A Metrics

Root mean square error The root mean square (RMSE) provides a measure of pointwise accuracy. While such a measure may not be useful for long term rollouts, due to the chaotic divergence of the system: even infinitesimal perturbations causes trajectories to diverge exponentially; it is a good proxy for short-term accuracy. Given a predicted LES state v^{pred} and the filtered DNS reference trajectory $\overline{u}$, we compute the root mean squared error over the domain as:

$$\text{RMSE}(v^{\text{pred}}, \overline{u}) = \sqrt{\int_{\Omega \subset \mathbb{R}^2} (v^{\text{pred}} - \overline{u})^2 \, dx}, \tag{20}$$

where Ω is the domain of N-S system in Eq. (1). Since both v^{pred} and $\overline{u}$ are represented as degree P polynomials on each quadrilateral element (see Appendix C), the integral is computed exactly using an appropriate quadrature rule; in this case we use the Gauss-Lobatto-Legendre (GLL) quadrature rule on each 2D quadrilateral element to compute the RMSE.

Turbulent kinetic energy (TKE) spectrum The TKE spectrum provides an aggregate view of the energy content of the fluid at different scales. It is one of the key quantities used to determine the spectral accuracy of simulations in the study of turbulence [13]. In particular, the TKE spectrum captures a snapshot of the energy cascade, which is the flow of energy from the large scales to the small scales, and vice versa. The TKE is computed by taking the Fourier transform $\hat{v} = (\hat{v}_0, \hat{v}_1)$ of the velocity field $v = (v_0, v_1)$, and computing the kinetic energy in the Fourier domain

$$\text{TKE} = \frac{1}{2}(\hat{v_0}^2 + \hat{v_1}^2). \tag{21}$$

The energy spectrum is then computed as the sum of the energy content in each 2D wavenumber bucket. Finally, the energy spectrum is averaged over several hundred steps in a long term rollout.

B Data generation

For the reference dataset, we use an ensemble of N trajectories, defined on a fine grid $\mathcal{G}$. These DNS trajectories are obtained from a numerical solver using a high order spectral element spatial discretization on the domain Ω, at a temporal resolution of Δt. These trajectories are then filtered and sampled in time to obtain an ensemble of trajectories $\{\overline{u^{(i)}}\}_{i=1}^{N}$ defined on the coarse grid $\overline{\mathcal{G}}$ at a temporal resolution of $10\Delta t$. The filtered DNS contains only explicitly the large scales. However, it was obtained from the full order model so it contains the true evolution of the large scales resulting from *the interactions between the large and the small scales*.

We use the spectral element method [23] to obtain trajectories of the N-S equation 2. The solver uses a fine-grained domain $\mathcal{G}$ with E elements and polynomial order P to spatially discretize Ω. These DNS trajectories obtained at fine temporal resolution consistent with the CFL condition. See Appendix C for more details on the numerical solver. We generated trajectories of Kolmogorov flow by running $N = 32$ independent DNS runs over the domain $[0, 1]^2$. We discretized the domain uniformly into 48^2 elements and solved with a spectral element solver at polynomial order 8, and used a timestep of $\Delta t = 10^{-4}$. For each independent run, we perturbed the initial velocity field with a Gaussian field with mean uniformly in $[0, 1]^2$ and standard deviation of 0.1. We discarded the first $50,000$ steps in order to 'forget' the initialization. The next $128,000$ steps were taken to form the dataset. Then, for the LES we use the same spectral element solver but on a coarse-grained domain $\overline{\mathcal{G}}$ of 12^2 elements and a polynomial order of 4. The DNS trajectories were filtered on a per-element basis following [12] and interpolated to the coarse grid $\overline{\mathcal{G}}$. We sampled every 10th step to obtain trajectories of 12,800 steps, at a temporal resolution of $10\Delta t = 10^{-3}$. Out of the 32 independent runs, 24 of them were used for training, 4 of them for validation and 4 for the test dataset.

For the flow past cylinder dataset, we used a rectangular domain $[-10, 30] \times [-10, 10]$. The circular cylinder is placed with center at the origin $(0, 0)$ and diameter 1.0. The boundary conditions are periodic along the y-axis. The left inflow wall has an inflow of fluid at 1 units per second in the x-direction which is implemented via constant Dirichlet boundary condition. The right outflow wall has a homogeneous Neumann condition to allow the fluid to exit the domain. The domain was discretized into 292 quadrilateral elements at polynomial order 13. The temporal resolution of the

DNS simulation was $t = 10^{-4}$. A single long trajectory was generated with $350,000$ timesteps, where the first $150,000$ timesteps were thrown out.

C Spectral element Navier-Stokes solver

Spatial Discretization Recall the Navier-Stokes equations Eq. (1).

$$\partial_t u + (u \cdot \nabla) u = -\nabla p + \nu \nabla^2 u, \tag{22}$$

$$\nabla \cdot u = 0. \tag{23}$$

Following the weak formulation of the equations and spatially discretizing via the spectral element method [23]. The velocity field is discretized using a nodal basis at Gauss-Lobatto-Legendre (GLL) points on each quadrilateral element using polynomial order P, while the pressure field is discretized using a nodal basis at Gauss-Legendre (GL) points using polynomial order $P - 2$. This staggered discretization is stable and avoids spurious pressure modes in the solution.

The semi-discretized version of the Navier-Stokes equations become:

$$M \frac{du}{dt} + Cu + \nu K u - D^T p = 0, \tag{24}$$

$$-Du = 0, \tag{25}$$

where M is the diagonal mass matrix, K, D^T and D are the discrete Laplacian, gradient and divergence operators, and C represents the action of the nonlinear advection term.

Time integration The time integration is third order, so the values of u and p at the previous three timesteps are used to solve for the new values at the current timestep [23]. A third order backward differentiation formula (BDF3) is used for both linear and nonlinear terms. However, the nonlinear term Cu may require solving a nonlinear implicit relation – to avoid this, the advection term is extrapolated using the third order extrapolation formula (EX3).

Linear solves The one step forward time evolution of the fractional step method involves solving two symmetric positive definite (SPD) linear systems, one for the intermediate velocity which is not divergence free, and next for the pressure correction term. Both the linear solves involve large linear systems, which makes it it infeasible to materialize the dense matrices in memory. Hence, we use a matrix-free conjugate gradients iteration to solve them. Further speedups may be gained by using appropriate preconditioners, which is especially important to overcome the poor conditioning when scaling up to larger systems and higher orders.

Differentiation For the reverse mode differentiation, we simply solve the transposed linear systems during the backward pass. Since the systems are symmetric, we have to solve the same system with a different right hand side. We use Jax's [15] custom automatic differentiation toolbox to override the reverse mode differentiation rule.

D Neural SDE solver

The encoder stage reduces the input sequence length $E = 144$ to a a smaller sequence length $E' = 9$; see Appendix G. The neural SDE stage operates on reduced sequence length $E' = 9$, with embedding dimension 192. The drift h_Θ and diffusion g_Θ parameterizing the latent Neural SDE uses a Transformer architecture and element-wise multi-layer perceptron (MLP) respectively. The prior drift follows the same architecture as the posterior drift.

We write $\Theta = (\theta, \phi)$ where θ and ϕ are the prior and posterior parameters respectively. The SDE is solved over the unit time interval $\tau \in [0, 1]$ with the state dimension $9 \times 192 = 1728$. The prior and posterior SDEs evolve according to the following equations:

$$d\tilde{Z}_\tau = h_\theta(\tilde{Z}_\tau, \tau) d\tau + g_\theta(\tilde{Z}_\tau, \tau) \circ dW_\tau, \tag{26}$$

$$dZ_\tau = h_\phi(Z_\tau, \tau) d\tau + g_\theta(Z_\tau, \tau) \circ dW_\tau. \tag{27}$$

Note that the diffusion term g_θ is shared by the prior and posterior SDEs. Both the drift and diffusion functions are also functions of the time τ. Additionally, the posterior drift is a function of an additional context term which is simply fixed at μ_{Z_0} and does not evolve with time.

Four independent trajectories of both the prior and posterior SDEs are sampled at both training and inference time. The initial state of the prior SDE is sampled from a zero-mean Gaussian of the same dimensionality with constant diagonal variance 0.1^2. The initial state Z_0 of the approximate posterior SDE is obtained from a multivariate Gaussian distribution parameterized by the output of the encoder $\mathcal{E}$. Specifically, the encoder outputs $(\mu_{Z_0}, \sigma_{Z_0})$, and each trajectory of the SDE is sampled independently from a Gaussian with mean μ_{Z_0} and diagonal variance $\sigma^2_{Z_0}$.

KL Divergence The KL divergence term of the Neural SDE is given by the sum of the KL divergence due to the distribution of the initial value Z_0 as well as the entire trajectory $\{Z_\tau\}_{0\leq\tau\leq1}$

$$\mathrm{KL}^{\mathrm{NSDE}} = \mathrm{KL}_{Z_0} + \mathrm{KL}_{\{Z_\tau\}}. \tag{28}$$

The KL_{Z_0} term is a standard KL divergence between two Gaussians with diagonal covariances, letting σ^{prior} is a fixed hyperparameter set to 0.1, and resulting in the following:

$$\mathrm{KL}(\mathcal{N}(\mu_{Z_0}, \sigma_{Z_0}) \| \mathcal{N}(\mathbf{0}_n, \sigma^{\mathrm{prior}}\mathbf{1}_n)) = \sum_{i=1}^{n} \log \frac{\sigma^{\mathrm{prior}}}{\sigma^{(i)}_{Z_0}} + \frac{(\sigma^{(i)}_{Z_0})^2 + (\mu^{(i)}_{Z_0})^2}{2(\sigma^{\mathrm{prior}})^2}. \tag{29}$$

The KL divergence term $\mathrm{KL}_{\{Z_\tau\}}$ is given by the following integral

$$\mathrm{KL}_{\{Z_\tau\}} = \int_0^1 \frac{1}{2}\left(\frac{h_\phi(Z_\tau, \tau) - h_\theta(Z_\tau, \tau)}{g_\theta(Z_\tau, \tau)}\right)^2 d\tau, \tag{30}$$

which is computed along with the SDE solve by augmenting the state with a single dimension. The additional scalar KL contribution term is then computed by the forward SDE solve by integrating the drift given by Eq. (30) and zero diffusion.

Architectural Choices The architecture of the drift functions are based on the Transformer architecture. Each Transformer block contains a self-attention and an MLP block, each preceded by a layernorm and GeLU nonlinear activations. Four layers of Transformers were stacked for both the prior and posterior drift. The diffusion functions are parameterized by a non-linear diagonal function, where each output coordinate is a function of only the corresponding input coordinate. Each coordinates diffusion MLP has single dimension input and output with four hidden layers of 32 neurons each. In the diffusion functions tanh activations were used for added stability and the final activation was exponential function so that the output is positive. The total number of parameters in the drift and diffusion functions is 1,862,977.

Numerical Solver The SDE is numerically solved using the reversible Heun scheme [46], which converges at strong order 0.5 to the Stratonovich SDE solution. A uniform timestep of 0.0625 is used.

Backpropagation The SDE solver is differentiated through in the optimization process. For the backward pass, an adjoint SDE is solved, following the derivation in [52]. Even though the Itô integral is equivalent to the Stratonovich integral, using the Stratonovich SDE makes the form of the adjoint SDE convenient to derive and constructs a more efficient backward process [46].

E Hyperparameters

The following hyperparameters were tuned over the validation set: learning rate, KL penalty and the number of layers in each transformer block. We selected the best performing model based on mean squared error on the validation trajectories, averaged over 8 rollout steps.

Learning rate schedule We trained the model for 25 epochs. The learning rate schedule followed a linear warmup phase from 0 to the base learning rate α over the first epoch. Over the remaining 24 epochs, the learning rate decays to 0 according to cosine decay schedule. The base learning rate α is tuned separately for both the niLES and the deterministic NN model from among a logarithmic distribution of learning rates.

KL penalty The KL penalty term follows a linear warmup to the final value β over the first ten epochs. Beyond that, the KL penalty remains fixed at the same value for the rest of the training period. The KL penalty term β was selected from among the values 0.001, 0.01, 0.1, $1.$ and 10.

Number of layers The number of layers in each phase of the MViT (see Appendix G) is tuned. The validation error is found to saturate at 6 layers for each phase, while the inference time degrades with increase in the number of layers.

Both deterministic NN and the niLES model were hyperparameter-tuned in the same way, except when the hyperparameter (such as KL penalty) were not applicable to the deterministic NN model.

F Computational resources

We used 8 Nvidia V100s to train both the deterministic NN baseline and the niLES model. Training was done for 25 epochs of the training data which took up to 20 hours. For the inference phase we used Google Colab runtime with a single Nvidia V100. For the spectral element numerical solver, we used double (float64) precision. The model parameters and intermediate model variables during training and inference used single (float32) precision.

G Encoder-decoder architecture

The overall architecture of both the niLES model and the deterministic NN model consists of a multiscale ViT (MViT) structure [27, 53]. The architecture of the deterministic NN consists of an encoder and decoder, where the encoder follows the MViT architecture. The encoder maps the input to a much lower latent dimension, while decoder mirrors the same architecture in the reverse sequence. The total number of parameters in the Encoder-Decoder is 2,752,178.

Encoder The encoder $\mathcal{E}$ takes as input the LES field v defined on the coarse-grained discretization $\overline{\mathcal{G}}$. We tackle 2D geometries using a mesh of E elements, each possessing $(P+1)^2$ nodal points at which the velocity is defined where P is the polynomial order. The field v may be interpreted as seqeuence of $E = 144$ tokens, with an input embedding dimension of $d(P+1)^2$. In the first layer, the input is projected to an embedding dimension $W = 48$.

Taking cues from MViT [27, 53]; in a sequence of stages we decrease the number of tokens from E to $E' = 9$ ($E' \ll E$), and each stage has $M = 6$ layers of Transformer blocks. In each stage, the token length is reduced by 4x and the embedding dimension is doubled. The reduction of the number of tokens by attained by mean-pooling the embeddings of the constituent tokens. The increase in the embedding dimension is facilitated by the last MLP in the transformer block. We have a total of two stages, so the final token length is therefore $144 \div (4 \times 4) = 16$ and the final embedding dimension is $48 \times 2 \times 2 = 192$.

Decoder The decoder $\mathcal{D}$ is similar to the encoder, except it operates in reverse order: i.e., it increases the sequence length from E' back to E. The same number of stages are employed, with M layers of transformer blocks in each stage. It has the same number of parameters as $\mathcal{E}$. The decoder, mirroring the encoder, increases the token length by a factor of four at the end of each stage, and decreases the embedding dimension by two.

Skip Connections At the end of every stage of the MViT, a skip connection is added between corresponding layers from the encoder or decoder. For instance, there is a skip connection from the end of the last encoder stage to the beginning of the first decoder stage.

References

[1] Ronald J Adrian. On the role of conditional averages in turbulence theory. *Turbulence in Liquids*, pages 323–332, 01 1977.

[2] Ronald J Adrian. *Stochastic estimation of the structure of turbulent fields*. Springer, 1996.

[3] Ronald J Adrian, BG Jones, MK Chung, Yassin Hassan, CK Nithianandan, and AT-C Tung. Approximation of turbulent conditional averages by stochastic estimation. *Physics of Fluids A: Fluid Dynamics*, 1(6):992–998, 1989.

[4] Nadine Aubry, Philip Holmes, John L Lumley, and Emily Stone. The dynamics of coherent structures in the wall region of a turbulent boundary layer. *Journal of fluid Mechanics*, 192:115–173, 1988.

[5] Gary S Ayton, Will G Noid, and Gregory A Voth. Multiscale modeling of biomolecular systems: in serial and in parallel. *Current opinion in structural biology*, 17(2):192–198, 2007.

[6] Yohai Bar-Sinai, Stephan Hoyer, Jason Hickey, and Michael P Brenner. Learning data-driven discretizations for partial differential equations. *Proceedings of the National Academy of Sciences*, 116(31):15344–15349, 2019.

[7] Peter Bauer, Alan Thorpe, and Gilbert Brunet. The quiet revolution of numerical weather prediction. *Nature*, 525(7567):47–55, 2015.

[8] Andrea Beck, David Flad, and Claus-Dieter Munz. Deep neural networks for data-driven les closure models. *Journal of Computational Physics*, 398:108910, 2019.

[9] Andrea Beck and Marius Kurz. A perspective on machine learning methods in turbulence modeling. *GAMM-Mitteilungen*, 44(1):e202100002, 2021.

[10] G Berkooz. An observation on probability density equations, or, when do simulations reproduce statistics? *Nonlinearity*, 7(2):313, 1994.

[11] Kaifeng Bi, Lingxi Xie, Hengheng Zhang, Xin Chen, Xiaotao Gu, and Qi Tian. Pangu-weather: A 3d high-resolution model for fast and accurate global weather forecast. *arXiv preprint arXiv:2211.02556*, 2022.

[12] Hugh M Blackburn and S Schmidt. Spectral element filtering techniques for large eddy simulation with dynamic estimation. *Journal of Computational Physics*, 186(2):610–629, 2003.

[13] Guido Boffetta and Robert E Ecke. Two-dimensional turbulence. *Annual review of fluid mechanics*, 44:427–451, 2012.

[14] Christoph Bosshard, Michel O Deville, Abdelouahab Dehbi, Emmanuel Leriche, et al. Udns or les, that is the question. *Open Journal of Fluid Dynamics*, 5(04):339, 2015.

[15] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, et al. Jax: composable transformations of python+ numpy programs. 2018.

[16] P Bradshaw. Turbulence modeling with application to turbomachinery. *Progress in Aerospace Sciences*, 32(6):575–624, 1996.

[17] Johannes Brandstetter, Daniel Worrall, and Max Welling. Message passing neural pde solvers. *arXiv preprint arXiv:2202.03376*, 2022.

[18] Ashesh Chattopadhyay, Jaideep Pathak, Ebrahim Nabizadeh, Wahid Bhimji, and Pedram Hassanzadeh. Long-term stability and generalization of observationally-constrained stochastic data-driven models for geophysical turbulence. *Environmental Data Science*, 2:e1, 2023.

[19] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[20] Carlo Cintolesi and Etienne Mémin. Stochastic modelling of turbulent flows for numerical simulations. *Fluids*, 5(3):108, 2020.

[21] R. Courant, K. Friedrichs, and H. Lewy. On the partial difference equations of mathematical physics. *IBM Journal of Research and Development*, 11(2):215–234, 1967.

[22] Daan Crommelin and Eric Vanden-Eijnden. Subgrid-scale parameterization with conditional markov chains. *Journal of the Atmospheric Sciences*, 65(8):2661–2675, 2008.

[23] M. O. Deville, P. F. Fischer, and E. H. Mund. *High-Order Methods for Incompressible Fluid Flow*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2002.

[24] Gideon Dresdner, Dmitrii Kochkov, Peter Norgaard, Leonardo Zepeda-Núñez, Jamie A Smith, Michael P Brenner, and Stephan Hoyer. Learning to correct spectral methods for simulating turbulent flows. *arXiv preprint arXiv:2207.00556*, 2022.

[25] Karthik Duraisamy, Gianluca Iaccarino, and Heng Xiao. Turbulence modeling in the age of data. *Annual review of fluid mechanics*, 51:357–377, 2019.

[26] Paul A Durbin. Some recent developments in turbulence closure modeling. *Annu. Rev. Fluid Mech.*, 50, 2018.

[27] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6824–6835, 2021.

[28] Paul Fischer and Julia Mullen. Filter-based stabilization of spectral element methods. *Comptes Rendus de l'Académie des Sciences-Series I-Mathematics*, 332(3):265–270, 2001.

[29] Kai Fukami, Koji Fukagata, and Kunihiko Taira. Super-resolution reconstruction of turbulent flows with machine learning. *Journal of Fluid Mechanics*, 870:106–120, 2019.

[30] Kai Fukami, Koji Fukagata, and Kunihiko Taira. Machine-learning-based spatio-temporal super resolution reconstruction of turbulent flows. *Journal of Fluid Mechanics*, 909:A9, 2021.

[31] Massimo Germano, Ugo Piomelli, Parviz Moin, and William H Cabot. A dynamic subgrid-scale eddy viscosity model. *Physics of Fluids A: Fluid Dynamics*, 3(7):1760–1765, 1991.

[32] Ian Grooms and Andrew J Majda. Efficient stochastic superparameterization for geophysical turbulence. *Proceedings of the National Academy of Sciences*, 110(12):4464–4469, 2013.

[33] Ian Grooms and Andrew J Majda. Stochastic superparameterization in quasigeostrophic turbulence. *Journal of Computational Physics*, 271:78–98, 2014.

[34] Yifei Guan, Ashesh Chattopadhyay, Adam Subel, and Pedram Hassanzadeh. Stable a posteriori les of 2d turbulence using convolutional neural networks: Backscattering analysis and generalization to higher re via transfer learning. *Journal of Computational Physics*, 458:111090, 2022.

[35] Yifei Guan, Adam Subel, Ashesh Chattopadhyay, and Pedram Hassanzadeh. Learning physics-constrained subgrid-scale closures in the small-data regime for stable and accurate les. *Physica D: Nonlinear Phenomena*, 443:133568, 2023.

[36] 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.

[37] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International conference on learning representations*, 2017.

[38] Thomas JR Hughes, Gonzalo R Feijóo, Luca Mazzei, and Jean-Baptiste Quincy. The variational multiscale method—a paradigm for computational mechanics. *Computer methods in applied mechanics and engineering*, 166(1-2):3–24, 1998.

[39] Thomas JR Hughes, Luca Mazzei, and Kenneth E Jansen. Large eddy simulation and the variational multiscale method. *Computing and visualization in science*, 3:47–59, 2000.

[40] Thomas JR Hughes, Assad A Oberai, and Luca Mazzei. Large eddy simulation of turbulent channel flows by the variational multiscale method. *Physics of fluids*, 13(6):1784–1799, 2001.

[41] Javier Jimenez and Robert D Moser. Large-eddy simulations: where are we and what can we expect? *AIAA journal*, 38(4):605–612, 2000.

[42] Myeongseok Kang, Youngmin Jeon, and Donghyun You. Neural-network-based mixed subgrid-scale model for turbulent flow. *Journal of Fluid Mechanics*, 962:A38, 2023.

[43] Petr Karnakov, Sergey Litvinov, and Petros Koumoutsakos. Computing foaming flows across scales: From breaking waves to microfluidics. *Science Advances*, 8(5):eabm0590, 2022.

[44] Patrick Kidger. On neural differential equations. *arXiv preprint arXiv:2202.02435*, 2022.

[45] Patrick Kidger, James Foster, Xuechen Li, and Terry J Lyons. Neural sdes as infinite-dimensional gans. In *International Conference on Machine Learning*, pages 5453–5463. PMLR, 2021.

[46] Patrick Kidger, James Foster, Xuechen Chen Li, and Terry Lyons. Efficient and accurate gradients for neural sdes. *Advances in Neural Information Processing Systems*, 34:18747–18761, 2021.

[47] Dmitrii Kochkov, Jamie A Smith, Ayya Alieva, Qing Wang, Michael P Brenner, and Stephan Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21):e2101784118, 2021.

[48] Andrei Nikolaevich Kolmogorov. The local structure of turbulence in incompressible viscous fluid for very large reynolds numbers. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 434(1890):9–13, 1991.

[49] Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Alexander Pritzel, Suman Ravuri, Timo Ewalds, Ferran Alet, Zach Eaton-Rosen, et al. Graphcast: Learning skillful medium-range global weather forecasting. *arXiv preprint arXiv:2212.12794*, 2022.

[50] Jacob A Langford and Robert D Moser. Optimal les formulations for isotropic turbulence. *Journal of fluid mechanics*, 398:321–346, 1999.

[51] CE Leith. Stochastic backscatter in a subgrid-scale model: Plane shear mixing layer. *Physics of Fluids A: Fluid Dynamics*, 2(3):297–299, 1990.

[52] Xuechen Li, Ting-Kam Leonard Wong, Ricky TQ Chen, and David K Duvenaud. Scalable gradients and variational inference for stochastic differential equations. In *Symposium on Advances in Approximate Bayesian Inference*, pages 1–28. PMLR, 2020.

[53] Yanghao Li, Chao-Yuan Wu, Haoqi Fan, Karttikeya Mangalam, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. Mvitv2: Improved multiscale vision transformers for classification and detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4804–4814, 2022.

[54] Yi Li, Eric Perlman, Minping Wan, Yunke Yang, Charles Meneveau, Randal Burns, Shiyi Chen, Alexander Szalay, and Gregory Eyink. A public turbulence database cluster and applications to study lagrangian evolution of velocity increments in turbulence. *Journal of Turbulence*, 9(9):N31, 2008.

[55] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv:2010.08895 [cs, math]*, May 2021. arXiv: 2010.08895.

[56] Björn List, Li-Wei Chen, and Nils Thuerey. Learned turbulence modelling with differentiable fluid solvers: physics-based loss functions and optimisation horizons. *Journal of Fluid Mechanics*, 949:A25, 2022.

[57] Peter Lynch. *The emergence of numerical weather prediction: Richardson's dream*. Cambridge University Press, 2006.

[58] Jonathan F MacArt, Justin Sirignano, and Jonathan B Freund. Embedded training of neural-network subgrid-scale turbulence models. *Physical Review Fluids*, 6(5):050502, 2021.

[59] Romit Maulik, Kai Fukami, Nesar Ramachandra, Koji Fukagata, and Kunihiko Taira. Probabilistic neural networks for fluid flow surrogate modeling and data recovery. *Physical Review Fluids*, 5(10):104401, 2020.

[60] Romit Maulik, Omer San, Jamey D Jacob, and Christopher Crick. Sub-grid scale model classification and blending through deep learning. *Journal of Fluid Mechanics*, 870:784–812, 2019.

[61] Siddhartha Mishra. A machine learning framework for data driven acceleration of computations of differential equations. *arXiv preprint arXiv:1807.09519*, 2018.

[62] Steven A. Orszag. Analytical theories of turbulence. *Journal of Fluid Mechanics*, 41(2):363–386, 1970.

[63] Shaowu Pan and Karthik Duraisamy. Data-driven discovery of closure models. *SIAM Journal on Applied Dynamical Systems*, 17(4):2381–2413, 2018.

[64] Jonghwan Park and Haecheon Choi. Toward neural-network-based large eddy simulation: Application to turbulent channel flow. *Journal of Fluid Mechanics*, 914:A16, 2021.

[65] Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, et al. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022.

[66] Stephen B Pope. Pdf methods for turbulent reactive flows. *Progress in energy and combustion science*, 11(2):119–192, 1985.

[67] Stephen B Pope. Ten questions concerning the large-eddy simulation of turbulent flows. *New Journal of Physics*, 6:35–35, March 2004.

[68] Stephen B Pope and Stephen B Pope. *Turbulent flows*. Cambridge university press, 2000.

[69] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *International Conference on Machine Learning*, pages 5301–5310. PMLR, 2019.

[70] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[71] 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.

[72] Pierre Sagaut. *Large eddy simulation for incompressible flows: an introduction*. Springer Science & Business Media, 2005.

[73] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International conference on machine learning*, pages 8459–8468. PMLR, 2020.

[74] Tapio Schneider, João Teixeira, Christopher S Bretherton, Florent Brient, Kyle G Pressel, Christoph Schär, and A Pier Siebesma. Climate goals and computing the future of clouds. *Nature Climate Change*, 7(1):3–5, 2017.

[75] Claude Elwood Shannon. Communication in the presence of noise. *Proceedings of the Institute of Radio Engineers*, 37(1):10–21, 1949.

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

[77] Kimberly Stachenfeld, Drummond B Fielding, Dmitrii Kochkov, Miles Cranmer, Tobias Pfaff, Jonathan Godwin, Can Cui, Shirley Ho, Peter Battaglia, and Alvaro Sanchez-Gonzalez. Learned coarse models for efficient turbulence simulation. *arXiv preprint arXiv:2112.15275*, 2021.

[78] Adam Subel, Ashesh Chattopadhyay, Yifei Guan, and Pedram Hassanzadeh. 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, 2021.

[79] Salar Taghizadeh, Freddie D Witherden, and Sharath S Girimaji. Turbulence closure modeling with data-driven techniques: physical compatibility and consistency considerations. *New Journal of Physics*, 22(9):093023, 2020.

[80] Belinda Tzen and Maxim Raginsky. Neural stochastic differential equations: Deep latent gaussian models in the diffusion limit. *arXiv preprint arXiv:1905.09883*, 2019.

[81] Kiwon Um, Robert Brand, Yun Raymond Fei, Philipp Holl, and Nils Thuerey. Solver-in-the-loop: Learning from differentiable physics to interact with iterative pde-solvers. *Advances in Neural Information Processing Systems*, 33:6111–6122, 2020.

[82] Rui Wang, Karthik Kashinath, Mustafa Mustafa, Adrian Albert, and Rose Yu. Towards physics-informed deep learning for turbulent flow prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1457–1466, 2020.

[83] Winnie Xu, Ricky TQ Chen, Xuechen Li, and David Duvenaud. Infinitely deep bayesian neural networks with stochastic differential equations. In *International Conference on Artificial Intelligence and Statistics*, pages 721–738. PMLR, 2022.