DATA-DRIVEN MULTISCALE MODELING OF SUBGRID PARAMETERIZATIONS IN CLIMATE MODELS

Karl Otness, Laure Zanna & Joan Bruna
Courant Institute of Mathematical Sciences
New York University
`karl.otness@nyu.edu`

ABSTRACT

Subgrid parameterizations that represent physical processes occurring below the resolution of current climate models are any important component in producing accurate, long-term predictions for the climate. A variety of approaches have been tested to design these components, including deep learning methods. In this work, we evaluate a proof of concept illustrating a multiscale approach to this prediction problem. We train neural networks to predict subgrid forcing values on a testbed model and examine improvements in prediction accuracy which can be obtained by using additional information in both fine-to-coarse and coarse-to-fine directions.

1 INTRODUCTION

Climate models, which simulate the long-term evolution of the Earth's atmosphere, oceans, and terrestrial weather, are critical tools for projecting the impacts of climate change around the globe. Due to limits on available computational resources, these models must be run at a coarsened spatial resolution which cannot resolve all physical processes relevant to the climate system (Fox-Kemper et al., 2014). To reflect the contribution of these subgrid-scale processes, closure models are added to climate models to provide the needed subgrid-scale forcing. These parameterizations modeling the contribution of these fine-scale dynamics are critical to high quality and accurate long term predictions (Ross et al., 2023; Fox-Kemper et al., 2019). A variety of approaches to designing these parameterizations have been tried ranging from hand-designed formulations (Smagorinsky, 1963), to modern machine learning with genetic algorithms (Ross et al., 2023), or neural networks trained on collected snapshots (Zanna & Bolton, 2020; Guillaumin & Zanna, 2021; Maulik et al., 2019; Perezhogin et al., 2023) or in an online fashion through the target simulation (Frezat et al., 2022).

In this work we examine the impact of decomposing the problem of predicting subgrid forcings into prediction problems across scales. The problem of learning subgrid forcing is inherently multiscale; the subgrid dynamics which must be restored represent the impact of the subgrid and resolved scales on each other. Closure models for climate are designed to be resolution-aware (Jansen et al., 2019), but even so existing deep learning subgrid models do not explicitly leverage the interactions between scales, leaving it to the neural networks to implicitly learn these relationships. Explicitly including this structure as part of a deep learning approach may help regularize the learned closure models and support learning in regimes with limited training data or in the presence of underlying uncertainty. We explore the impact of this decomposition, providing proof of concept results illustrating the potential of imposing this prediction structure on a simple fully-convolutional neural network closure model.

2 APPROACH

We consider the problem of learning subgrid forcings for an idealized fluid model. In particular we study a two layer quasi-geostrophic model as implemented in PyQG (Abernathey et al., 2022) which we have ported to JAX[1] (Bradbury et al., 2018). In this model the variable of interest is the potential vorticity q which evolves in two layers each with two dimensions and periodic boundary conditions. The model can be evaluated with a configurable grid resolution and states can be ported to lower resolutions by coarse-graining and filtering. Further details of this model are included in Appendix A.

[1]The ported QG model is available at `https://github.com/karlotness/pyqg-jax`

To generate ground truth data, we run the model at a very high ("true") resolution. This produces trajectories $q_{\text{true}}(t)$ and time derivatives $\partial q_{\text{true}}(t)/\partial t$. Next we generate training data at a high resolution by applying a coarsening and filtering operator C giving variables $\bar{q} \triangleq C(q)$. Given nonlinearities in the model, this coarsening does not commute with the dynamics of the model. To correct for this we must apply a subgrid forcing term S:

$$S \triangleq \overline{\frac{\partial q}{\partial t}} - \frac{\partial \bar{q}}{\partial t}. \tag{1}$$

Note that formally the forcing S is a function of the state q_{true}. In a climate modeling application we do not have access to this variable and so we train a model $f_\theta(\bar{q}) \approx S$ which may be stochastic.

We continue this process, introducing another downscaling[2] operator D and upscaling D^+. Taking $q_{\text{hr}} \triangleq \bar{q}$ as our high resolution samples, we produce low resolution samples $q_{\text{lr}} \triangleq D(q_{\text{hr}})$ and $S_{\text{lr}} \triangleq D(S)$. For any of these quantities v we have a decomposition $v = D^+Dv + v'$, where v' are the details removed by D. Our experiments thus involve three resolutions, from fine to coarse: a "true" resolution; a high resolution, hr; and a low resolution, lr. The closures S try to update hr to match the "true" resolution.

Just as predicting S from q_{true} is fully deterministic, while predicting it from q_{hr} involves uncertainty, we anticipate a similar trend to hold for $D(S)$. In other words, predicting $D(S)$ from q_{hr} should be easier than predicting $D(S)$ directly from q_{lr}. Then, using this coarse-grained prediction $D(S)$ as a foundation, we can learn to predict only the missing details and add them. This process splits the problem of predicting S into two phases: (1) a "downscale" prediction to form $D(S)$, and (2) a "buildup" prediction combining q_{hr} and $D(S)$ to predict S, adding the missing details. This decomposition takes advantage of self-similarity in the closure problem to pass information between the coarse and fine scales and improve predictions.

As a proof of concept of this approach we test the "downscale" and "buildup" processes against baselines without the additional information. In both experiments, all required inputs are provided by an oracle backed by a ground truth data set. For all experiments we train a feedforward convolutional neural network to perform the prediction task, three copies of each network. These neural networks have one of two architectures, a "small" architecture as used in Guillaumin & Zanna (2021) and a "large" architecture with larger convolution kernels. Details of these experiments are provided below, and information on the network architectures and training are included in Appendix B.

Downscale prediction: we compare the task of predicting $S_{\text{lr}} \triangleq D(S)$ with access to high resolution information q_{hr} or restricted to low resolution q_{lr}. This provides an estimate of the advantage gained by predicting the target forcing with access to details at a scale finer than that of the network's output. We train two networks with the same architecture to perform one of two prediction tasks:

$$D \circ f_\theta^{\text{downscale}}(q_{\text{hr}}) \approx S_{\text{lr}} \qquad \text{and} \qquad D \circ f_\theta^{\text{across}} \circ D^+(q_{\text{lr}}) \approx S_{\text{lr}}. \tag{2}$$

However, to ensure that the convolution kernels process information at the same spatial size, and differ only in the spectral scales included, we first upsample all inputs to the same fixed size using a spectral upscaling operator D^+ described in Appendix C.

We train networks to carry out this prediction task between three scales: resolutions of 128×128, 96×96, and 64×64, chosen so that the system requires closure (there are sufficient dynamics below the grid-scale cutoff), but does not diverge (Ross et al., 2023). We test all combinations of distinct high and low resolutions. The full prediction process including the re-sampling operators is illustrated in Figure 1 and experimental results are included in Table 1 discussed below.

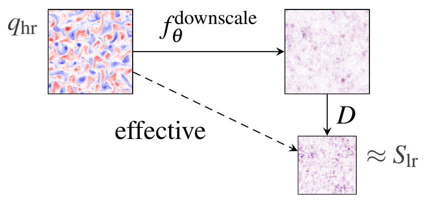
Buildup prediction: we also test a prediction problem in the opposite direction, predicting finer-scale details with access to lower-resolution predictions, similar to a learned super-resolution process used in recent generative modeling works (Singer et al., 2022; Ho et al., 2022). We train neural networks:

$$f_\theta^{\text{buildup}}(q_{\text{hr}}, D^+(S_{\text{lr}})) \approx S_{\text{hr}} - D^+(S_{\text{lr}}) \qquad \text{and} \qquad f_\theta^{\text{direct}}(q_{\text{hr}}) \approx S_{\text{hr}}\ , \tag{3}$$

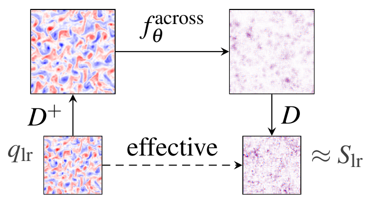
where $S_{\text{hr}} - D^+(S_{\text{lr}})$ are the details of S_{hr} which are not reflected in S_{lr}. The additional input S_{lr} is given by an oracle using ground truth data in the training and evaluation sets.

This experiment estimates the value in having a high-quality, higher-confidence prediction S_{lr}, in addition to q_{hr}, when predicting the details of S_{hr}. That is, the experiment estimates the value in

[2]We use "downscale" and "downscaling" to refer to *coarsening* a target variable, removing finer scales.

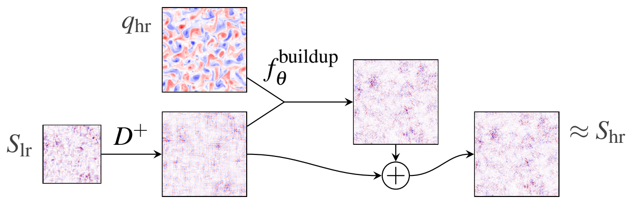


(a) Downscale prediction

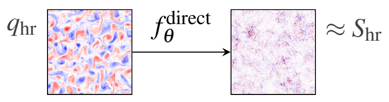


(b) Across prediction

Figure 1: Downscale vs. across prediction tasks. The networks referenced in Equation 2 are combinations of an inner network f_θ with the fixed rescaling operators D, D^+. The overall prediction is indicated with a dashed line.



(a) Buildup prediction



(b) Direct prediction

Figure 2: Buildup vs. direct prediction. The networks in Equation 3 are combinations of the networks f_θ with the indicated fixed operations. In Figure 2a f_θ predicts the details which are combined with S_{lr} from an oracle.

starting the prediction of S_{hr} by first locking in a coarse-grained version of the target, and separately enhancing it with finer-scale features.

As in the other experiment, we carry out this upscaling between three resolutions: 128, 96, and 64, stepping between all distinct combinations of low and high resolutions. The two prediction tasks are illustrated in Figure 2 and results are included in Table 2, discussed below.

Combined prediction: we combine the networks trained in the "downscale" and "buildup" experiments, passing the downscale prediction as an input to the buildup network. This removes the oracle providing predictions S_{lr}. We test all pairings of neural network architecture sizes and re-trained networks. Averaged results for each of our metrics are included in Table 3 and are discussed below.

3 Experimental Results

For each of the prediction tasks described in Section 2, we train three neural networks. Once trained, we evaluate their performance on a held-out evaluation set measuring performance with three metrics: a mean squared error (MSE), a relative ℓ_2 loss, and a relative ℓ_2 of the spectra of the predictions. Details of these metrics are provided in Appendix D.

Table 1 shows the results for the downscale experiments, comparing against "across" prediction which accesses only coarse-scale information. In these results we observe an advantage to performing the predictions with access to higher-resolution data (the "downscale" columns), suggesting potential advantages and a decrease in uncertainty in such predictions.

Results for experiments examining prediction in the opposite direction—predicting a high-resolution forcing with access to a low-resolution copy of the target from an oracle—are included in Table 2. We also observe an advantage in this task from having access to the additional information. The low resolution input in the "buildup" experiments yields lower errors at evaluation. This advantage is greater when the additional input is closer in scale to the target output. The predictions building up from 96×96 to 128×128 have lower errors than those which access an additional 64×64 input. This is not unexpected given that the input with nearer resolution resolves more of the target value, leaving fewer details which need to be predicted by the network.

The results of the combined experiments in Table 3 illustrate the early potential of this approach. The two networks being combined were trained separately using only ground-truth inputs. Even so, their

NN Size	Metric	$128 \to 96$		$128 \to 64$		$96 \to 64$	
		Downscale	Across	Downscale	Across	Downscale	Across
Small	MSE	0.054	0.073	0.003	0.006	0.034	0.060
	Rel ℓ_2	0.315	0.365	0.406	0.633	0.360	0.476
	Rel Spec ℓ_2	0.123	0.159	0.169	0.480	0.151	0.256
Large	MSE	0.048	0.069	0.003	0.006	0.031	0.058
	Rel ℓ_2	0.295	0.352	0.405	0.628	0.343	0.465
	Rel Spec ℓ_2	0.099	0.144	0.177	0.485	0.132	0.243

Table 1: Evaluation results for downscale vs. across generation. In all metrics, lower is better. The numbers in the first row of the table heading show the different scales involved in both prediction tasks.

NN Size	Metric	Buildup	Buildup	Direct	Buildup	Direct
		$64 \to 128$	$96 \to 128$	128	$64 \to 96$	96
Small	MSE	0.087	0.041	0.096	0.065	0.109
	Rel ℓ_2	0.303	0.207	0.318	0.262	0.335
	Rel Spec ℓ_2	0.132	0.067	0.139	0.099	0.156
Large	MSE	0.064	0.019	0.075	0.040	0.083
	Rel ℓ_2	0.259	0.140	0.280	0.203	0.289
	Rel Spec ℓ_2	0.082	0.026	0.091	0.049	0.106

Table 2: Evaluation results from buildup vs. direct experiments. In all metrics, lower is better. The numbers in the second row of the table heading show the different scales involved in both prediction tasks.

combination produces small improvements in cases with near-scale prediction. We include plots showing the distribution of values listed in each table as additional results in Appendix E.

4 Conclusion

Our proof of concept experiments in this work illustrate the potential advantages from decomposing the subgrid forcing problem into one across scales. We see this as an approach which may have regularization advantages, explicitly representing multiscale aspects of this prediction problem, and supporting learning in scarce data regimes and better handling underlying uncertainty in this task.

In our continuing work we will further investigate combining these prediction tasks to increase robustness to perturbations in the downscale predictions and remove the networks' reliance on an oracle to provide all inputs. We anticipate that augmentations during training can improve the performance of the combined experiments. We will further work to quantify regularization benefits from this approach in limited-data regimes, investigate other ways to structure this multiscale prediction task, and assess the possibility of incorporating more scale levels and other scale representations.

Acknowledgments

This research was made possible by Schmidt Futures, a philanthropic initiative founded by Eric and Wendy Schmidt, as part of its Virtual Earth System Research Institute (VESRI).

References

Ryan Abernathey, rochanotes, Andrew Ross, Malte Jansen, Ziwei Li, Francis J. Poulin, Navid C. Constantinou, Anirban Sinha, Dhruv Balwada, SalahKouhen, Spencer Jones, Cesar B Rocha, Christopher L. Pitt Wolfe, Chuizheng Meng, Hugo van Kemenade, James Bourbeau, James Penn, Julius Busecke, Mike Bueti, and Tobias. pyqg: v0.7.2, 5 2022. URL `https://doi.org/10.5281/zenodo.6563667`.

Igor Babuschkin, Kate Baumli, Alison Bell, Surya Bhupatiraju, Jake Bruce, Peter Buchlovsky, David Budden, Trevor Cai, Aidan Clark, Ivo Danihelka, Claudio Fantacci, Jonathan Godwin, Chris Jones, Ross Hemsley, Tom Hennigan, Matteo Hessel, Shaobo Hou, Steven Kapturowski,

NN Sizes Downscale $\to$ Buildup	Metric	Combined $128 \to 64 \to 128$	Combined $128 \to 96 \to 128$	Direct 128	Combined $96 \to 64 \to 96$	Direct 96
Small $\to$ Small	MSE	0.097	**0.093**	0.096	**0.105**	0.109
	Rel ℓ_2	**0.317**	**0.312**	0.318	**0.329**	0.335
	Rel Spec ℓ_2	0.143	**0.121**	0.139	**0.153**	0.156
Large $\to$ Small	MSE	0.097	**0.087**	0.096	**0.102**	0.109
	Rel ℓ_2	0.318	**0.300**	0.318	**0.324**	0.335
	Rel Spec ℓ_2	0.146	**0.107**	0.139	**0.147**	0.156
Small $\to$ Large	MSE	**0.074**	**0.071**	0.075	**0.081**	0.083
	Rel ℓ_2	**0.276**	**0.271**	0.280	**0.285**	0.289
	Rel Spec ℓ_2	0.095	0.100	0.091	0.117	0.106
Large $\to$ Large	MSE	**0.074**	**0.066**	0.075	**0.078**	0.083
	Rel ℓ_2	**0.277**	**0.260**	0.280	**0.281**	0.289
	Rel Spec ℓ_2	0.097	**0.085**	0.091	0.111	0.106

Table 3: Evaluation results combining "downscale" and "buildup" networks, eliminating the input forcing oracle. Bolded values show where the combined network outperformed the associated baseline (repeated from Table 2).

Thomas Keck, Iurii Kemaev, Michael King, Markus Kunesch, Lena Martens, Hamza Merzic, Vladimir Mikulik, Tamara Norman, John Quan, George Papamakarios, Roman Ring, Francisco Ruiz, Alvaro Sanchez, Rosalia Schneider, Eren Sezener, Stephen Spencer, Srivatsan Srinivasan, Luyu Wang, Wojciech Stokowiec, and Fabio Viola. The DeepMind JAX Ecosystem, 2020. URL `http://github.com/deepmind`.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `https://github.com/google/jax`.

Baylor Fox-Kemper, Scott Bachman, Brodie Pearson, and Scott Reckinger. Principles and advances in subgrid modelling for eddy-rich simulations. In *CLIVAR Exchanges (WGOMD Workshop on High Resolution Climage Modeling)*, volume 19, pp. 42–46, 2014.

Baylor Fox-Kemper, Alistair Adcroft, Claus W. Böning, Eric P. Chassignet, Enrique Curchitser, Gokhan Danabasoglu, Carsten Eden, Matthew H. England, Rüdiger Gerdes, Richard J. Greatbatch, Stephen M. Griffies, Robert W. Hallberg, Emmanuel Hanert, Patrick Heimbach, Helene T. Hewitt, Christopher N. Hill, Yoshiki Komuro, Sonya Legg, Julien Le Sommer, Simona Masina, Simon J. Marsland, Stephen G. Penny, Fangli Qiao, Todd D. Ringler, Anne Marie Treguier, Hiroyuki Tsujino, Petteri Uotila, and Stephen G. Yeager. Challenges and prospects in ocean circulation models. *Frontiers in Marine Science*, 6, 2019. ISSN 2296-7745. doi: 10.3389/fmars.2019.00065. URL `https://www.frontiersin.org/articles/10.3389/fmars.2019.00065`.

Hugo Frezat, Julien Le Sommer, Ronan Fablet, Guillaume Balarac, and Redouane Lguensat. A posteriori learning for quasi-geostrophic turbulence parametrization. *Journal of Advances in Modeling Earth Systems*, 14(11):e2022MS003124, 2022. doi: 10.1029/2022MS003124. URL `https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2022MS003124`. e2022MS003124 2022MS003124.

Arthur P. Guillaumin and Laure Zanna. Stochastic-deep learning parameterization of ocean momentum forcing. *Journal of Advances in Modeling Earth Systems*, 13(9):e2021MS002534, 2021. doi: 10.1029/2021MS002534. URL `https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2021MS002534`. e2021MS002534 2021MS002534.

Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P. Kingma, Ben Poole, Mohammad Norouzi, David J. Fleet, and Tim Salimans. Imagen video: High definition video generation with diffusion models. *arXiv Preprint*, 2022. URL `https://arxiv.org/abs/2210.02303`.

Malte F. Jansen, Alistair Adcroft, Sina Khani, and Hailu Kong. Toward an energetically consistent, resolution aware parameterization of ocean mesoscale eddies. *Journal of Advances in Modeling*

Earth Systems, 11(8):2844–2860, 2019. doi: https://doi.org/10.1029/2019MS001750. URL https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2019MS001750.

Patrick Kidger and Cristian Garcia. Equinox: neural networks in JAX via callable PyTrees and filtered transformations. *Differentiable Programming workshop at Neural Information Processing Systems 2021*, 2021.

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

Pavel Perezhogin, Laure Zanna, and Carlos Fernandez-Granda. Generative data-driven approaches for stochastic subgrid parameterizations in an idealized ocean model. *arXiv preprint*, 2023. URL https://arxiv.org/abs/2302.07984.

Andrew Ross, Ziwei Li, Pavel Perezhogin, Carlos Fernandez-Granda, and Laure Zanna. Benchmarking of machine learning ocean subgrid parameterizations in an idealized model. *Journal of Advances in Modeling Earth Systems*, 15(1):e2022MS003258, 2023. doi: 10.1029/2022MS003258. URL https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2022MS003258. e2022MS003258 2022MS003258.

Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, Devi Parikh, Sonal Gupta, and Yaniv Taigman. Make-a-video: Text-to-video generation without text-video data. *arXiv Preprint*, 2022. URL https://arxiv.org/abs/2209.14792.

J. Smagorinsky. General circulation experiments with the primitive equations: I. the basic experiment. *Monthly Weather Review*, 91(3):99 – 164, 1963. doi: 10.1175/1520-0493(1963)091<0099:GCEWTP>2.3.CO;2.

Laure Zanna and Thomas Bolton. Data-driven equation discovery of ocean mesoscale closures. *Geophysical Research Letters*, 47(17):e2020GL088376, 2020. doi: 10.1029/2020GL088376. URL https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2020GL088376. e2020GL088376 10.1029/2020GL088376.

Juntang Zhuang, Tommy Tang, Yifan Ding, Sekhar C Tatikonda, Nicha Dvornek, Xenophon Papademetris, and James Duncan. Adabelief optimizer: Adapting stepsizes by the belief in observed gradients. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 18795–18806. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/d9d4f495e875a2e075a1a4a6e1b9770f-Paper.pdf.

A Quasi-Geostrophic Model

For our experiments we target the two-layer quasi-geostrophic model implemented in PyQG which is a simplified approximation of fluid dynamics (Abernathey et al., 2022). This model follows the evolution of a potential vorticity q, divided into two layers $q = [q_1, q_2]$. This system is pseudo-spectral and has periodic boundaries along the edges of each layer. The evolution of the quantities in Fourier space (indicated by a hat) is:

$$\frac{\partial \hat{q}_1}{\partial t} = -\widehat{J(\psi_1, q_1)} - ik\beta_1 \hat{\psi}_1 + \widehat{\mathrm{ssd}} \tag{4}$$

$$\frac{\partial \hat{q}_2}{\partial t} = -\widehat{J(\psi_2, q_2)} - ik\beta_2 \hat{\psi}_2 + r_{\mathrm{ek}} \kappa^2 \hat{\psi}_2 + \widehat{\mathrm{ssd}} \tag{5}$$

where $J(A,B) \triangleq A_x B_y - A_y B_x$, "ssd" is a small scale dissipation, and the quantity ψ is related to q by:

$$\begin{bmatrix} -(\kappa^2 + F_1) & F_1 \\ F_2 & -(\kappa^2 + F_2) \end{bmatrix} \begin{bmatrix} \hat{\psi}_1 \\ \hat{\psi}_2 \end{bmatrix} = \begin{bmatrix} \hat{q}_1 \\ \hat{q}_2 \end{bmatrix}. \tag{6}$$

The values κ are the radial wavenumbers $\sqrt{k^2 + l^2}$ while k and l are wavenumbers in the zonal and meridional directions (the axis-aligned directions in our grid), respectively (Ross et al., 2023).

We use the “eddy” configuration from Ross et al. (2023) which sets the following values for model constants:

$$
\begin{aligned}
r_{\text{ek}} &= 5.787 \times 10^{-7} \\
\delta &= \frac{H_1}{H_2} = 0.25 \\
\beta &= 1.5 \times 10^{-11} \\
r_d &= 15000 \\
F_1 &= \frac{1}{r_d^2(1+\delta)} \\
F_2 &= \delta F_1 \\
W &= L = 10^6
\end{aligned}
$$

where H_1, H_2 are the heights of each of the two layers of q and r_d is a deformation radius. For more information on the model configuration, consult Ross et al. (2023) and the documentation for the PyQG package.

We generate our data at a “true” resolution on a grid of dimension 256×256 using the PyQG default third order Adams-Bashforth method for time stepping. We use a time step of $\Delta t = 3600$ generating 86400 steps from which we keep every eighth leaving 10800 per trajectory. Our training set consists of 100 such trajectories, and our evaluation set contains 10.

Each step produces a ground truth potential vorticity q_{true} along with a spectral time derivative $\partial \hat{q}_{\text{true}}/\partial t$. From these we apply our family of coarsening operators C (described in Appendix C to produce filtered and coarsened values $q_{\text{lr}} \triangleq C_{\text{lr}}(q_{\text{true}})$ at resolutions of 128×128, 96×96, and 64×64.

For each of these, we recompute spectral time derivatives in a coarsened PyQG model $\partial \hat{q}_{\text{lr}}/\partial t$, we pass each time derivative to spatial variables and compute the target forcing for this scale:

$$S_{\text{lr}} = C_{\text{lr}}\left(\frac{\partial q_{\text{true}}}{\partial t}\right) - \frac{\partial q_{\text{lr}}}{\partial t}.$$

These forcings—at each of the three scales—along with the high resolution variables are stored in the training and evaluation sets for each step.

B Network Architecture and Training

We use the feedforward CNN architecture from Guillaumin & Zanna (2021) without batch norm as our standard “small” architecture. The “large” size roughly doubles the size of each convolution kernel. This produces the architectures listed in Table 4. We use ReLU activations between each convolution. Each convolution is performed with periodic padding, matching the boundary conditions of the system. All convolutions are with bias. The input and output channel counts are determined by the inputs of the network. Each input and output quantity has two layers, each of which is handled as a separate channel. These parameters are adjusted for each task to accommodate the inputs and make the required predictions. We implement our networks with Equinox (Kidger & Garcia, 2021).

We train each network with the adabelief optimizer (Zhuang et al., 2020) following a cosine warmup learning rate schedule, both implemented in Optax (Babuschkin et al., 2020). The learning rate schedule is updated per step and spends one epoch performing a linear ramp up from 0 to a peak learning rate, then the remainder of training follows a cosine decay to a final learning rate of 0. The peak learning rate was 1×10^{-3} for the small networks and 2.22×10^{-4} for the large networks. The networks are trained to minimize MSE loss. Large chunks of 10850 steps are sampled with replacement from the dataset which is pre-shuffled uniformly. Then each of these chunks is shuffled again and divided into batches of size 256 without replacement. One epoch corresponds to one expected pass through the data set. We train the small networks for 66 epochs, and the large networks for 48 epochs. We store the network weights which produced the lowest training set loss and use these for evaluation.

Conv. Layer	Chans. Out	Small Kernel Size	Large Kernel Size
1	128	(5,5)	(9,9)
2	64	(5,5)	(9,9)
3	32	(3,3)	(5,5)
4	32	(3,3)	(5,5)
5	32	(3,3)	(5,5)
6	32	(3,3)	(5,5)
7	32	(3,3)	(5,5)
8	out layers	(3,3)	(5,5)

Table 4: Architecture specifications for each neural network. Convolution kernel sizes vary between the architecture sizes. The channel counts are adjusted to accommodate the inputs and outputs of each task.

For all input and target data, we compute empirical means and standard deviations and standardize the overall distributions by these values before passing them to the network. The means and standard deviations from the training set are used in evaluation as well.

C Coarsening Operators

In this work we make use of two families of coarsening operators to transform system states across scales. The first, denoted C, is used when generating our data. This operator is applied to the "true" resolution system outputs q_{true} and $\partial q_{\text{true}}/\partial t$ to produce training and evaluation set samples as well as target forcings S. The second operator D (with associated upscaling D^+) is applied as a part of each prediction task to adjust scales around the neural networks as needed. These are the operators referenced in Figure 1 and Figure 2.

Each of these operators is built around a core spectral truncation operation, $\mathcal{D}$. For an input resolution hr and an output resolution lr, this operator truncates the 2D-Fourier spectrum to the wavenumbers which are resolved at the output resolution, then spatially resamples the resulting signal for the target size lr. These operators also apply a scalar multiplication to adjust the range of the coarsened values. We define a ratio $\rho \triangleq \text{hr}/\text{lr}$.

C.1 Data Filtering

The data filtering operator C is "Operator 1" as described in Ross et al. (2023). It is a combination of the truncation operator $\mathcal{D}$ with a spectral filter $\mathcal{F}$

$$C \triangleq \rho^{-2} \cdot \mathcal{F} \circ \mathcal{D}$$

where the filter $\mathcal{F}$ acts on the 2D-Fourier spectrum of the truncated value. $\mathcal{F}$ is defined in terms of the radial wavenumber $\kappa = \sqrt{k^2 + l^2}$ where k and l are the wavenumbers in each of the two dimensions of the input. For an input $\hat{v}_\kappa$ at radial wavenumber κ we define:

$$\mathcal{F}(\hat{v}_\kappa) = \begin{cases} \hat{v}_\kappa & \text{if } \kappa \leq \kappa^c \\ \hat{v}_\kappa \cdot e^{-23.6(\kappa-\kappa^c)^4 \Delta x_{\text{lr}}^4} & \text{if } \kappa > \kappa^c \end{cases}$$

where $\Delta x_{\text{lr}} \triangleq L/\text{lr}$ (L is a system parameter see Appendix A for details), and $\kappa^c \triangleq (0.65\pi)/\Delta x_{\text{lr}}$ is a cutoff wavenumber where decay begins.

C.2 Rescaling Operator

For scale manipulations as part of our learned model we make use of a scaled spectral truncation operator. We define a downscaling operator D as well as an upscaling operator D^+:

$$D \triangleq \rho^{-2}\mathcal{D} \qquad \text{and} \qquad D^+ \triangleq \rho^2 \mathcal{D}^T. \tag{7}$$

Note that D^+ is a right side inverse $DD^+ = I$, and that D^+ is the pseudoinverse $D^+ = D(DD^T)^{-1}$ because $\mathcal{D}\mathcal{D}^T = I$. This operator omits the filtering $\mathcal{F}$ performed as part of coarsening operator C to avoid numerical issues when inverting the additional spectral filtering.

D Evaluation Metrics

In this work we consider three metrics for evaluating the quality of network predictions: a mean squared error (MSE), a relative ℓ_2 loss, and a relative ℓ_2 of the spectra of the predictions. The MSE is a standard mean squared error evaluated over each sample and averaged. The other two metrics are derived from Perezhogin et al. (2023) (where they were called $\mathcal{L}_{\text{rmse}}$ and $\mathcal{L}_{\text{S}}$). These were originally designed to measure performance for stochastic subgrid forcings. Here we use the two metrics from that work which do not collapse to trivial results for deterministic models.

These metrics are defined as:

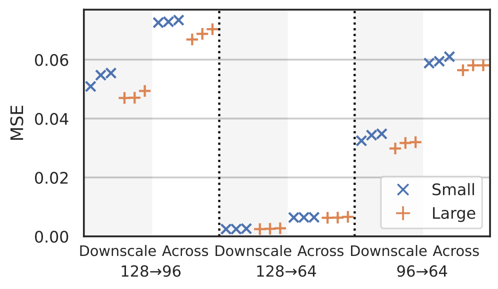
$$\text{Rel } \ell_2 = \frac{\|S - \tilde{S}\|_2}{\|S\|} \qquad \text{and} \qquad \text{Rel Spec } \ell_2 = \frac{\|\text{sp}(S) - \text{sp}(\tilde{S})\|_2}{\|\text{sp}(S)\|_2}$$

where S is the true target forcing, $\tilde{S}$ is a neural network prediction being evaluated, and sp is the isotropic power spectrum. See `calc_ispec` in PyQG for calculation details (Abernathey et al., 2022). Each of these three metrics is averaged across the same batch of 1024 samples selected at random from the set of held out trajectories in the evaluation set.

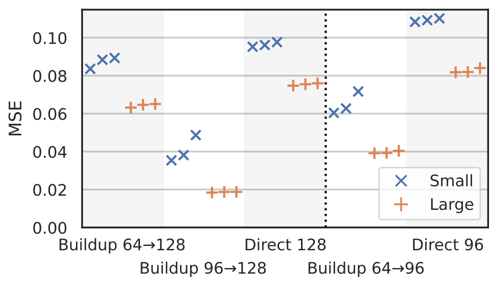
E Additional Results

Figure 3 and Figure 4 illustrate the results from the tables in the paper body. They illustrate the individual MSE evaluation results which are averaged when producing the results in the tables.

We observe relatively high variance in the combined experiments (see Figure 4) despite improved mean performance (as reported in Table 3). We attribute most of this to the fact that each component network was trained on ground-truth data with no corruption. At evaluation time, the buildup network is exposed to predicted data from the downscale network and may not handle the resulting errors well. We anticipate that changes to training, such as the inclusion of simulated noise, should help reduce the variance we observe here.



(a) Downscale vs. across MSE



(b) Buildup vs. direct MSE

Figure 3: Evaluation results from both experiments for the MSE metric. These are the same numbers which are reported as averages in Table 1 and Table 2. The plot here shows the three samples—one from each trained network—used to compute the means.

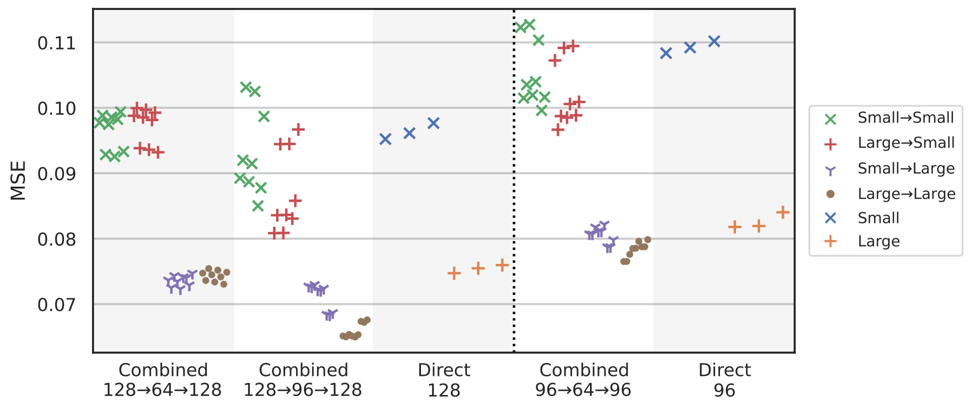


Figure 4: Evaluation results from the combined experiments for the MSE metric. These are the same values reported as averages in Table 3. The plot here shows nine samples for the “combined” tests (one for each pairing of networks) and three for the “direct” experiments used as a baseline.