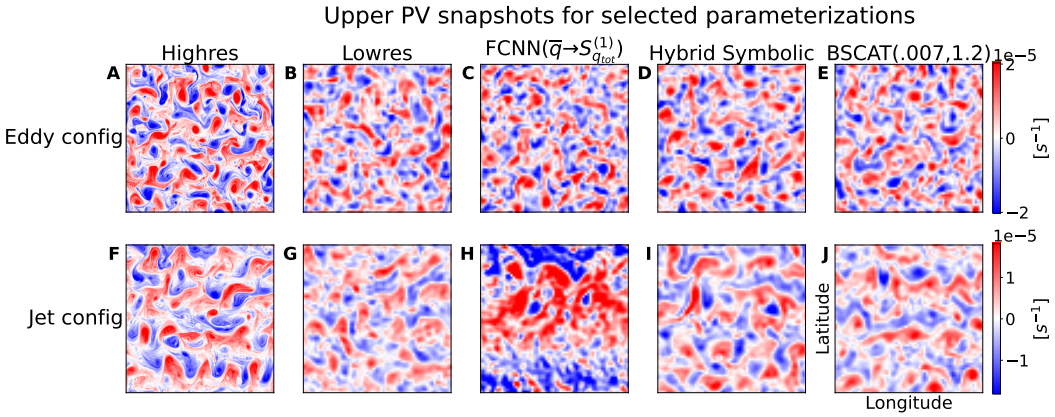

Figure E2. Like Figure 20, but showing upper potential vorticity q_1 rather than KE density.

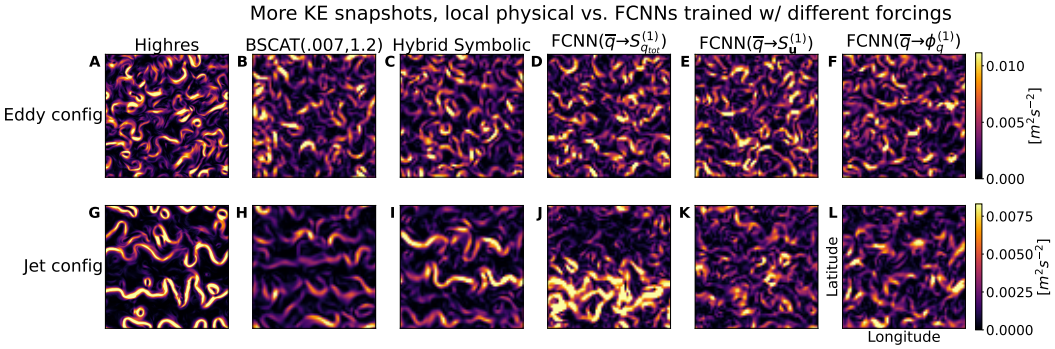

Figure E3. Like Figure 20, but additionally showing KE snapshots for FCNNs trained on eddy configuration data with different forcing formulations (see Figures E5 and E6). All FCNNs produce reasonable results on eddy configuration (D-F), but on jet configuration (J-L), the snapshots do not resemble high-resolution (G), with either latitude-specific increases in energy (J) or disruption of jets in favor of isotropic eddies (K-L), resembling FCNN training conditions.

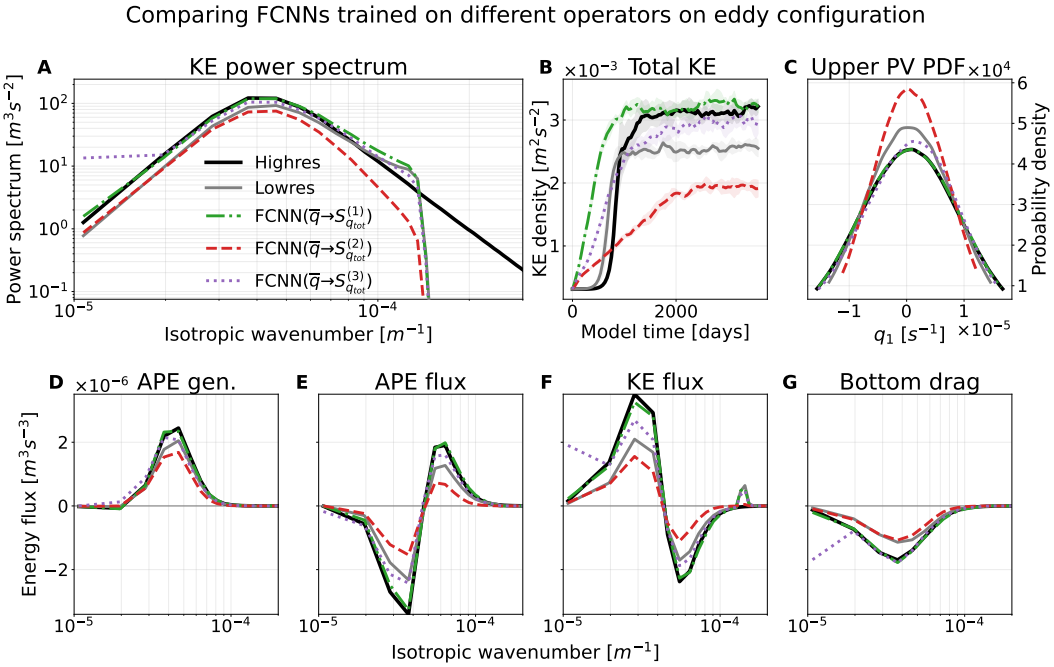

Figure E4. Like Figure 17, but comparing FCNNs trained to predict $S_{q_{tot}}$ computed with each filtering and coarse-graining operator. Only the model trained with Operator 1 (Section 3.4.1) performs near-optimally, though the model trained with Operator 3 (Section 3.4.3) does well except for deviations in spectral metrics at large scales (A,F,G). These results suggest the filtering and coarse-graining operator is important for parameterization performance.

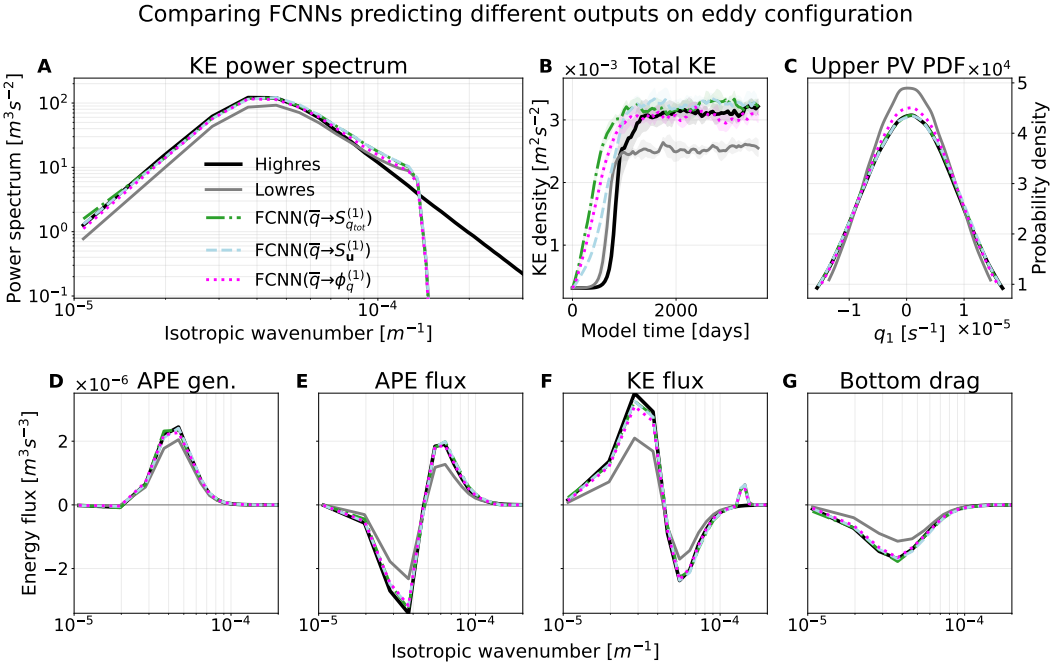

Figure E5. Like Figure 17, but comparing the online eddy configuration performance of FCNNs trained to predict different subgrid forcing formulations ($S_{q_{tot}}$, $S_{\mathbf{u}}$, and ϕ_q) computed with Operator 1 (Equation 11). All perform almost equally well, suggesting that the forcing formulation may matter much less than the filtering and coarse-graining operator (Figure E4).

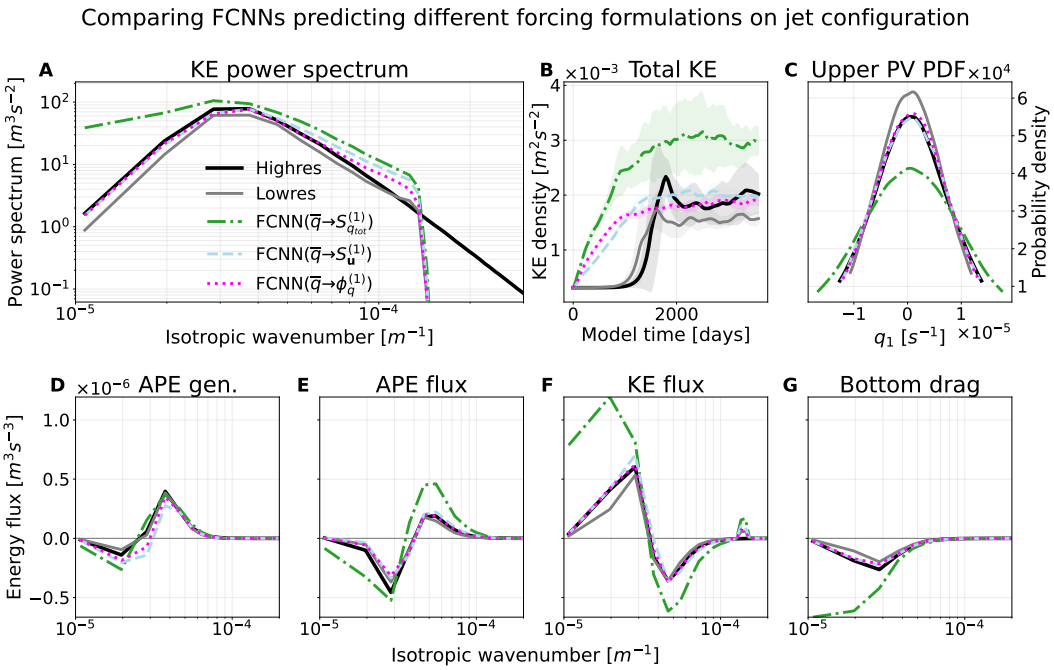

Figure E6. Like Figure 19, but comparing the online jet configuration performance of FCNNs trained to predict different subgrid forcing formulations ($S_{q_{tot}}$, $S_{\mathbf{u}}$, and ϕ_q) computed with Operator 1 (Equation 11). In this case, the models trained to predict $S^{(1)}_{\mathbf{u}}$ and $\phi^{(1)}_q$ appear to generalize better. However, their average scores across the full set of metrics (e.g. Figure E9) remain low, and in KE snapshots from these FCNNs (Figure E3), the characteristic jet behavior we see in high-resolution is absent.

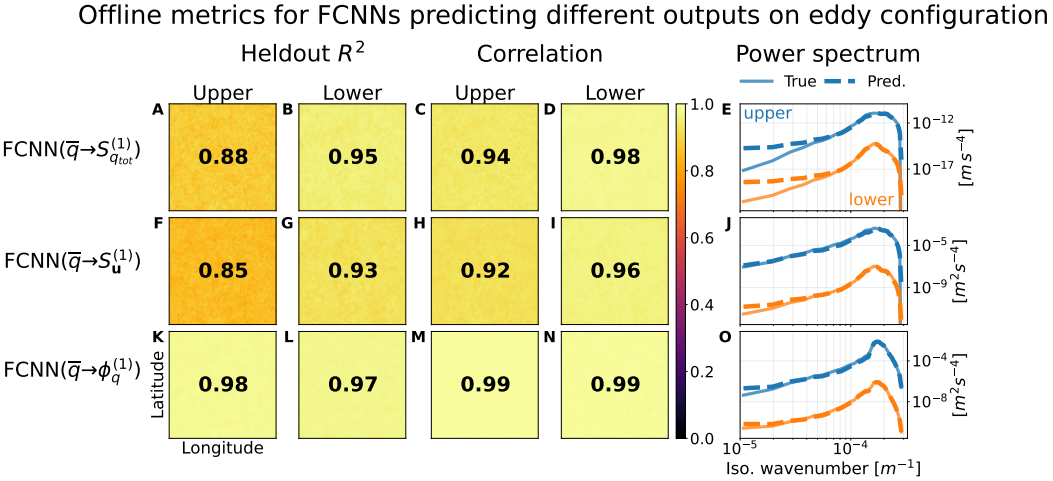

Figure E7. Offline results for more forcing formulations ($S^{(1)}_{\mathbf{u}}$ and $\phi^{(1)}_q$ results show averages over u and v terms). Many performance metrics are generally higher for models trained to predict subgrid fluxes (K-N), but this difference disappears if we compute them with respect to the implied subgrid forcing (i.e. by taking the divergence of the predicted quantities and comparing that to the true subgrid forcing, rather than comparing predicted to true subgrid fluxes).

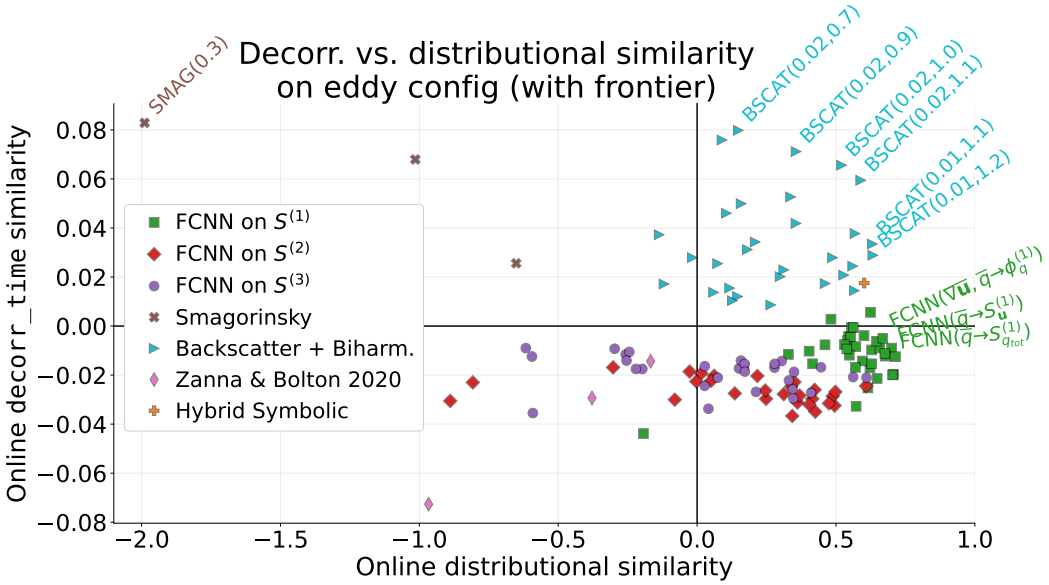

Figure E8. Like Figure 8, but comparing distributional similarity (Section 4.2.2) with decorrelation time similarity (Section 4.2.3). Smagorinsky and backscatter parameterizations (which form most of the Pareto frontier) increase decorrelation time, though only by about 8% of the gap between low- and high-resolution decorrelation times (which is what the y-axis signifies). Neural networks almost universally reduce it, while the hybrid symbolic parameterization modestly increases it.

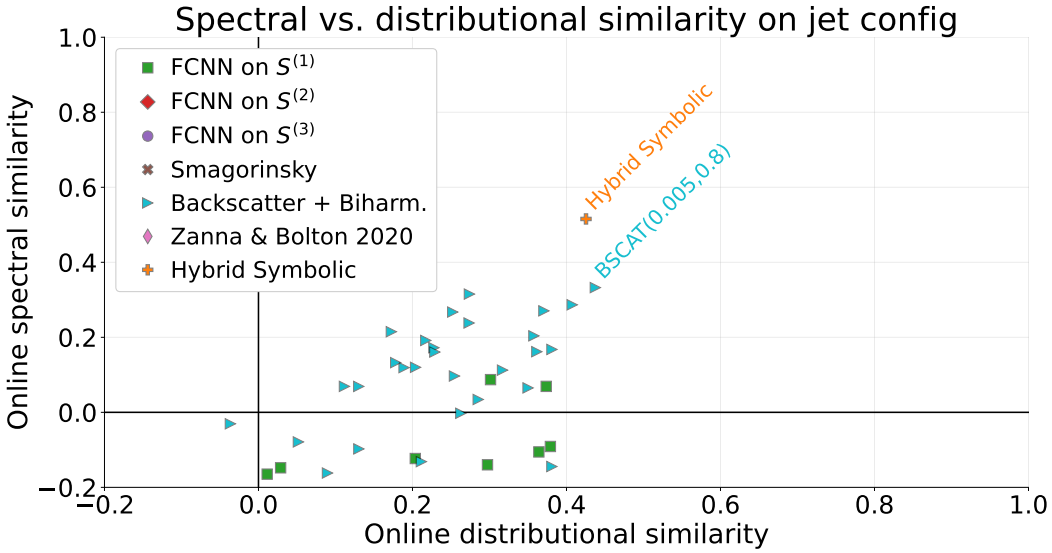

Figure E9. Like Figure 8, but evaluated on jet configuration. In this regime, the only models which appear on the Pareto frontier (highlighted in text) are the hybrid symbolic model and one parameter setting of the backscatter parameterization, which differs significantly from the eddy-configuration Pareto-optimal settings shown in Figures 8 and E8.

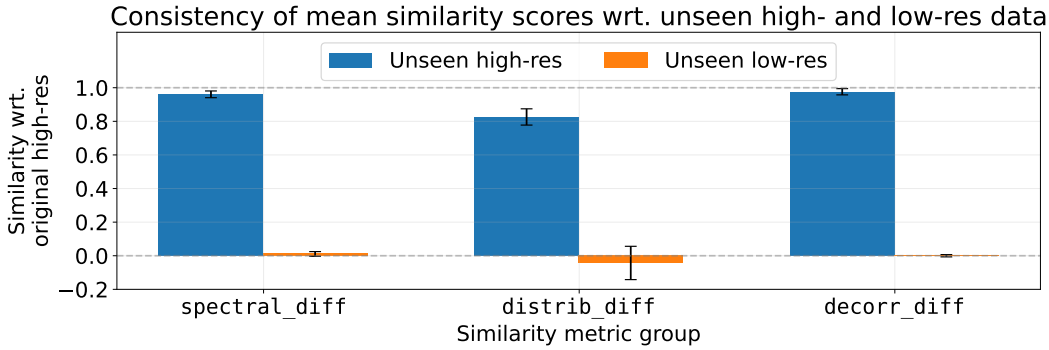

Figure E10. Mean similarity scores for *unseen* high- and low-resolution simulations with respect to the actual high- and low-resolution datasets used to evaluate parameterizations. Error-bars show means and standard deviations over 10 random samples of 5 simulations from a set of 25 unseen simulations. Spectral and decorrelation time similarity scores between different randomly re-run high-res simulations are >0.95 on average (and ≤0.01 on average for unseen low-resolution simulations), indicating they are fairly reliable (they should be near 1 for high-res and near 0 for low-res). End-of-simulation distributional similarity scores are a bit noisier, averaging 0.83 (so such scores in our results of above ≈0.8 are potentially near-optimal). Although distributional similarity scores are still precise enough to provide meaningful insight into parameterization performance, future experiments could improve their precision by increasing the size of ensembles, or by comparing distributions marginalized over more than just the final timestep.

Tobias (2022, May). *pyqg/pyqg: v0.7.2.* Zenodo. Retrieved from `https://doi.org/10.5281/zenodo.6563667` doi: 10.5281/zenodo.6563667

Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., & Kim, B. (2018). Sanity checks for saliency maps. *Advances in neural information processing systems*, *31*.

Anstey, J. A., & Zanna, L. (2017). A deformation-based parametrization of ocean mesoscale eddy reynolds stresses. *Ocean Modelling*, *112*, 99–111.

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., & Samek, W. (2015). On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, *10*(7), e0130140.

Baehrens, D., Schroeter, T., Harmeling, S., Kawanabe, M., Hansen, K., & Müller, K.-R. (2010). How to explain individual classification decisions. *The Journal of Machine Learning Research*, *11*, 1803–1831.

Beck, A., Flad, D., & Munz, C.-D. (2019). Deep neural networks for data-driven les closure models. *Journal of Computational Physics*, *398*, 108910.

Beck, A., & Kurz, M. (2021). A perspective on machine learning methods in turbulence modeling. *GAMM-Mitteilungen*, *44*(1), e202100002.

Berloff, P., Ryzhov, E., & Shevchenko, I. (2021). On dynamically unresolved oceanic mesoscale motions. *Journal of Fluid Mechanics*, *920*.

Berloff, P. S. (2005). Random-forcing model of the mesoscale oceanic eddies. *Journal of Fluid Mechanics*, *529*, 71–95. doi: 10.1017/S0022112005003393

Beucler, T., Pritchard, M. S., Yuval, J., Gupta, A., Peng, L., Rasp, S., ... Gentine, P. (2021). Climate-invariant machine learning. *CoRR*, *abs/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. doi: 10.1029/2018MS001472

Brenowitz, N. D., & Bretherton, C. S. (2018). Prognostic validation of a neural net-

977 work unified physics parameterization. *Geophysical Research Letters*, *45*(12),
978 6289-6298. doi: https://doi.org/10.1029/2018GL078510
979 Brunton, S. L., Proctor, J. L., & Kutz, J. N. (2016). Sparse identification of nonlin-
980 ear dynamics with control (sindyc). *IFAC-PapersOnLine*, *49*(18), 710–715.
981 Champion, K., Lusch, B., Kutz, J. N., & Brunton, S. L. (2019). Data-driven
982 discovery of coordinates and governing equations. *Proceedings of the*
983 *National Academy of Sciences*, *116*(45), 22445-22451. Retrieved from
984 `https://www.pnas.org/doi/abs/10.1073/pnas.1906995116` doi:
985 10.1073/pnas.1906995116
986 Fox, D. G., & Orszag, S. A. (1973). Pseudospectral approximation to two-
987 dimensional turbulence. *Journal of Computational Physics*, *11*(4), 612–619.
988 Fox-Kemper, B., Adcroft, A., Böning, C. W., Chassignet, E. P., Curchitser, E.,
989 Danabasoglu, G., ... Yeager, S. G. (2019). Challenges and prospects in
990 ocean circulation models. *Frontiers in Marine Science*, *6*. doi: 10.3389/
991 fmars.2019.00065
992 Fox-Kemper, B., Bachman, S. D., Pearson, B. C., & Reckinger, S. J. (2014). Princi-
993 ples and advances in subgrid modelling for eddy- rich simulations..
994 Frezat, H., Sommer, J. L., Fablet, R., Balarac, G., & Lguensat, R. (2022). A poste-
995 riori learning for quasi-geostrophic turbulence parametrization. *arXiv preprint*
996 *arXiv:2204.03911*.
997 Gent, P. R., & Mcwilliams, J. C. (1990). Isopycnal mixing in ocean circulation mod-
998 els. *Journal of Physical Oceanography*, *20*(1), 150 - 155. doi: 10.1175/1520
999 -0485(1990)020⟨0150:IMIOCM⟩2.0.CO;2
1000 Gent, P. R., Willebrand, J., McDougall, T. J., & McWilliams, J. C. (1995). Pa-
1001 rameterizing eddy-induced tracer transports in ocean circulation models. *Jour-*
1002 *nal of Physical Oceanography*, *25*(4), 463 - 474. doi: 10.1175/1520-0485(1995)
1003 025⟨0463:PEITTI⟩2.0.CO;2
1004 Griffies, S., Adcroft, A., Hewitt, H., Oning, C., Chassignet, E., Danabasoglu, G., ...
1005 Lab, D. (2009, 01). Problems and prospects in large-scale ocean circulation
1006 models. *OceanObs09 Proceedings*.
1007 Griffies, S., Winton, M., Anderson, W. G., Benson, R., Delworth, T. L., Dufour,
1008 C. O., ... Zhang, R. (2015). Impacts on ocean heat from transient mesoscale
1009 eddies in a hierarchy of climate models. *Journal of Climate*, *28*(3), 952 - 977.
1010 Retrieved from `https://journals.ametsoc.org/view/journals/clim/28/3/`
1011 `jcli-d-14-00353.1.xml` doi: 10.1175/JCLI-D-14-00353.1
1012 Grooms, I., Loose, N., Abernathey, R., Steinberg, J., Bachman, S. D., Marques, G.,
1013 ... Yankovsky, E. (2021). Diffusion-based smoothers for spatial filtering of
1014 gridded geophysical data. *Journal of Advances in Modeling Earth Systems*,
1015 *13*(9), e2021MS002552.
1016 Grooms, I., Nadeau, L.-P., & Smith, K. S. (2013). Mesoscale eddy energy locality in
1017 an idealized ocean model. *Journal of physical oceanography*, *43*(9), 1911–1923.
1018 Guan, Y., Chattopadhyay, A., Subel, A., & Hassanzadeh, P. (2022). Stable a poste-
1019 riori les of 2d turbulence using convolutional neural networks: Backscattering
1020 analysis and generalization to higher re via transfer learning. *Journal of Com-*
1021 *putational Physics*, *458*, 111090.
1022 Guillaumin, A. P., & Zanna, L. (2021). Stochastic-deep learning parameterization
1023 of ocean momentum forcing. *Journal of Advances in Modeling Earth Systems*,
1024 *n/a*(n/a), e2021MS002534. Retrieved from `https://agupubs.onlinelibrary`
1025 `.wiley.com/doi/abs/10.1029/2021MS002534` (e2021MS002534
1026 2021MS002534) doi: https://doi.org/10.1029/2021MS002534
1027 Hallberg, R. (2013). Using a resolution function to regulate parameterizations of
1028 oceanic mesoscale eddy effects. *Ocean Modelling*, *72*, 92–103.
1029 Hastie, T., Tibshirani, R., & Wainwright, M. (2015). Statistical learning with spar-
1030 sity. *Monographs on statistics and applied probability*, *143*, 143.
1031 Hewitt, H. T., Roberts, M., Mathiot, P., Biastoch, A., Blockley, E., Chassignet,

E. P., ... others (2020). Resolving and parameterising the ocean mesoscale in earth system models. *Current Climate Change Reports*, *6*(4), 137–152.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*, 359–366. (arXiv: 1011.1669v3 ISBN: 08936080 (ISSN)) doi: 10.1016/0893-6080(89)90020-8

Jansen, M. F., & Held, I. M. (2014). Parameterizing subgrid-scale eddy effects using energetically consistent backscatter. *Ocean Modelling*, *80*, 36–48.

Jansen, M. F., Held, I. M., Adcroft, A., & Hallberg, R. (2015). Energy budget-based backscatter in an eddy permitting primitive equation model. *Ocean Modelling*, *94*, 15–26.

Kent, J., Jablonowski, C., Thuburn, J., & Wood, N. (2016). An energy-conserving restoration scheme for the shallow-water equations. *Quarterly Journal of the Royal Meteorological Society*, *142*(695), 1100–1110.

Khani, S., & Porté-Agel, F. (2017). Evaluation of non-eddy viscosity subgrid-scale models in stratified turbulence using direct numerical simulations. *European Journal of Mechanics - B/Fluids*, *65*, 168-178. doi: https://doi.org/10.1016/j.euromechflu.2017.03.009

Killworth, P. D. (1997). On the parameterization of eddy transfer part i. theory. *Journal of Marine Research*, *55*(6), 1171–1197.

Kindermans, P.-J., Hooker, S., Adebayo, J., Alber, M., Schütt, K. T., Dähne, S., ... Kim, B. (2019). The (un) reliability of saliency methods. In *Explainable ai: Interpreting, explaining and visualizing deep learning* (pp. 267–280). Springer.

Koza, J. R. (1994). Genetic programming as a means for programming computers by natural selection. *Statistics and computing*, *4*(2), 87–112.

Kraichnan, R. H. (1976). Eddy viscosity in two and three dimensions. *Journal of Atmospheric Sciences*, *33*(8), 1521–1536.

Krasnopolsky, V. M., Fox-Rabinovitz, M. S., Hou, Y. T., Lord, S. J., & Belochitski, A. 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. Retrieved from `https://journals.ametsoc.org/view/journals/mwre/138/5/2009mwr3149.1.xml` doi: 10.1175/2009MWR3149.1

Layton, W. J., & Rebholz, L. G. (2012). *Approximate deconvolution models of turbulence: analysis, phenomenology and numerical analysis* (Vol. 2042). Springer Science & Business Media.

Li, H., Zhao, Y., Wang, J., & Sandberg, R. D. (2021). Data-driven model development for large-eddy simulation of turbulence using gene-expression programming. *Physics of Fluids*, *33*(12), 125127.

Loose, N., Abernathey, R., Grooms, I., Busecke, J., Guillaumin, A., Yankovsky, E., ... Martin, P. (2022). Gcm-filters: A python package for diffusion-based spatial filtering of gridded data. *Journal of Open Source Software*, *7*(70), 3947. Retrieved from `https://doi.org/10.21105/joss.03947` doi: 10.21105/joss.03947

Lund, T. (1997). On the use of discrete filters for large eddy simulation. *Annual Research Briefs*, 83–95.

Marshall, D. P., & Adcroft, A. J. (2010). Parameterization of ocean eddies: Potential vorticity mixing, energetics and arnold's first stability theorem. *Ocean Modelling*, *32*(3-4), 188–204.

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.

Meneveau, C., & Katz, J. (2000). Scale-invariance and turbulence models for large-eddy simulation. *Annual Review of Fluid Mechanics*, *32*(1), 1–32.

Mojgani, R., Chattopadhyay, A., & Hassanzadeh, P. (2021). Closed-form discovery of structural errors in models of chaotic systems by integrating bayesian sparse

regression and data assimilation. *arXiv preprint arXiv:2110.00546*.

Natale, A., & Cotter, C. J. (2017). Scale-selective dissipation in energy-conserving finite-element schemes for two-dimensional turbulence. *Quarterly Journal of the Royal Meteorological Society*, *143*(705), 1734–1745.

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

Orszag, S. A. (1971). On the elimination of aliasing in finite-difference schemes by filtering high-wavenumber components. *Journal of Atmospheric Sciences*, *28*(6), 1074–1074.

Pawar, S., San, O., Rasheed, A., & Vedula, P. (2020). A priori analysis on deep learning of subgrid-scale parameterizations for kraichnan turbulence. *Theoretical and Computational Fluid Dynamics*, *34*(4), 429–455.

Piomelli, U., Moin, P., & Ferziger, J. H. (1988). Model consistency in large eddy simulation of turbulent channel flows. *The Physics of Fluids*, *31*(7), 1884-1891. doi: 10.1063/1.866635

Pope, S. B. (2000). *Turbulent flows*. Cambridge university press.

Porta Mana, P., & Zanna, L. (2014). Toward a stochastic parameterization of ocean mesoscale eddies. *Ocean Modelling*, *79*, 1-20. doi: 10.1016/j.ocemod.2014.04.002

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

Recht, B., Roelofs, R., Schmidt, L., & Shankar, V. (2018). Do cifar-10 classifiers generalize to cifar-10? *arXiv preprint arXiv:1806.00451*.

Redi, M. H. (1982). Oceanic isopycnal mixing by coordinate rotation. *Journal of Physical Oceanography*, *12*(10), 1154–1158.

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International conference on medical image computing and computer-assisted intervention* (pp. 234–241).

Rosenblatt, M. (1956). Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics*, *27*(3), 832–837.

Ross, A. S. (2022, June). *pyqg subgrid forcing datasets.* Zenodo. Retrieved from `https://doi.org/10.5281/zenodo.6609035` doi: 10.5281/zenodo.6609035

Ross, A. S., Perezhogin, P., & Zanna, L. (2022, June). *m2lines/pyqg_parameterization_benchmarks: v1.0.1.* Zenodo. Retrieved from `https://doi.org/10.5281/zenodo.6656313` doi: 10.5281/zenodo.6656313

Rudy, S. H., Brunton, S. L., Proctor, J. L., & Kutz, J. N. (2017). Data-driven discovery of partial differential equations. *Science Advances*, *3*(4), e1602614. Retrieved from `https://www.science.org/doi/abs/10.1126/sciadv.1602614` doi: 10.1126/sciadv.1602614

Sagaut, P. (2006). *Large eddy simulation for incompressible flows: an introduction*. Springer Science & Business Media.

Sagaut, P., & Grohens, R. (1999). Discrete filters for large eddy simulation. *International Journal for Numerical Methods in Fluids*, *31*(8), 1195–1220.

Salmon, R. (1980). Baroclinic instability and geostrophic turbulence. *Geophysical & Astrophysical Fluid Dynamics*, *15*(1), 167-211. doi: 10.1080/03091928008241178

Schmidt, M., & Lipson, H. (2009). Distilling free-form natural laws from experimental data. *Science*, *324*(5923), 81-85. doi: 10.1126/science.1165893

Schneider, T., Teixeira, J., Bretherton, C. S., Brient, F., Pressel, K. G., Schär, C., & Siebesma, A. P. (2017). Climate goals and computing the future of clouds. *Nature Climate Change*, *7*(1), 3–5. Retrieved from `https://doi.org/10.1038/nclimate3190` doi: 10.1038/nclimate3190

Shevchenko, I., & Berloff, P. (2021). On a minimum set of equations for parameterisations in comprehensive ocean circulation models. *Ocean Modelling*, *168*, 101913.

Smagorinsky, J. (1963). General circulation experiments with the primitive equations: I. the basic experiment. *Monthly weather review*, *91*(3), 99–164.

Springenberg, J. T., Dosovitskiy, A., Brox, T., & Riedmiller, M. (2014). Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*.

Stephens, T. (2019). *Genetic Programming in Python, with a scikit-learn inspired API.* Retrieved from `https://gplearn.readthedocs.io/en/stable`

Stevens, B., & Bony, S. (2013). What are climate models missing? *Science*, *340*, 1053–1054. doi: 10.1126/science.1237554

Stoffer, R., Van Leeuwen, C. M., Podareanu, D., Codreanu, V., Veerman, M. A., Janssens, M., ... Van Heerwaarden, C. C. (2021). Development of a large-eddy simulation subgrid model based on artificial neural networks: a case study of turbulent channel flow. *Geoscientific Model Development*, *14*(6), 3769–3788.

Subel, A., Chattopadhyay, A., Guan, Y., & Hassanzadeh, P. (2021). Data-driven subgrid-scale modeling of forced burgers turbulence using deep learning with generalization to higher reynolds numbers via transfer learning. *Physics of Fluids*, *33*(3), 031702.

Subel, A., Guan, Y., Chattopadhyay, A., & Hassanzadeh, P. (2022). Explaining the physics of transfer learning a data-driven subgrid-scale closure to a different turbulent flow. *arXiv preprint arXiv:2206.03198*.

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.

Turing, A. (1950, 10). I.—COMPUTING MACHINERY AND INTELLIGENCE. *Mind*, *LIX*(236), 433-460. Retrieved from `https://doi.org/10.1093/mind/LIX.236.433` doi: 10.1093/mind/LIX.236.433

Vallis, G. K. (2017). *Atmospheric and oceanic fluid dynamics.* Cambridge University Press.

Vallis, G. K., & Hua, B.-l. (1988). Eddy viscosity of the anticipated potential vorticity method. *Journal of Atmospheric Sciences*, *45*(4), 617–627.

Xie, C., Wang, J., & Weinan, E. (2020). Modeling subgrid-scale forces by spatial artificial neural networks in large eddy simulation of turbulence. *Physical Review Fluids*, *5*(5), 054606.

Xing, H., Zhang, J., Ma, W., & Wen, D. (2022). Using gene expression programming to discover macroscopic governing equations hidden in the data of molecular simulations. *Physics of Fluids*, *34*(5), 057109. doi: 10.1063/5.0090134

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. Retrieved from `https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2020GL091363` (e2020GL091363 2020GL091363) doi: https://doi.org/10.1029/2020GL091363

Yuval, J., & O'Gorman, P. A. (2021). Neural-network parameterization of subgrid momentum transport in the atmosphere. *Earth and Space Science Open Archive*, 15. doi: 10.1002/essoar.10507557.1

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., & Bolton, T. (2021). Deep learning of unresolved turbulent ocean pro-

cesses in climate models. In *Deep learning for the earth sciences* (p. 298-306). John Wiley Sons, Ltd. doi: https://doi.org/10.1002/9781119646181.ch20

Zhang, S., & Lin, G. (2018). Robust data-driven discovery of governing physical laws with error bars. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *474*(2217), 20180305. Retrieved from `https://royalsocietypublishing.org/doi/abs/10.1098/rspa.2018.0305` doi: 10.1098/rspa.2018.0305

Zhou, Z., He, G., Wang, S., & Jin, G. (2019). Subgrid-scale model for large-eddy simulation of isotropic turbulent flows using an artificial neural network. *Computers Fluids*, *195*, 104319. doi: https://doi.org/10.1016/j.compfluid.2019.104319