

Differentiable Turbulence: Closure as a PDE-constrained optimization

Varun Shankar,¹ Dibyajyoti Chakraborty,² Venkatasubramanian Viswanathan,³ and Romit Maulik^{2,4}

¹*Carnegie Mellon University*

²*Pennsylvania State University*

³*University of Michigan*

⁴*Argonne National Laboratory*

(*Electronic mail: rmaulik@psu.edu)

(Dated: 29 March 2024)

Deep learning is increasingly becoming a promising pathway to improving the accuracy of sub-grid scale (SGS) turbulence closure models for large eddy simulations (LES). We leverage the concept of differentiable turbulence, whereby an end-to-end differentiable solver is used in combination with physics-inspired choices of deep learning architectures to learn highly effective and versatile SGS models for two-dimensional turbulent flow. We perform an in-depth analysis of the inductive biases in the chosen architectures, finding that the inclusion of small-scale non-local features is most critical to effective SGS modeling, while large-scale features can improve pointwise accuracy of the *a-posteriori* solution field. The velocity gradient tensor on the LES grid can be mapped directly to the SGS stress via decomposition of the inputs and outputs into isotropic, deviatoric, and anti-symmetric components. We see that the model can generalize to a variety of flow configurations, including higher and lower Reynolds numbers and different forcing conditions. We show that the differentiable physics paradigm is more successful than offline, *a-priori* learning, and that hybrid solver-in-the-loop approaches to deep learning offer an ideal balance between computational efficiency, accuracy, and generalization. Our experiments provide physics-based recommendations for deep-learning based SGS modeling for generalizable closure modeling of turbulence.

I. INTRODUCTION

Simulations of turbulent flow constitute an integral part of modeling and analysis for many scientific and engineering problems driven by fluid flow. Many real-world flows, such as climate dynamics, jets, blood flow, or external aerodynamics, are turbulent in nature, characterized by chaotic and multi-scale behavior^{1,2}. Numerical solutions to turbulent flow are often extraordinarily challenging to obtain, due to the vast number of temporal and spatial scales that must be resolved. In practice, direct numerical simulation (DNS) of the governing Navier-Stokes equations is infeasible and hence DNS is limited to canonical flow configurations³. Major efforts in computational fluid dynamics (CFD) methods have therefore focused on the development of turbulence models, which approximate the effects of turbulence and ultimately reduce the computational burden of simulations by averaging unresolved flow features⁴. For example, Reynolds-Averaged Navier-Stokes (RANS) methods develop steady-state solutions to the governing equations by modeling the time-averaged turbulent fluctuations in the flow encompassed by the Reynolds stresses^{2,5}. The smooth solutions provided by the RANS equations enable a significant reduction in grid-resolution requirements, and RANS has remained the workhorse of CFD methods for many engineering tasks. The increase in computational capabilities over the last few decades has placed new emphasis on Large Eddy Simulation (LES) methods^{4,6-8}. With LES, the large scales of the flow are directly resolved on the computational grid, and the effects of the subgrid-scale (SGS) flow are accounted for using an SGS model. The advantage of LES is that

the temporal evolution of the flow field can be modeled, which is needed for certain applications such as weather forecasting⁹. On the other hand, the discretization requirements for LES are generally more strict than RANS, leading to larger computational costs that limit applicability.

The averaged or ‘filtered’ equations used in turbulence modeling are near identical to the governing Navier-Stokes equations, with the exception of an effective source term that embodies the influence of the unresolved flow. The accuracy of the turbulence model therefore has a significant impact on the quality of the resulting solution field^{10,11}. Traditional turbulence models have been developed from theory and empirical testing. While these methods have provided a strong foundation for further research and improvement, novel approaches will be required to address the inherent limitations of current models^{12,13}. Given the partially data-driven nature of modeling approaches, it is a natural extension to consider machine learning (ML) paradigms for development such as deep learning. Data-driven turbulence modeling through deep learning has seen explosive interest in the last few years, particularly for RANS models^{14,15}. Ling, et al.’s seminal work outlines an approach to model the Reynolds stress anisotropy tensor with an artificial neural network (ANN)¹⁶. Other avenues have been explored as well, including iterative methods¹⁷, field inversion¹⁸, and wall-modeling¹⁹. Simultaneously, recent efforts have been devoted to learning more accurate SGS models for LES. Maulik et al. explored learning the SGS stress using an artificial neural network (ANN) with promising results²⁰. Wang et al. examined the necessary input features for an SGS

model using random forests and ANNs²¹. Frezat et al. modeled the SGS scalar flux and constrained their model with invariances and symmetries²². Furthermore, Guan et al. used convolutional networks to learn SGS closures and investigated the effectiveness of transfer learning for generalization^{23–25}.

Before proceeding further, we elaborate on the notion of the ‘filter’ in LES. In several data-driven closure modeling studies for LES, and in particular for most *a-priori*^{20,23} variants, optimal subgrid stress quantities are computed for the purpose of ascertaining ground truth. These are obtained through the computation of filtered flow-fields assuming a low-pass spatial filter such as a Gaussian filter or a sharp spectral cut-off filter. It has also been observed that the optimal subgrid stress is a function of this filter²⁶. Several studies have demonstrated that such assumptions are unreasonably strong and result in ad-hoc clipping requirements or strong inductive biases for *a-priori* trained data-driven closures during *a-posteriori* deployments^{20,27–29}. We note that while *a-priori* insights are critical to deeper understanding of SGS effects, the desired goal is to accurately reproduce the *a-posteriori* flow field. Notably, *a-posteriori* evaluation introduces several additional consequences that cannot be modeled with *a-priori* strategies alone, including numerical and discretization errors and temporal effects.

One approach to overcome with *a-posteriori* data-driven turbulence modeling is the development of differentiable CFD solvers, which are required to backpropagate an *a-posteriori* error to the parameters of the turbulence model. We term the application of differentiable programming techniques to enhance turbulence models “differentiable turbulence”. While alternative machine learning techniques such as reinforcement learning have been explored to circumvent the need for differentiable simulations^{30–32}, the majority of literature on ML for spatiotemporal turbulence forecasting has leveraged pure ML architectures, where the flow solution is obtained not through traditional finite-volume or finite-element numerical methods, but rather a deep learning model alone^{33–35}. For example, LSTMs³⁶, GANs³⁷, and other physics-informed approaches³⁸ have been used to model turbulent flow. The limitation of these approaches is that without explicit knowledge of the underlying physics, generalization to unseen flow configurations can be especially challenging. Despite the challenges, the adoption of differentiable programming paradigms has led to several publications of differentiable CFD solvers in the last few years^{39–41}. These differentiable solvers are still very much in the nascent stages of development, given the difficulties associated with writing a CFD solver from the ground up, and for now, cannot compete with the vast ecosystem of robust non-differentiable numerical solvers in terms of applicability. However, they offer a pathway to exploring differentiable turbulence techniques for designing hybrid physics and ML solvers that combine the learning power of deep learning methods with the generalization abil-

ity of well-tested numerical methods. However, there are also some recent examples of *a-posteriori* learning methods that utilize emulators of non-differentiable solvers⁴².

The concept of “reverse-mode differentiation” in CFD simulations has existed for decades and is analogous to solving the adjoint problem^{43–45}. Adjoint solutions have historically been used for applications such as shape and control optimization^{46–49}, but have garnered renewed interest with the growth of machine learning. Differentiable simulations have been integrated with ML as “solver-in-the-loop” approaches particularly for correcting and improving the accuracy of coarse-grained or unresolved simulations, which offers a balance between computational cost and physics-embedding. Previous works have targeted learning correction operators to the solution field⁵⁰, or learning interpolation functions within the numerical scheme³⁹, which have shown promising results relative to both baseline traditional numerical solution methods and pure ML models. Given that this work is specifically focused on LES, we choose to learn an existing term that appears within the governing PDE, the SGS stress, which is an interpretable and well-studied quantity that is amenable to analysis with conventional techniques. Some works have leveraged differentiable simulations for learning turbulence models, notably List et al.⁵¹, who extend the PhiFlow solver for SGS modeling, Sirignano et al.⁵², who use the stochastic adjoint method for LES modeling, and Frezat et al.⁵³, who examine backscatter in quasi-geostrophic turbulence. Our aim here is to demonstrate the effectiveness of “solver-in-the-loop” approaches to learning turbulence models, to offer an in depth examination of the inductive biases contained within the models, and to provide insight for future development of data-driven SGS models.

In this work, we learn SGS closure models using a deep learning-embedded differentiable CFD solver. We consider two-dimensional (2D) homogeneous isotropic turbulence as a candidate test problem to evaluate our approach and demonstrate the capability of the learned models to produce accurate *a-posteriori* solution trajectories. Our novel contributions are as follows: (1) we design and test several methods to model the SGS stress tensor, inspired by existing eddy viscosity models and tensor theory, within the differentiable physics paradigm, (2) we perform a deep analysis of network architectures to understand the importance of non-local SGS models and which length scales are necessary to capture for accurate modeling, and (3) we compare our approach with offline learning of the SGS stress to validate the need for end-to-end optimization enabled by the differentiable solver.

II. METHODS

A. Governing Equations

Turbulence models are developed for the two-dimensional incompressible Navier-Stokes equations in a doubly periodic domain, given in its non-dimensional form as:

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u} \otimes \mathbf{u}) = \frac{1}{Re} \nabla^2 \mathbf{u} - \frac{1}{\rho} \nabla p + \mathbf{f} \quad (1)$$

$$\nabla \cdot \mathbf{u} = 0, \quad (2)$$

where \mathbf{u} is the velocity vector, defined $\{u, v\}$, p is the pressure, ρ is the density, Re is the Reynolds number, and \mathbf{f} represents any external forces. The dimensionless parameter Re represents the ratio of convective to diffusive forces in the flow and characterizes many critical aspects of the flow behavior. At low Reynolds numbers, viscous forces dampen instabilities in the flow leading to smooth solutions with low frequency components. At high Reynolds numbers, inertial forces can amplify minor perturbations in the flow to produce energy-containing fluctuations or eddies in the solution field that span many orders of magnitude in space and time¹. If the high frequency components of the solution field exceed the resolution of the numerical grid, a direct numerical simulation of the Navier-Stokes equations can become under-resolved, resulting in discretization errors and inaccurate solutions. High Reynolds number flows can therefore be cost-prohibitive or infeasible to evaluate with DNS, given the grid spacing required to fully resolve the flow field.

In many scientific and engineering applications, it is desirable to simulate high Reynolds number flows at computationally tractable resolutions much lower than what is imposed by DNS. In such a scenario, the governing equations must be augmented to account for the unresolved length-scales in the solution field. LES resolves the flow field up to some cut-off length scale Δ . Conceptually, it is assumed that the high frequency contributions are removed via a filtering operation, i.e., a filter G_Δ with characteristic length scale Δ is convolved with the velocity field to arrive at the target solution field:

$$\bar{\mathbf{u}}(x, t) = G_\Delta \star \mathbf{u}(x, t). \quad (3)$$

Consequently, to solve for the filtered velocity field, the governing equations can be represented in terms of filtered variables with the introduction of a source term in the momentum equation to account for unresolved interactions between resolved and unresolved scales in the flow-field. Therefore, we have,

$$\frac{\partial \bar{\mathbf{u}}}{\partial t} + \nabla \cdot (\bar{\mathbf{u}} \otimes \bar{\mathbf{u}}) = \frac{1}{Re} \nabla^2 \bar{\mathbf{u}} - \frac{1}{\rho} \nabla \bar{p} + \bar{\mathbf{f}} + \nabla \cdot \boldsymbol{\tau}, \quad (4)$$

where $\boldsymbol{\tau}$ represents the effects of the unresolved velocity components on the resolved field and is defined as

$$\tau_{ij} = \overline{u_i u_j} - \bar{u}_i \bar{u}_j. \quad (5)$$

This quantity is termed the subgrid-scale (SGS) stress and has a nontrivial impact on the filtered velocity field. Ignoring the term can lead to numerical errors or spurious solution fields, however, it cannot be directly computed without access to the unresolved velocity components. Instead, it is the role of the SGS turbulence model to approximate $\boldsymbol{\tau}$ from the resolved field such that the problem can be closed. We emphasize, again, that the notion of ‘filtering’ a DNS field is a conceptual tool to describe the coarse-grained evolution of an LES. The nature of this filter is generally unknown, but can be crudely approximated through *a-priori* knowledge of the numerical scheme (finite volume, spectral, etc.). However, for almost all practical problems, the nature of this filtering operation is unknown which complicates data-driven modeling for τ_{ij} .

Solutions to the Navier-Stokes and filtered LES equations are computed using finite-volume code written in domain-specific language designed for differentiable programming, JAX-CFD³⁹. JAX-CFD takes advantage of native JAX autodifferentiation⁵⁴ to allow users to compute gradients of any parameter in the solver using reverse-mode differentiation. The implementation uses the discrete adjoint method⁵⁵ for efficiently propagating gradients through linear solves in the algorithm. The numerical scheme uses a staggered grid for the velocity and pressure fields, second-order central difference schemes for fluxes, and explicit Euler time integration.

True solutions are obtained from a high-resolution simulation of the governing equations. Target fields are computed by filtering and coarse-graining using G_Δ , a Gaussian filter with width twice the grid spacing and a spectral cutoff at the Nyquist frequency of the LES grid. This choice of filter is effective for data-driven SGS modeling⁵⁶ and additionally selected based on the use of second-order numerical schemes. More details on the specifics of the datasets are provided in Sec. III.

B. Subgrid-scale (SGS) modeling

The goal of the SGS model is to estimate the SGS stress $\boldsymbol{\tau}$, whose divergence functionally appears as an additional source term on the right-hand side of the momentum equation. Over the last few decades, several approaches have been proposed to model the SGS stress. Given that the primary function of the SGS model is to remove energy from the flow to account for the transfer of energy from the resolved scales to the unresolved scales, the most widely used SGS models are dissipative linear eddy viscosity models, commonly used in RANS turbulence models as well. The eddy viscosity model postulates that the SGS stress is proportional to the rate-of-strain through an effective eddy viscosity:

$$\tau_{ij} = -2\nu_t \bar{S}_{ij}, \quad (6)$$

where ν_t is the eddy viscosity and $\bar{S}_{ij} = \frac{1}{2}(\partial_j \bar{u}_i + \partial_i \bar{u}_j)$ is the rate-of-strain tensor. The most well-known SGS eddy

viscosity model is the Smagorinsky model⁵⁷, where the eddy viscosity is determined through the characteristic length scale Δ and a characteristic velocity $\Delta|\bar{\mathbf{S}}|$, where $|\bar{\mathbf{S}}| = (2\bar{S}_{ij}\bar{S}_{ji})^{1/2}$ to give

$$\begin{aligned}\nu_t &= (C_s\Delta)^2|\bar{\mathbf{S}}| \\ \boldsymbol{\tau}_{smag} &= -2\nu_t\bar{\mathbf{S}},\end{aligned}\quad (7)$$

where C_s is a dimensionless empirical coefficient. The Smagorinsky model has been effectively used in a variety of application areas, however, its simplistic nature is directly tied to assumptions that can lead to deficiencies for certain flows, namely the use of an isotropic, positive eddy viscosity. The restriction of ν_t to positive values means the model is purely dissipative and cannot model the transfer of energy from subgrid to resolved scales, also known as backscatter.

The approach taken in this study is to learn an additive correction to the Smagorinsky model using a data-driven modeling paradigm to account for SGS effects that cannot be captured by the Smagorinsky model alone. The SGS stress is therefore given by

$$\hat{\boldsymbol{\tau}} = \boldsymbol{\tau}_{smag} + \boldsymbol{\tau}_{ml}, \quad (8)$$

where $\boldsymbol{\tau}_{ml}$ represents a tensor-valued correction to SGS stress that is computed from a deep neural network. Such an approach has strong foundations in existing SGS modeling strategies and is considered a mixed model⁵⁸, where an eddy viscosity term is added to the stress from an alternative modeling approach such as dynamic⁵⁹ or deconvolution⁶⁰ methods.

Given this formulation, the question of how to compute $\boldsymbol{\tau}_{ml}$ using a neural network remains nontrivial. We first assume that $\boldsymbol{\tau}_{ml}$ is a function of the instantaneous filtered velocity field $\bar{\mathbf{u}}$. Then, in general, the SGS model can be represented as,

$$\boldsymbol{\tau}_{ml} = \mathcal{M}(\bar{\mathbf{u}}, f_\theta, \phi_{in}, \phi_{out}), \quad (9)$$

where f_θ is a neural network with parameters θ and ϕ_{in} and ϕ_{out} represent transformations of the input velocity field and neural network outputs respectively. Each of the three functions can be varied to optimize model accuracy and learning. In this section, we discuss ϕ_{in} and ϕ_{out} . Variation of f_θ will be considered in the following section. We show a general schematic of the algorithm in Fig. 1.

A naive implementation of the model would be to take ϕ_{in} and ϕ_{out} as identity maps, but this approach has several drawbacks. ϕ_{in} as the identity function has been commonly used in previous studies of data-driven turbulence modeling, particularly for LES⁵¹, however, there is an immediately observable flaw in that the model does not respect Galilean invariance. A change in inertial reference frame would result in different network outputs and subsequently $\boldsymbol{\tau}_{ml}$. A straightforward solution is to start by incorporating the gradient operator into ϕ_{in} such that the model is a function of $\nabla\bar{\mathbf{u}}$. While this preserves

invariance with respect to translations of the reference frame, more consideration must be taken for rotation or reflection symmetries. Regarding ϕ_{out} , the identity would mean that the network outputs each of the tensor components of $\boldsymbol{\tau}_{ml}$. In practice, this can easily lead to numerical instabilities in the simulation during both training and evaluation and is thus not an optimal design choice. We examine several alternatives of ϕ_{in} and ϕ_{out} , which are evaluated in this study.

The first set of models is motivated by tensor basis theory. The use of tensor bases has been extensively leveraged in literature for RANS data-driven turbulence modeling^{16,61,62}. If we assume that the model is an arbitrary function of input tensors, one can construct a finite set of basis tensors whose linear combination via certain weight coefficients is equal to any tensor function of the input, given by:

$$\boldsymbol{\tau}_{ml} = \sum_n \alpha^{(n)}\mathbf{T}^{(n)}, \quad (10)$$

where $\mathbf{T}^{(n)}$ are the basis tensors and $\alpha^{(n)}$ are the coefficients. A common choice of input tensors in turbulence modeling is the strain and rotation rate tensors \mathbf{S} and \mathbf{R} , which are the symmetric and anti-symmetric components of the velocity gradient tensor respectively. Pope derived the integrity basis for the \mathbf{S} and \mathbf{R} tensors in two and three dimensions⁶³. The bases in 2D are:

$$\mathbf{T}^{(0)} = \mathbf{I}, \quad \mathbf{T}^{(1)} = \mathbf{S}, \quad \mathbf{T}^{(2)} = \mathbf{SR} - \mathbf{RS}. \quad (11)$$

The task of the neural network, then, is to learn the coefficients of the tensor basis functions. It is advantageous to have the coefficients be functions of the invariants of the input tensors, such that the model is ultimately Galilean invariant to rotations and reflections as well as translations. The number of invariants is also finite, $\{\mathbf{S}^2\}$ and $\{\mathbf{R}^2\}$ in 2D, where $\{\mathbf{T}^2\} = T_{ij}T_{ji}$. Ling, et al. used this approach¹⁶, also considered a nonlinear eddy viscosity model⁵, to model the Reynolds anisotropy tensor using a neural network.

For the model outlined above,

$$\phi_{in}(\bar{\mathbf{u}}) = \{\bar{\mathbf{S}}^2\}, \{\bar{\mathbf{R}}^2\} \quad (12)$$

$$f_\theta(\{\bar{\mathbf{S}}^2\}, \{\bar{\mathbf{R}}^2\}) = \alpha \quad (13)$$

$$\phi_{out}(\alpha) = \sum_{n=0}^2 \alpha^{(n)}\mathbf{T}^{(n)}, \quad (14)$$

such that $\boldsymbol{\tau}_{ml} = (\phi_{out} \circ f_\theta \circ \phi_{in})(\bar{\mathbf{u}})$. Variations of this model are also considered, in terms of the tensor basis functions and thus ϕ_{out} . The tensor bases above are restricted to symmetric tensors to provide a physically valid stress. Despite this, the basis can be expanded to include asymmetric terms. In 2D, we must only include one additional tensor, $\mathbf{T}^{(3)} = \mathbf{R}$. On the other hand, we can create another model that removes any nonlinear terms in the basis, such that only $\mathbf{T}^{(0)}$ and $\mathbf{T}^{(1)}$ are used.

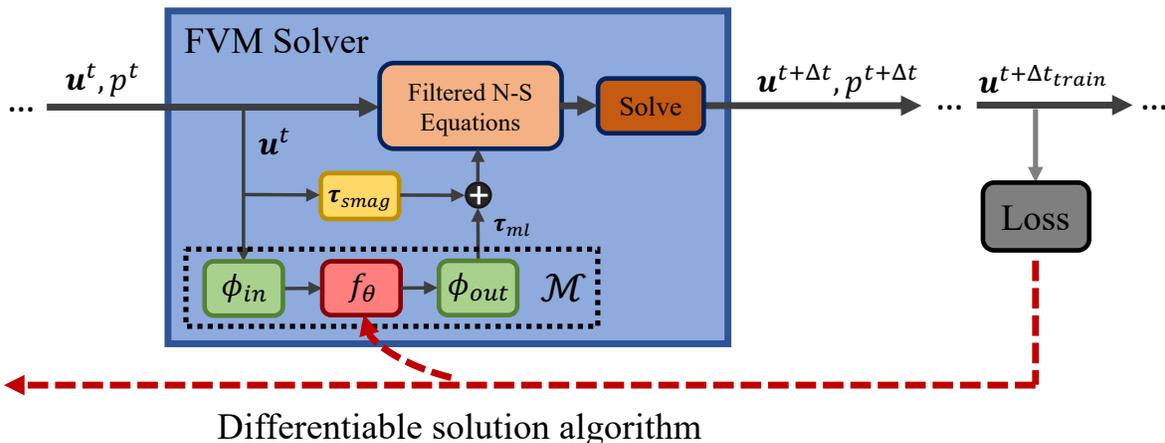


FIG. 1. A schematic of the deep learning-embedded solution algorithm. At each time step, the eddy-viscosity contribution of the subgrid stress is computed from τ_{smag} and the ML contribution from \mathcal{M} , where f_θ represents a neural network with trainable parameters and ϕ_{in}, ϕ_{out} are fixed transformations of the inputs and outputs respectively. The contributions are summed and used in the LES equations, which are solved using standard numerical schemes. The solution trajectory is propagated and the loss is evaluated with respect to the partial (i.e., subsampled) observations of the ground truth DNS field. Because the solution algorithm is differentiable, the loss can be backpropagated through all time steps and linear solves to update the trainable parameters.

In this case, we recover essentially a linear eddy viscosity model, albeit the effective eddy viscosity is computed from a neural network.

An alternative approach for ϕ_{out} can be taken where we disregard the use of tensor bases and instead learn the stress tensor itself, without adapting an existing eddy viscosity model. As mentioned earlier, taking the four components of τ_{ml} directly from the neural network can lead to numerical issues, and we found that it was impractical to try and train a model in this fashion. We found that it was feasible to split the tensor into its isotropic, deviatoric, and optionally asymmetric components and learn these individually. The stress is then given by

$$\tau_{ml} = \alpha \mathbf{I} + \mathbf{D} + \mathbf{A}, \quad (15)$$

where \mathbf{I} is the identity, \mathbf{D} is the deviatoric tensor with two free components, and \mathbf{A} is the anti-symmetric tensor with one free component. The network f_θ outputs a scalar α , the two components comprising \mathbf{D} , and possibly one component comprising \mathbf{A} . We term this the “model-free” (MF) approach.

The secondary aspect to vary is ϕ_{in} . So far we have considered ANN inputs $\{\bar{\mathbf{S}}^2\}, \{\bar{\mathbf{R}}^2\}$. Instead of providing purely the invariants of the tensors $\bar{\mathbf{S}}$ and $\bar{\mathbf{R}}$, we can use the tensors themselves as input, in terms of their two or one free components respectively. While this comes at the expense of rotation invariance, it can provide additional information to the model for more accurate predictions, and the symmetry itself may be learned implicitly by the model.

We refer to various combinations of ϕ_{in} and ϕ_{out} with several names. A table of ϕ_{out} ’s and their names are

TABLE I. Model names

Name	ϕ_{out}
Linear (LIN)	$\sum_{n=0}^1 \alpha^{(n)} \mathbf{T}^{(n)}$
Non-linear (NL)	$\sum_{n=0}^2 \alpha^{(n)} \mathbf{T}^{(n)}$
Non-linear asymmetric (NLA)	$\sum_{n=0}^3 \alpha^{(n)} \mathbf{T}^{(n)}$
Model-free (MF)	$\alpha \mathbf{I} + \mathbf{D} + \mathbf{A}$
Model-free symmetric (MFS)	$\alpha \mathbf{I} + \mathbf{D}$

given in Tab. I. A “-I” is appended to the name if ϕ_{in} provides invariant inputs.

C. Deep learning architectures

The choice of ANN f_θ provides an important inductive bias to the SGS model, particularly with regards to the scales of the flow that are captured by the model. Previous works have focused on using two common architectures – multi-layer perceptrons (MLPs) and convolutional neural networks (CNNs)^{20,21,23,51,64}. In this context, MLPs are purely local in nature, in that the evaluation of the SGS stress is parallelized over grid points to produce a model that is only a function of the local strain and rotation rate tensors. The local formulation is supported by many existing SGS models, including the Smagorinsky model, which is a function of the local strain rate. Despite the success of these local models, it is well known that turbulent dynamics at a point in space are influenced by the surrounding flow as well⁶⁵, implying that features of the flow at length scales larger than the grid size may be used

to determine the SGS stress. For example, the dynamic Smagorinsky model or other scale-similarity approaches compute model coefficients on-the-fly by filtering the LES flow with a test filter that is larger than the grid filter. The result is that the scales of the flow between the test filter and the LES cutoff are used to estimate the SGS stress. Moreover, it is also common to average model coefficients in space (for instance spanwise direction averaging in channel flow).

The adoption of these non-local, dynamic models in practice motivates the use of CNNs in SGS modeling. The convolutional architecture incorporates information from the surrounding strain and rotation rate quantities through convolutional filters of a specified kernel width. By chaining many of these CNN layers together, the resulting deep model employs a large receptive field that captures a great degree of length scales in the flow. Concretely, the MLP and CNN architectures we use have 6 hidden layers with 64 channels each and ReLU activation functions and one linear output layer. The CNN uses a kernel size of 3 in each layer and includes circular padding to account for periodic boundary conditions. The CNN architecture is the same architecture used by Kochkov, et al.³⁹. We note that the input to all networks are size $N_x \times N_y \times M$, where N_x, N_y is the size of the filtered and coarse-grained DNS (FDNS) field or LES grid and M is the number of input channels – 2 in the case of invariant inputs and 3 in the case of non-invariant inputs. For the MLP, the grid dimensions N_x, N_y are taken as batch axes.

Thus far, another architectural choice has not yet been explored in the context of 2D turbulence modeling, the Fourier Neural Operator (FNO)⁶⁶. FNOs operate in both the real and frequency domain, combining local, pointwise linear operations with pointwise linear transformations of the Fourier modes of the input. From the convolution theorem, the pointwise operations in Fourier space are equivalent to global convolution operations, except the convolutional filter is no longer restricted to be locally supported, which allows for efficient manipulation of the large scales of the input without requiring very deep CNNs.

Assuming an input feature at layer l $h^l \in \mathbb{R}^{N_x \times N_y \times c_{in}}$, where c_{in} is the number of channels, the FNO layer is given by:

$$h_{nmo}^{l+1} = \sigma(B_{oi} h_{nmi}^l + \mathcal{F}^{-1}(W_{kloi} \mathcal{F}(h_{nmi}^l)_{kli})_{nmo}), \quad (16)$$

where $B \in \mathbb{R}^{c_{in} \times c_{out}}$ is a matrix that is contracted along dimension c_{in} of the input in real space and $W \in \mathbb{R}^{k_{max} \times k_{max} \times c_{in} \times c_{out}}$ is contracted along c_{in} pointwise in Fourier space (indexed by k, l for the wavenumbers in each direction) for each wavemode k up to k_{max} . The outputs are summed and a pointwise nonlinearity σ is applied. The parameters of the layer are the weight tensors W and B . We test this approach as well for f_θ , using the hyperparameters presented in the paper⁶⁶, which include 4 layers, $k_{max} = 12$, and a hidden channel width of 32.

The standard formulation of the FNO is ill-suited in the context of SGS modeling, in that the filters are only applied to the low wavenumbers of the flow. This means that the FNO is capturing the large scales of the flow but disregarding the small scales, which are typically more relevant when determining the SGS stress. We expect, then, that the vanilla FNO will not be able to reproduce the flow spectra accurately. To this end, we design an additional hybrid architecture that combines the benefits of both CNNs and FNOs. The input is first passed through a 2-layer FNO with 4 hidden channels and $k_{max} = 10$. The output is projected linearly to a size of 64 hidden channels. Finally, the projection is provided to a 3-layer CNN with otherwise equivalent specifications as the pure CNN architecture to produce the network output quantities. The hyperparameters for this model were chosen through empirical testing and optimized to reduce the memory footprint and computational cost of the architecture. We expect that this model will improve over the other networks because all the length scales in the flow are used to compute the SGS stress.

In summary, we test four different neural network architectures for f_θ . The MLP is local and receives no information regarding neighboring stress and rotation rate quantities. The CNN is in some sense semi-local, in that non-local $\bar{\mathbf{S}}$ and $\bar{\mathbf{R}}$ are provided within a specified receptive field. The FNO is global, but only operates on the large scales of $\bar{\mathbf{S}}$ and $\bar{\mathbf{R}}$. Lastly, the hybrid CNN+FNO ultimately captures all the scales of the flow by combining global information from an FNO and non-local information from a CNN.

D. Training

Model training is accomplished in an end-to-end fashion. Since the solver is implemented in JAX, gradients can be computed with respect to any solver parameter by backpropagating a loss using automatic differentiation. We define the simulation function \mathcal{S} as the numerical algorithm that computes solutions to eq. 4 along with the incompressibility constraint, using the SGS model given by eqs. 8 and 9. The solution of the ML-LES model is given by:

$$\hat{\mathbf{u}}^0 \dots \hat{\mathbf{u}}^t = \mathcal{S}_{ML}(\bar{\mathbf{u}}^0, C_s, \mathcal{M}(\theta)), \quad (17)$$

where $\hat{\mathbf{u}}^t$ is predicted solution at time t , C_s is the Smagorinsky constant, and θ are the parameters of the ANN inside \mathcal{M} .

The target solution is given by the coarse-grained DNS solution field which we hereby denote ‘subsamped DNS’:

$$\bar{\mathbf{u}}^0 \dots \bar{\mathbf{u}}^t = G_S \star \mathcal{S}_{DNS}(\mathbf{u}^0), \quad (18)$$

where \mathcal{S}_{DNS} is a high resolution simulation of the unfiltered initial condition and G_S is a coarse-graining operation. Here, we emphasize that the operation G_S makes no assumption of a low-pass spatial filter and

merely partially observes the high-dimensional state. The objective of the optimization problem is given as:

$$\min_{C_s, \theta} \mathcal{L}(\bar{\mathbf{u}}^0 \dots \bar{\mathbf{u}}^t, \mathcal{S}_{ML}(\bar{\mathbf{u}}^0, C_s, \mathcal{M}(\theta))), \quad (19)$$

where we aim to optimize both the Smagorinsky coefficient in τ_{smag} and the network parameters that produce τ_{ml} to minimize the loss function \mathcal{L} . We take the loss function to be the mean squared error of the trajectories:

$$\mathcal{L} = \frac{1}{T \cdot N_x \cdot N_y} \sum_{t=1}^T \sum_{n=1}^{N_x} \sum_{m=1}^{N_y} (\bar{\mathbf{u}}^t - \hat{\mathbf{u}}^t)^2. \quad (20)$$

The length of the simulation and temporal sampling of the solution trajectory is of particular interest for training. We would like the loss to remain low for very large T , but the memory costs incurred during backpropagation are linear with T . Therefore, generally $T_{train} \ll T_{eval}$, where T_{train} denotes the length of the simulation during training and T_{eval} denotes the length of simulation during evaluation or testing. As observed by List, et al., T_{train} should be large enough to capture the important temporal scales of the flow, but there are diminishing returns with very large T_{train} . We take $T_{train} = 512\Delta t_{DNS} = 64\Delta t_{LES}$, where Δt_{DNS} and Δt_{LES} are the time steps for DNS and LES simulations respectively. Furthermore, we choose a lower temporal sampling frequency for computing the loss function, such that $\Delta t_{train} = 128\Delta t_{DNS} = 16\Delta t_{LES}$, for a total of 4 snapshots per training trajectory. We found this sufficient for extrapolation to much larger T_{eval} , where we tested up to $T_{eval} = 131,072\Delta t_{DNS} = 16,384\Delta t_{LES}$, a total of $256\times$ longer than the training simulation.

The models were trained with the Adam optimizer⁶⁷ using a decaying learning rate varying from 2×10^{-3} to 4×10^{-4} on 8 V100 GPUs for 300 epochs. In addition, stochastic weight averaging⁶⁸ was used in the final 20% of training, where model weights were averaged over the last 20% of epochs to promote better generalization.

III. DATASETS

Two-dimensional homogeneous isotropic turbulence (2D-HIT)²⁰ provides an idealized case study to assess model performance. 2D-HIT is a common choice of testbed for training and deploying data-driven models, given its foundations in theory and ability to represent a variety of turbulent flow characteristics. The models are evaluated on several realizations of the same test case, which have been parameterized with respect to Reynolds number and forcing conditions. This yields a sufficiently diverse set of flows on which the generalization ability of the learned SGS models can be demonstrated. For all flow scenarios, the ground truth datasets are obtained from simulating the governing equations in a doubly periodic square domain with $L = 2\pi$, discretized on a uni-

TABLE II. Dataset parameters

Dataset	Re	A	k	r	\mathbf{f} direction
DE	1000	0	0	0	$\hat{\mathbf{e}}_x$
G1	1000	1	4	0.1	$\hat{\mathbf{e}}_x$
G2	30	1	4	0.1	$\hat{\mathbf{e}}_x$
G3	10^5	1	4	0.1	$\hat{\mathbf{e}}_x$
G4	8000	2	8	0.1	$\hat{\mathbf{e}}_y$

form grid of size 2048×2048 . The true fields are coarse-grained with G_S to produce target fields with a resolution of 64×64 .

The training dataset, denoted DE, consists of unforced, decaying turbulence at an initial Reynolds number of 1000. A total of 32 trajectories were used for training, each with length $T = 16,384\Delta t_{DNS}$. The trajectories were additionally temporally sampled at Δt_{train} to produce 128 snapshots per trajectory, further split into data samples of 4 snapshots each for training. For consistency, we fix $\Delta t_{DNS} = 2.191 \times 10^{-4}$ for all datasets, which was sufficient to satisfy the Courant–Friedrichs–Lewy (CFL) criterion with $C_{max} < 0.5$. The models are tested on four generalization datasets of forced turbulence. The forcing function \mathbf{f} is given by:

$$\mathbf{f} = A \sin(ky) \hat{\mathbf{e}}_x - r \mathbf{u}, \quad (21)$$

and parameterized by the amplitude A , wavenumber k , and linear drag r . We additionally change the forcing direction ($A \sin(kx) \hat{\mathbf{e}}_y$) in one dataset to ensure the models are not overfitting to a particular forcing orientation. A table of the datasets and their relevant parameters are given in Tab. II. We show example trajectories from each of the datasets and their energy spectrum in Fig. 2, averaged over time for the cases of statistically stationary forced turbulence. The filtered target field’s spectrum is also marked. For dataset G3, given the high Reynolds number of 10^5 , DNS is not feasible on a computational grid of 2048×2048 . Instead, the ground truth data is generated from a high resolution (2048×2048) LES simulation using the Smagorinsky turbulence model, and the target field is coarse-grained from this high-resolution solution.

IV. RESULTS AND DISCUSSION

A. Model sweep

In the interest of examining all model hyperparameter selections, we train 40 models in a grid search, corresponding to two choices of ϕ_{in} , four choices of f_θ , and five choices of ϕ_{out} . We first examine the models’ performance on generalization dataset G1 by measuring two quantities to probe the pointwise and statistical accuracy. The pointwise accuracy is determined from the Pearson correlation coefficient of the predicted velocity field with

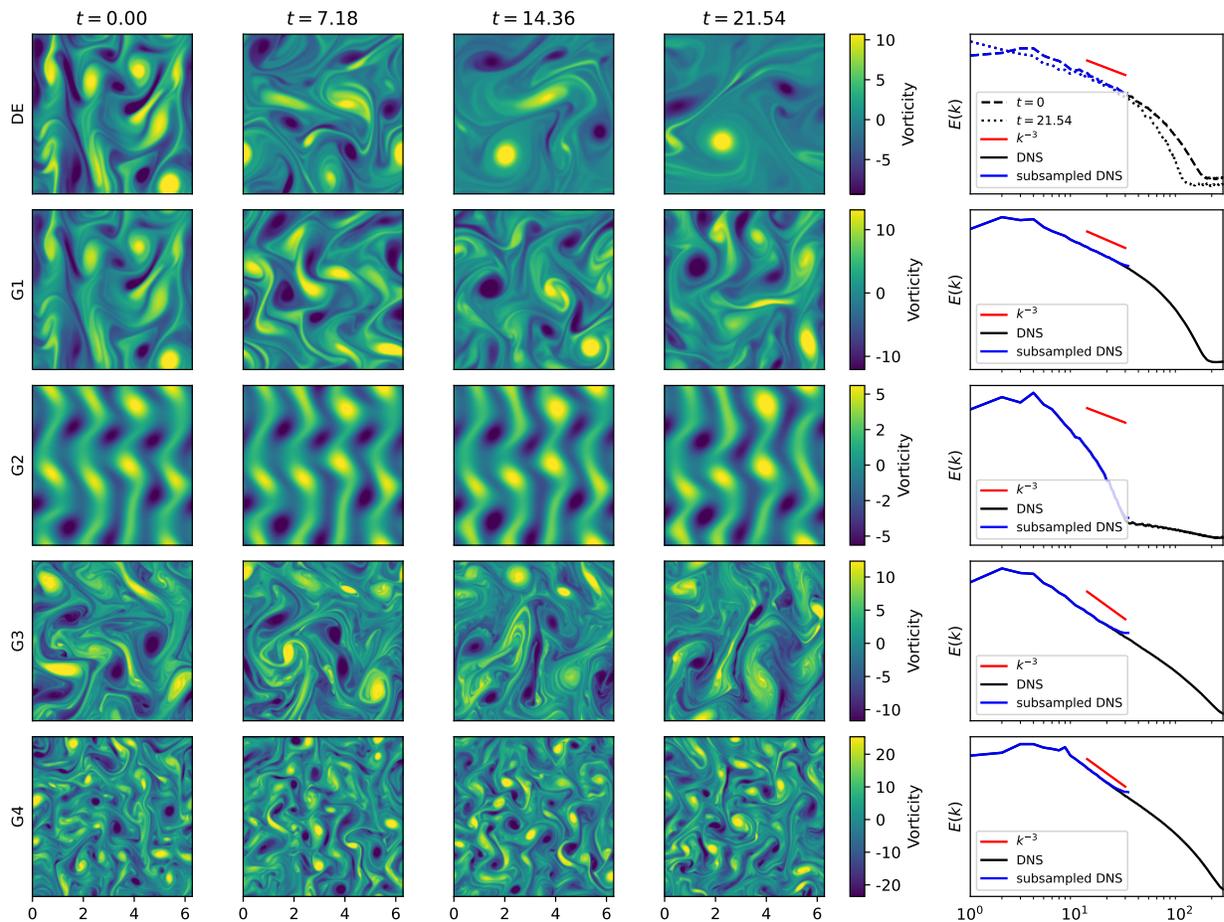


FIG. 2. Example ground truth vorticity field trajectories and energy spectra from each dataset. (DE) Unforced, decaying turbulence at an initial $Re = 1000$. This dataset was used for training. (G1) Forced turbulence using a sinusoidal forcing function with wavenumber $k = 4$, $Re = 1000$. (G2) Low Reynolds number flow, $Re = 30$, also forced at $k = 4$. (G3) High Reynolds number flow, $Re = 10^5$, forced at $k = 4$. Given the high Re , the ground truth was computed from LES on a 2048×2048 grid. (G4) Forced turbulence at a higher wavenumber $k = 8$, and a higher $Re = 8000$. Additionally, the direction of forcing is rotated 90 deg from the previous datasets. All plots show DNS spectra in addition to spectra obtained from subsampled observations of the DNS. The ideal k^{-3} scaling for the inverse cascade in 2D-HIT is also shown for reference.

respect to the ground truth FDNS velocity field. We report the time at which the correlation coefficient drops below 0.99, where higher values indicate greater pointwise accuracy. Pointwise accuracy is useful in certain situations, e.g. short-term weather forecasting, where it is valuable to match the true flow field exactly despite the chaotic nature of the system. In other scenarios, if pointwise accuracy cannot be guaranteed for long-term predictions, it is sufficient to ensure that the statistical behavior of the coarse-grained system is consistent with the ground truth statistics. Here, we report a statistical error based on the time-averaged energy spectrum $E(k)$ of the flow, which displays a characteristic scaling behavior in the inertial range of the spectrum. The error is

computed as:

$$\left(\frac{1}{K} \sum_{k=1}^K \log \left(\frac{\hat{E}(k)}{E(k)} \right)^2 \right)^{1/2}, \quad (22)$$

where $\hat{E}(k)$ is the predicted energy spectrum and $E(k)$ is the energy spectrum of the DNS field. Importantly, the errors at all wavenumbers are weighted equally in this formulation, as opposed to e.g. mean-squared error (MSE), which would favor accuracy at the higher energy-containing low wavenumbers. We also perform the time-averaging after the prediction has decorrelated with the ground truth trajectory.

Fig. 3 shows the pointwise and statistical measures on the G1 dataset of all models tested. The left two grids and right two grids display the pointwise accuracy and statistical error respectively. Examining the point-

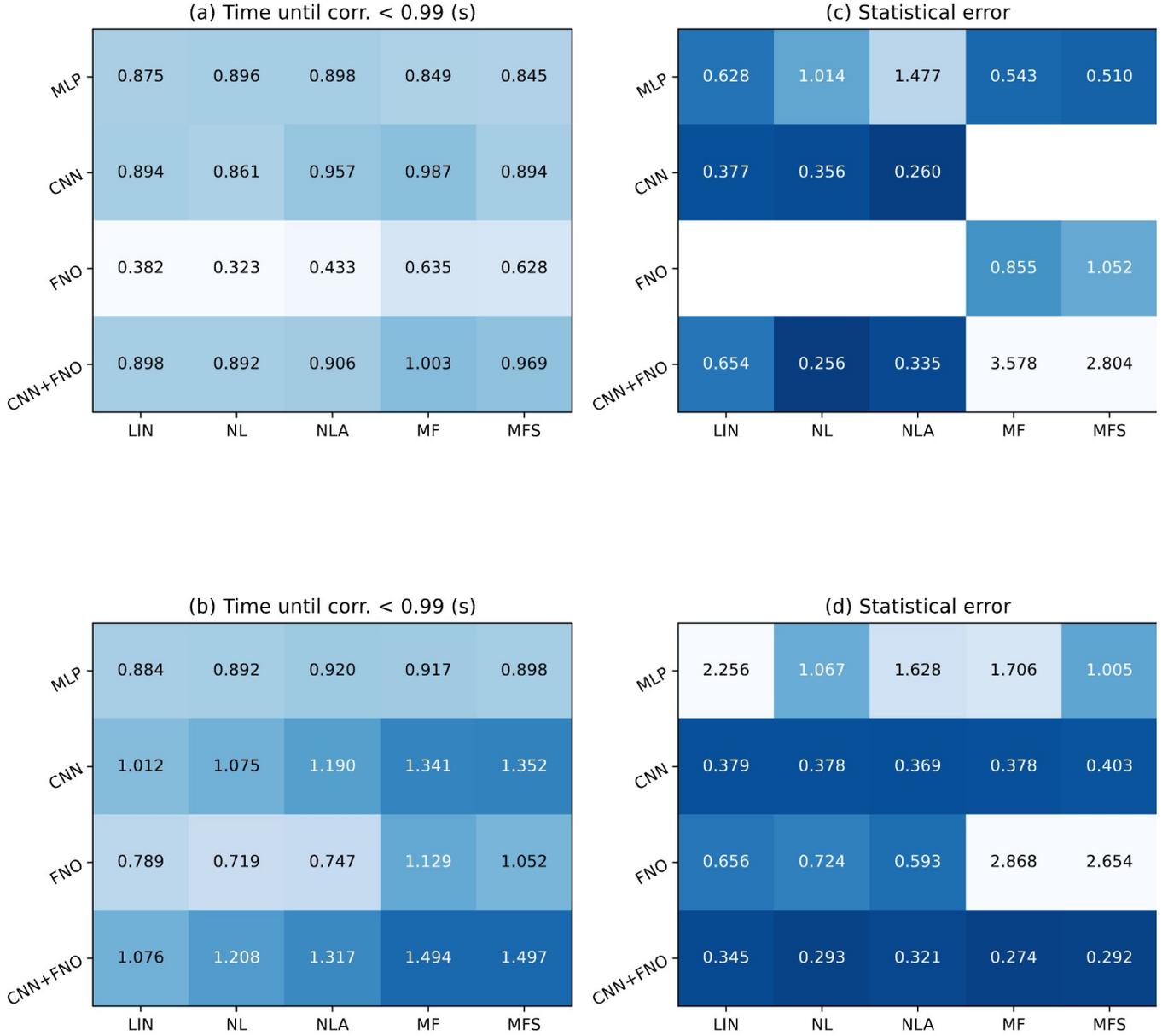


FIG. 3. Ensemble average pointwise and statistical metrics of the predicted solution fields from each of the 40 models tested. (a) and (b) report the pointwise accuracy of invariant input and non-invariant input SGS models respectively, computed as the time before correlation with the ground truth drops below 0.99, where higher values indicate better agreement. (c) and (d) report the statistical error of invariant input and non-invariant input SGS models respectively, computed from the deviation of predicted and true energy spectra, where lower values indicate better agreement. The CNN+FNO-MF and -MFS models achieve the highest pointwise accuracy and the non-invariant input CNN and CNN+FNO models achieve the lowest statistical error. The use of invariant inputs generally results in poorer performance for the non-local architectures, but the reverse is true for the local MLP.

wise accuracy, we find that the use of invariant inputs (the upper row of models) negatively impacts accuracy, although the impact is nearly insignificant for the MLP models. Overall, the CNN and CNN+FNO architectures achieve the highest accuracy, particularly in combination

with MF or MFS. Looking at the statistical error, again the CNN and CNN+FNO architectures achieve the lowest error, but only in conjunction with non-invariant inputs. Many of the non-local architectures with invariant inputs resulted in unstable simulations that diverged

TABLE III. Evaluation time for fixed forecast length

Model	Time (s)
DNS	137 ± 0.490
SMAG	2.28 ± 0.071
MLP	3.35 ± 0.129
CNN	5.32 ± 0.148
FNO	6.81 ± 0.093
CNN+FNO	5.75 ± 0.016

before time-averaging could be performed; as a result, the error for these models is not reported. In contrast, the use of invariant inputs is in fact beneficial for the purely local MLP models. We conclude that the orientation of the velocity gradient tensor, provided by the non-invariant inputs, is useful and perhaps necessary information to provide when constructing non-local data-driven SGS models, but harmful when given to local SGS models. We also see that the FNO-only architecture generally performs most poorly, likely because it cannot capture the small-scale behavior of the flow that is relevant for determining the SGS stress. Ultimately, the CNN+FNO-MFS model achieves the best combination of pointwise and statistical accuracy, and we select this as the best performing model for subsequent analysis.

We report the evaluation times of several models for a fixed forecast length of $2^{17}\Delta t_{DNS}$ in Tab. III. Since the ANN architecture was determined to be the most significant contributor to variable runtime, we compare the four architectures against the ground truth DNS and the baseline Smagorinsky closure using a coefficient of 0.172. The MLP is cheapest to compute, while the FNO is the most expensive. Both the CNN and CNN+FNO had similar runtimes, a little over double the baseline model.

B. Spectral analysis of architectures

To further understand the impact, mechanisms, and spectral behavior of the neural network architectures, we can plot the radially averaged power spectrum of the network outputs. The power spectrum is computed from integrating the squared 2D Fourier transform of the network outputs over constant magnitude wavenumbers $k = \sqrt{k_x^2 + k_y^2}$. The spectrum can assist in revealing which length scales in the flow each of the network architectures is responding to. We start by isolating f_θ from the rest of the components of the SGS model. To provide a consistent input baseline, we probe the spectral response to white noise, which has a uniform power spectrum. We examine MFS trained models with different architectures and plot α , the isotropic component of the SGS stress, and $|\mathbf{D}|$, the magnitude of the deviatoric component of the SGS stress. While these outputs have no physical significance regarding the true SGS stress con-

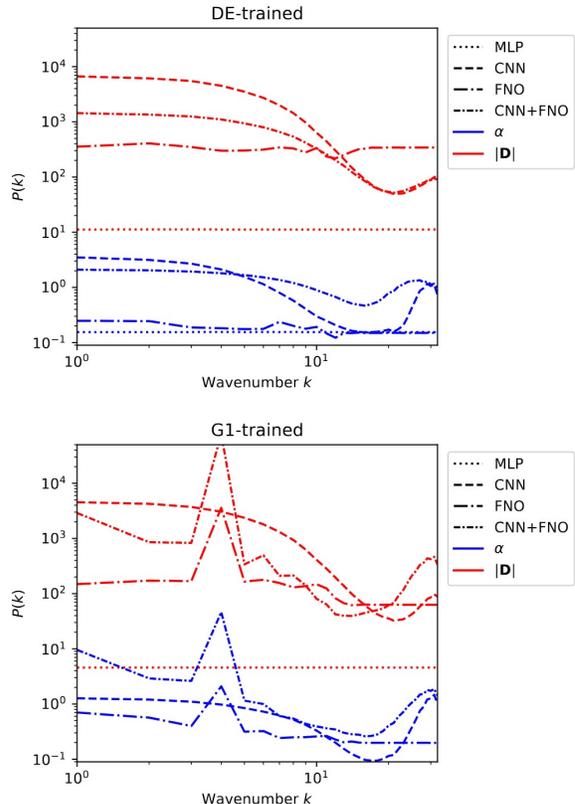


FIG. 4. Examination of the power spectral response of neural architectures to white noise. Outputs of MFS model networks trained on (a) the DE dataset and (b) the forced G1 dataset are shown, with the isotropic component α and magnitude of the deviatoric component $|\mathbf{D}|$ visualized. In (a) while the MLP outputs are flat at all wavenumbers and the FNO outputs are flat at high wavenumbers, the CNN and CNN+FNO models display more significant variation at the high wavenumbers, leveling out at the low wavenumbers. In (b) the FNO and CNN+FNO display sharp peaks at the forcing wavenumber, indicating their ability to capture (or overfit to) large-scale features.

sidering the synthetic input, they can provide insight into the scales captured by the model. We would expect, for example, an MLP to produce entirely flat power spectra as it only operates on local quantities and contains no interactions between length scales. We also expect the FNO power to be uniform at high wavenumbers because the FNO cannot capture any length scales smaller than L/k_{max} , where $k_{max} = 12$.

Fig. 4(a) shows the power spectra of various networks that have been trained on the DE dataset. We see that our expectations regarding the MLP and FNO are validated, in that the MLP power is uniform for all wavenumbers and the FNO power is uniform at wavenumbers greater than 12. On the other hand, we see that both the CNN and CNN+FNO networks display significantly more variation in the power spectra at high wavenum-

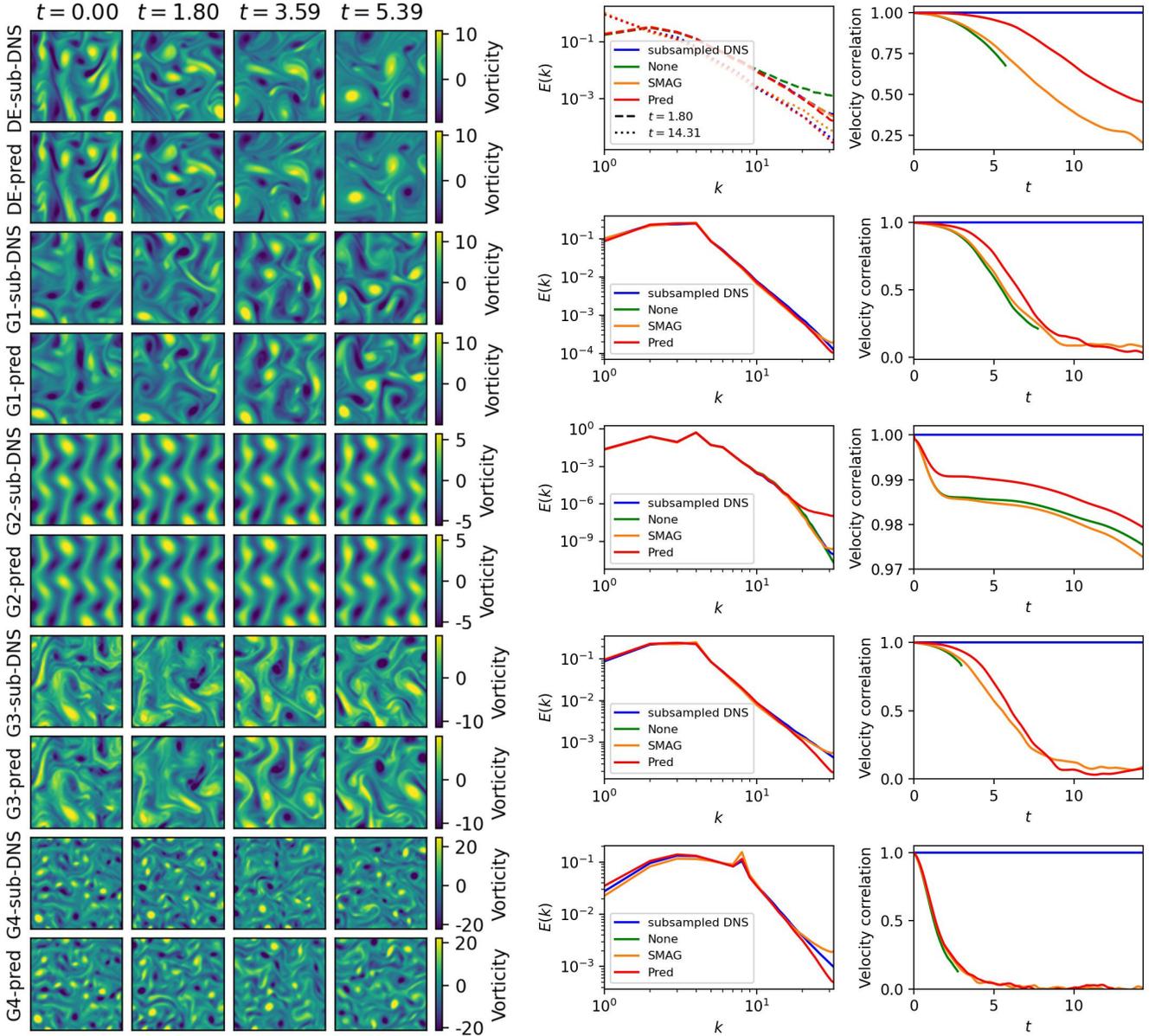


FIG. 5. Summary of predictions from the CNN+FNO-MFS model. The left column displays an example predicted vorticity trajectory for each dataset compared with the true field. The right column shows the ensemble average energy spectra and velocity correlation over time for each dataset. Comparisons to baseline SMAG and no turbulence model are included. The CNN+FNO-MFS model improves over the baseline in both metrics for nearly every case. Only in the low Reynolds number dataset G2 does the model overpredict energy at the highest wavenumbers.

bers, but are much more uniform at low wavenumbers. This indicates that the CNN models are mostly responding to the smallest length scales in the flow, which is consistent with the network design. Considering the accuracy of the CNN and CNN+FNO models compared to the FNO models, it is clear that the smallest length scales in the flow are most important for determining the SGS stress, again aligned with our expectations regarding turbulence.

We can contrast this with the spectral response of networks trained on the G1 dataset, which is forced at a wavenumber of 4. The FNO-containing models display a very clear peak at this forcing wavenumber, which indicates that they are overfitting to the training set. Because the FNO operates directly in spectral space, it can easily capture large scale behavior, such as external forcing. The CNN, however, has no such peak since its small receptive field limits its ability to respond to

the low frequency modes of the flow. There is a trade-off, then, in terms of incorporating very non-local, i.e. large scale, features in the SGS model; while this will lead to improved accuracy on in-distribution test cases, it will likely hamper performance on out-of-distribution generalization cases.

C. Generalization to new flows

A turbulence model’s value is strongly tied to its versatility. A significant reason the Smagorinsky model is still widely used in practice today despite its simplicity is that it is effective in a very wide variety of flow configurations. Oftentimes data-driven models cannot match the flexibility of theory-based models because they are trained only on a specific distribution of data. Here, we demonstrate the versatility of our model design by applying, without retraining, the same DE-trained SGS model to each of the four generalization datasets. Fig. 5 shows a summary of the model’s performance on all datasets, where we have selected the CNN+FNO-MFS model as the data-driven model to test. An example trajectory for each dataset, both the true field and the predicted field, is shown on the left. We observe qualitatively accurate predictions for over 3000 LES time steps in nearly all cases, except for G4, where the eddy turnover time is much shorter due to the smaller characteristic length scale. On the right, we show the energy spectra and transient velocity field correlation, comparing statistics from the true filtered field, no model, Smagorinsky model, and our ML model. The ML model matches the true energy spectrum and improves over the Smagorinsky baseline in nearly all datasets. The exception is the very low Reynolds number case, G2, where the model overestimates the energy at the smallest length scales. This is characteristic of the use of turbulence models when DNS is capable of resolving the flow, exemplified by a similar trend in the Smagorinsky model compared to no model. We note that the energy at these high wavenumbers is very low, and there is no significant impact on the pointwise accuracy. The velocity correlation plots show that the ML model achieves greater pointwise accuracy than the Smagorinsky model for every dataset, with the greatest improvement predictably seen on the DE dataset.

D. Comparison with *a-priori* approach

Lastly, we compare our end-to-end, differentiable physics approach to learning SGS models with a more conventional offline learning approach. In the *a-priori* setting, the learning task is to match the SGS stress field computed from the true DNS solution using an arbitrary filter, which can be accomplished without integrating the ML model into a simulation during training. This approach eliminates the initial overhead of developing a differentiable solver and the training cost of backpropa-

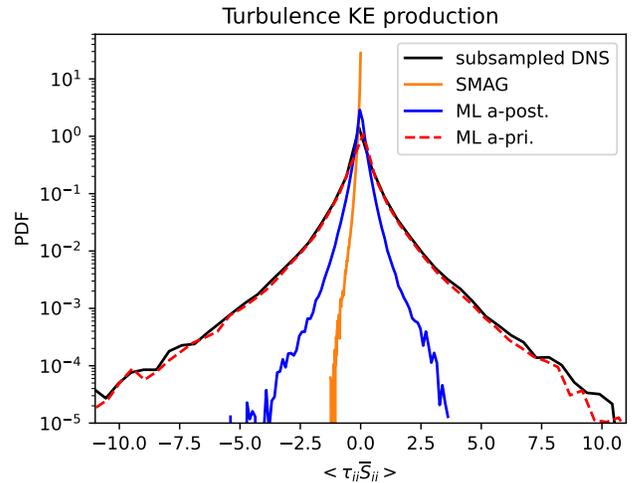


FIG. 6. The *a-priori* computed turbulence kinetic energy production PDF is visualized for the *a-priori* trained and *a-posteriori* trained ML models, as well as the true DNS and SMAG baseline. While SMAG is purely dissipative, the ML models can replicate backscatter, indicated by both positive and negative values.

gating through the simulation, but there are implications in terms of *a-posteriori* accuracy. When deployed online, numerical stability is typically lost and the choice of numerical scheme may lead to unexpected *a-posteriori* performance. Usually, this instability is closely associated with the choice of a filtering operation G_{Δ}^* that approximates some true, but unknown, G_{Δ} .

To compare our differentiable physics approach with an *a priori* trained model, we must first make an assumption for a low-pass spatial filter to compute the target subgrid stress. This procedure is explained in the following. We train an equivalent CNN+FNO-MFS model in an *a-priori* setting and evaluate both *a-priori* and *a-posteriori* accuracy on the G1 dataset. The SGS model definition is unchanged, given by eqs. 8 and 9, but the loss function is more straightforward, simply:

$$\mathcal{L} = MSE(\hat{\boldsymbol{\tau}}, \boldsymbol{\tau}_{FDNS}), \quad (23)$$

where $\hat{\boldsymbol{\tau}}$ is the predicted SGS stress and $\boldsymbol{\tau}_{FDNS}$ is the SGS stress computed from the DNS field using eqn. 5 with an approximate filter G_{Δ}^* . Here we utilize a popular filtering technique³⁹ to compute the mean of the values that lie on the face of the new control volume in a given direction and discard the values that do not lie in that particular direction. This ensures that zero divergence properties are propagated to the downsampled fields too. In such a manner, we can filter ground truth DNS fields which can be followed by subsampling to the coarse grid. Ultimately, we can generate target subgrid stress data on the coarse-grid. Note that different choices of filters may lead to different $\boldsymbol{\tau}_{FDNS}$ computations.

Fig. 6 shows the probability distribution function (PDF) of the *a-priori* computed turbulent kinetic en-

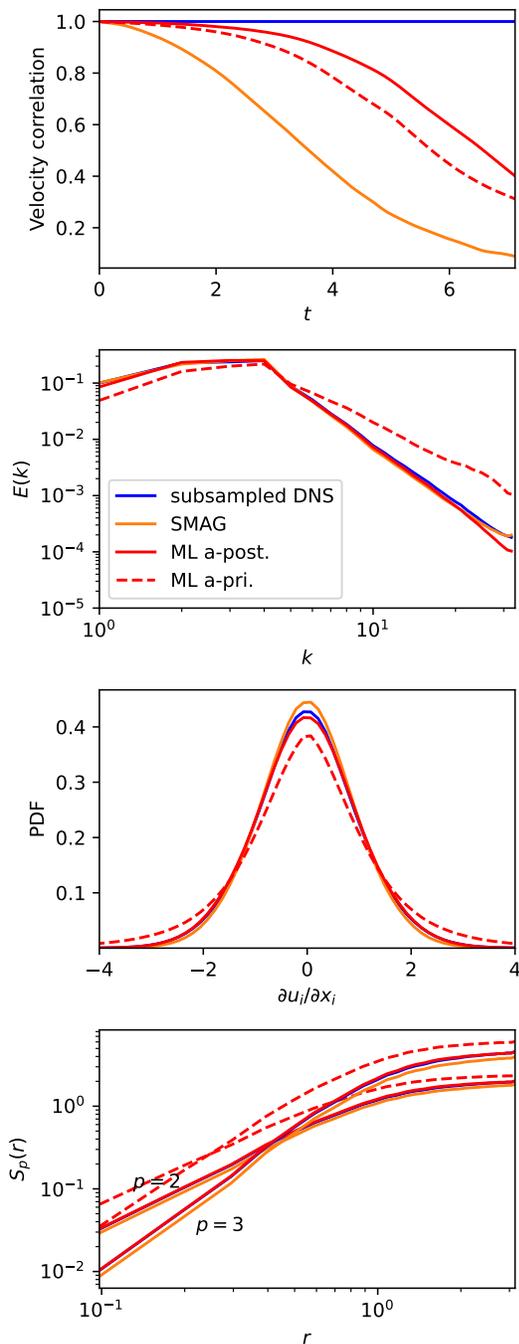


FIG. 7. Selected *a-posteriori* statistics from results on G1 dataset comparing *a-priori* trained and *a-posteriori* trained ML models. (a) velocity correlation with FDNS. The *a-priori* model displays only marginal improvement over SMAG. (b) averaged energy spectra of the flow. While there is near perfect agreement with the DNS from the *a-posteriori* trained model, the *a-priori* model displays more deviation from the ground truth. (c) PDF of velocity gradients. The *a-priori* model has a slightly narrower distribution than the DNS. (d) second and third order structure functions. Both SMAG and the *a-posteriori* model display no noticeable disagreement with the DNS, in contrast to the *a-priori* model.

ergy production term, given by $\langle \tau_{ij} \bar{S}_{ij} \rangle$, computed from the DNS data with the aforementioned approximate filter G_{Δ}^* , the Smagorinsky model, the *a-posteriori* trained ML model, and the *a-priori* trained ML model. First, we notice that because the Smagorinsky model is purely dissipative, the production term is always negative. In addition, the distribution is much narrower than the true distribution. On the other hand, the *a-priori* trained predictions match this approximately generated PDF exactly. The *a-posteriori* model's PDF is generated by simply computing the turbulent kinetic energy production term using the grid-resolved quantities during LES deployment. Surprisingly, this PDF is dramatically different from that computed using the approximately filtered data in *a-priori*. **This plot demonstrates that *a-priori* trained machine learning closures effectively solve the wrong problem by making extremely strong assumptions about the nature of the true filter which is unknown.** Moreover, it also becomes clear that the differentiable physics paradigm can be interpreted as an inverse problem solve, that discovers the nature of the true subgrid stress in the absence of assumptions about the filter.

To further validate this claim - we analyze other statistical measures in Fig. 7. The *a-priori* trained model as-is was found to be unstable and as such, it was necessary to introduce a clipping method, similar to what was suggested by Maulik, et al.²⁰, to ensure the SGS model was purely dissipative. Fig. 7(a) shows a comparison of the average energy spectra, Fig. 7(b) shows the PDFs of velocity gradients, Fig. 7(c) shows the second- and third-order structure functions, and Fig. 7(d) shows the velocity field correlation over time. For all measures, we see deviations in the *a-priori* model from the filtered DNS field, often greater than those from the Smagorinsky model, while the *a-posteriori* model shows near perfect agreement. We find that although there are some marginal benefits to training SGS models in an *a-priori* setting, these are substantially outweighed by the improvements offered from end-to-end differentiable solvers.

V. CONCLUSIONS

In this work, we seek to learn subgrid-scale models for LES using differentiable turbulence, by leveraging a differentiable CFD solver paired with deep learning to minimize an *a-posteriori* loss function. In LES, the large scales of the flow are directly resolved while the small scales are modeled; the effect of the unresolved small scales on the large scales must be accounted for through the use of an SGS model to produce accurate solution trajectories. Filtering out the small scales of the solution results in an additional effective SGS stress within the governing equations, which we approximate with a deep learning model. Given that the goal of LES is to realize simulations that can accurately predict the filtered and coarse-grained true solution field, we use an adjoint

solver implementation to optimize the *a-posteriori* solution directly, instead of the optimizing the SGS stress computed from a resolved simulation *a-priori*.

Models are evaluated on a variety of two-dimensional turbulent flows. While models are trained on decaying turbulent flows with an initial Reynolds number of 1000, they are tested on turbulent flow with different forcing conditions and Reynolds numbers that span four orders of magnitude. We find that our approach can generalize well to each test case, and improves over the Smagorinsky closure baseline at a fixed coefficient of 0.172 in every scenario. Model design is investigated in detail, including the choice of ANN inputs, outputs, and architecture. We observe that non-linear eddy viscosity model approaches are not as effective as directly learning the mapping between filtered velocity gradients and SGS stresses, which can be achieved through decomposition of the tensors into isotropic, deviatoric, and anti-symmetric components. Furthermore, the non-locality of the network architecture strongly influences model performance, where it is most critical to include small scale interactions near the grid size. The insertion of large scale behavior through an FNO boosts the accuracy of predictions, but may come at the expense of generalization ability depending on the training data.

We demonstrate the benefits of a fully differentiable solver by outlining the improvements in results from *a-posteriori* learning when compared to *a-priori* learning. Given the accuracy that can be achieved even at very large coarse-graining factors, we believe hybrid deep learning-solver algorithms are a very encouraging avenue for cheap and accurate CFD simulations. The incorporation of physics through the solver allows significantly more potential for out-of-distribution performance relative to pure ML algorithms. While this preliminary study has been restricted to idealized flows with a specific discretization, we anticipate future work to investigate the application of this approach to more complex wall-bounded flows on unstructured grids.

VI. DATA AVAILABILITY

Data and software utilized for this study will be made available by the corresponding author on reasonable request.

VII. ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE 1745016 awarded to VS. The authors from CMU and Michigan acknowledge the support from the Technologies for Safe and Efficient Transportation University Transportation Center, and Mobility21, A United States Department of Transportation National University Transportation Center. This work was sup-

ported in part by Oracle Cloud credits and related resources provided by the Oracle for Research program. This work was supported by the U.S. Department of Energy (DOE), Office of Science, Office of Advanced Scientific Computing Research (ASCR), under Contract No. DEAC02-06CH11357, at Argonne National Laboratory and Penn State University. We also acknowledge funding support from ASCR for DOE-FOA-2493 “Data-intensive scientific machine learning”

- ¹S. B. Pope, *Turbulent Flows* (Cambridge University Press, 2000).
- ²H. Tennekes and J. L. Lumley, *A First Course in Turbulence*, The MIT Press (MIT Press, London, England, 2018).
- ³P. Moin and K. Mahesh, “DIRECT NUMERICAL SIMULATION: A tool in turbulence research,” *Annual Review of Fluid Mechanics* **30**, 539–578 (1998).
- ⁴C. Argyropoulos and N. Markatos, “Recent advances on the numerical modelling of turbulent flows,” *Applied Mathematical Modelling* **39**, 693–732 (2015).
- ⁵T. Gatski and C. Rumsey, “Linear and nonlinear eddy viscosity models,” in *Closure Strategies for Turbulent and Transitional Flows* (Cambridge University Press, 2001) pp. 9–46.
- ⁶P. Sagaut, *Large Eddy Simulation for Incompressible Flows*, 3rd ed., Scientific Computation (Springer, New York, NY, 2006).
- ⁷S. B. Pope, “Ten questions concerning the large-eddy simulation of turbulent flows,” *New Journal of Physics* **6**, 35–35 (2004).
- ⁸J. Schalkwijk, H. J. J. Jonker, A. P. Siebesma, and E. V. Meijgaard, “Weather forecasting using GPU-based large-eddy simulations,” *Bulletin of the American Meteorological Society* **96**, 715–723 (2015).
- ⁹Z. Shen, A. Sridhar, Z. Tan, A. Jaruga, and T. Schneider, “A library of large-eddy simulations forced by global climate models,” *Journal of Advances in Modeling Earth Systems* **14** (2022), 10.1029/2021ms002631.
- ¹⁰P. R. Spalart, “Philosophies and fallacies in turbulence modeling,” *Progress in Aerospace Sciences* **74**, 1–15 (2015).
- ¹¹B. J. Geurts and J. Fröhlich, “A framework for predicting accuracy limitations in large-eddy simulation,” *Physics of Fluids* **14**, L41–L44 (2002).
- ¹²W. C. Reynolds, “The potential and limitations of direct and large eddy simulations,” in *Whither Turbulence? Turbulence at the Crossroads* (Springer Berlin Heidelberg) pp. 313–343.
- ¹³K. Duraisamy, G. Iaccarino, and H. Xiao, “Turbulence modeling in the age of data,” *Annual Review of Fluid Mechanics* **51**, 357–377 (2019).
- ¹⁴A. Beck and M. Kurz, “A perspective on machine learning methods in turbulence modeling,” *GAMM-Mitteilungen* **44** (2021), 10.1002/gamm.202100002.
- ¹⁵J. N. Kutz, “Deep learning in fluid dynamics,” *Journal of Fluid Mechanics* **814**, 1–4 (2017).
- ¹⁶J. Ling, A. Kurzawski, and J. Templeton, “Reynolds averaged turbulence modelling using deep neural networks with embedded invariance,” *Journal of Fluid Mechanics* **807**, 155–166 (2016).
- ¹⁷W. Liu, J. Fang, S. Rolfo, C. Moulinec, and D. R. Emerson, “An iterative machine-learning framework for RANS turbulence modeling,” *International Journal of Heat and Fluid Flow* **90**, 108822 (2021).
- ¹⁸A. P. Singh, K. Duraisamy, and Z. J. Zhang, “Augmentation of turbulence models using field inversion and machine learning,” in *55th AIAA Aerospace Sciences Meeting* (American Institute of Aeronautics and Astronautics, 2017).
- ¹⁹Y. Bin, L. Chen, G. Huang, and X. I. A. Yang, “Progressive, extrapolative machine learning for near-wall turbulence modeling,” *Physical Review Fluids* **7** (2022), 10.1103/physrevfluids.7.084610.
- ²⁰R. Maulik, O. San, A. Rasheed, and P. Vedula, “Subgrid modelling for two-dimensional turbulence using neural networks,” *Journal of Fluid Mechanics* **858**, 122–144 (2018).

- ²¹Z. Wang, K. Luo, D. Li, J. Tan, and J. Fan, “Investigations of data-driven closure for subgrid-scale stress in large-eddy simulation,” *Physics of Fluids* **30**, 125101 (2018).
- ²²H. Frezat, G. Balarac, J. L. Sommer, R. Fablet, and R. Lguensat, “Physical invariance in neural networks for subgrid-scale scalar flux modeling,” *Physical Review Fluids* **6** (2021), 10.1103/physrevfluids.6.024607.
- ²³Y. Guan, A. Chattopadhyay, A. Subel, and P. 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).
- ²⁴Y. Guan, A. Subel, A. Chattopadhyay, and P. Hassanzadeh, “Learning physics-constrained subgrid-scale closures in the small-data regime for stable and accurate LES,” *Physica D: Nonlinear Phenomena* **443**, 133568 (2023).
- ²⁵A. Subel, Y. Guan, A. Chattopadhyay, and P. Hassanzadeh, “Explaining the physics of transfer learning in data-driven turbulence modeling,” *PNAS Nexus* **2** (2023), 10.1093/pnasnexus/pgad015.
- ²⁶U. Piomelli, W. H. Cabot, P. Moin, and S. Lee, “Subgrid-scale backscatter in turbulent and transitional flows,” *Physics of Fluids A: Fluid Dynamics* **3**, 1766–1771 (1991).
- ²⁷X. Fan and J.-X. Wang, “Differentiable hybrid neural modeling for fluid-structure interaction,” *Journal of Computational Physics* **496**, 112584 (2024).
- ²⁸J. Zhao and Q. Li, “Mitigating distribution shift in machine learning-augmented hybrid simulation,” arXiv preprint arXiv:2401.09259 (2024).
- ²⁹B. Sanderse, P. Stinis, R. Maulik, and S. E. Ahmed, “Scientific machine learning for closure models in multiscale problems: a review,” arXiv preprint arXiv:2403.02913 (2024).
- ³⁰G. Novati, H. L. de Laroussilhe, and P. Koumoutsakos, “Automating turbulence modelling by multi-agent reinforcement learning,” *Nature Machine Intelligence* **3**, 87–96 (2021).
- ³¹Y. Zhao, H. D. Akolekar, J. Weatheritt, V. Michelassi, and R. D. Sandberg, “RANS turbulence model development using CFD-driven machine learning,” *Journal of Computational Physics* **411**, 109413 (2020).
- ³²O. Obiols-Sales, A. Vishnu, N. Malaya, and A. Chandramowlishwaran, “CFDNet,” in *Proceedings of the 34th ACM International Conference on Supercomputing* (ACM, 2020).
- ³³S. Pandey, J. Schumacher, and K. R. Sreenivasan, “A perspective on machine learning in turbulent flows,” *Journal of Turbulence* **21**, 567–584 (2020).
- ³⁴R. Vinuesa and S. L. Brunton, “Enhancing computational fluid dynamics with machine learning,” *Nature Computational Science* **2**, 358–366 (2022).
- ³⁵K. Stachenfeld, D. B. Fielding, D. Kochkov, M. Cranmer, T. Pfaff, J. Godwin, C. Cui, S. Ho, P. Battaglia, and A. Sanchez-Gonzalez, “Learned coarse models for efficient turbulence simulation,” (2022), arXiv:2112.15275 [physics.flu-dyn].
- ³⁶A. T. Mohan and D. V. Gaitonde, “A deep learning based approach to reduced order modeling for turbulent flow control using lstm neural networks,” (2018), arXiv:1804.09269 [physics.comp-ph].
- ³⁷M. Bode, M. Gauding, J. H. Göbber, B. Liao, J. Jitsev, and H. Pitsch, “Towards prediction of turbulent flows at high reynolds numbers using high performance computing data and deep learning,” in *Lecture Notes in Computer Science* (Springer International Publishing, 2018) pp. 614–623.
- ³⁸R. Wang, K. Kashinath, M. Mustafa, A. Albert, and R. Yu, “Towards physics-informed deep learning for turbulent flow prediction,” in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (ACM, 2020).
- ³⁹D. Kochkov, J. A. Smith, A. Alieva, Q. Wang, M. P. Brenner, and S. Hoyer, “Machine learning-accelerated computational fluid dynamics,” *Proceedings of the National Academy of Sciences* **118** (2021), 10.1073/pnas.2101784118.
- ⁴⁰P. Holl, N. Thuerey, and V. Koltun, “Learning to control pdes with differentiable physics,” in *International Conference on Learning Representations* (2020).
- ⁴¹D. A. Bezgin, A. B. Buhendwa, and N. A. Adams, “Jax-fluids: A fully-differentiable high-order computational fluid dynamics solver for compressible two-phase flows,” *Computer Physics Communications*, 108527 (2022).
- ⁴²H. Frezat, G. Balarac, J. L. Sommer, and R. Fablet, “Gradient-free online learning of subgrid-scale dynamics with neural emulators,” arXiv preprint arXiv:2310.19385 (2023).
- ⁴³A. Jameson, “Optimum aerodynamic design using CFD and control theory,” in *12th Computational Fluid Dynamics Conference* (American Institute of Aeronautics and Astronautics, 1995).
- ⁴⁴J.-D. Müller and P. Cusdin, “On the performance of discrete adjoint CFD codes using automatic differentiation,” *International Journal for Numerical Methods in Fluids* **47**, 939–945 (2005).
- ⁴⁵G. K. Kenway, C. A. Mader, P. He, and J. R. Martins, “Effective adjoint approaches for computational fluid dynamics,” *Progress in Aerospace Sciences* **110**, 100542 (2019).
- ⁴⁶O. Tonomura, M. Kano, and S. Hasebe, “Shape optimization of microchannels using CFD and adjoint method,” in *Computer Aided Chemical Engineering* (Elsevier, 2010) pp. 37–42.
- ⁴⁷W. Liu, R. Duan, C. Chen, C.-H. Lin, and Q. Chen, “Inverse design of the thermal environment in an airliner cabin by use of the CFD-based adjoint method,” *Energy and Buildings* **104**, 147–155 (2015).
- ⁴⁸M. P. Rumpfkeil and D. W. Zingg, “The optimal control of unsteady flows with a discrete adjoint method,” *Optimization and Engineering* **11**, 5–22 (2008).
- ⁴⁹M. B. Giles and N. A. Pierce, “An introduction to the adjoint approach to design,” *Flow, Turbulence and Combustion* **65**, 393–415 (2000).
- ⁵⁰K. Um, R. Brand, Yun, Fei, P. Holl, and N. Thuerey, “Solver-in-the-loop: Learning from differentiable physics to interact with iterative pde-solvers,” (2021), arXiv:2007.00016 [physics.comp-ph].
- ⁵¹B. List, L.-W. Chen, and N. Thuerey, “Learned turbulence modelling with differentiable fluid solvers: physics-based loss functions and optimisation horizons,” *Journal of Fluid Mechanics* **949** (2022), 10.1017/jfm.2022.738.
- ⁵²J. Sirignano, J. F. MacArt, and J. B. Freund, “DPM: A deep learning PDE augmentation method with application to large-eddy simulation,” *Journal of Computational Physics* **423**, 109811 (2020).
- ⁵³H. Frezat, J. L. Sommer, R. Fablet, G. Balarac, and R. Lguensat, “A posteriori learning for quasi-geostrophic turbulence parametrization,” *Journal of Advances in Modeling Earth Systems* **14** (2022), 10.1029/2022ms003124.
- ⁵⁴J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, “JAX: composable transformations of Python+NumPy programs,” (2018).
- ⁵⁵C. A. Mader, J. R. R. A. Martins, J. J. Alonso, and E. van der Weide, “ADjoint: An approach for the rapid development of discrete adjoint solvers,” *AIAA Journal* **46**, 863–873 (2008).
- ⁵⁶Z. Zhou, G. He, S. Wang, and G. Jin, “Subgrid-scale model for large-eddy simulation of isotropic turbulent flows using an artificial neural network,” *Computers & Fluids* **195**, 104319 (2019).
- ⁵⁷J. SMAGORINSKY, “GENERAL CIRCULATION EXPERIMENTS WITH THE PRIMITIVE EQUATIONS,” *Monthly Weather Review* **91**, 99–164 (1963).
- ⁵⁸Y. Zang, R. L. Street, and J. R. Koseff, “A dynamic mixed subgrid-scale model and its application to turbulent recirculating flows,” *Physics of Fluids A: Fluid Dynamics* **5**, 3186–3196 (1993).
- ⁵⁹D. K. Lilly, “A proposed modification of the germano subgrid-scale closure method,” *Physics of Fluids A: Fluid Dynamics* **4**, 633–635 (1992).
- ⁶⁰N. Adams and S. Stolz, “A subgrid-scale deconvolution approach for shock capturing,” *Journal of Computational Physics* **178**, 391–426 (2002).

- ⁶¹P. M. Milani, J. Ling, and J. K. Eaton, “On the generality of tensor basis neural networks for turbulent scalar flux modeling,” *International Communications in Heat and Mass Transfer* **128**, 105626 (2021).
- ⁶²S. Berrone and D. Oberto, “An invariances-preserving vector basis neural network for the closure of reynolds-averaged navier–stokes equations by the divergence of the reynolds stress tensor,” *Physics of Fluids* **34**, 095136 (2022).
- ⁶³S. B. Pope, “A more general effective-viscosity hypothesis,” *Journal of Fluid Mechanics* **72**, 331 (1975).
- ⁶⁴R. Stoffer, C. M. van Leeuwen, D. Podareanu, V. Codreanu, M. A. Veerman, M. Janssens, O. K. Hartogensis, and C. C. van Heerwaarden, “Development of a large-eddy simulation sub-grid model based on artificial neural networks: a case study of turbulent channel flow,” *Geoscientific Model Development* **14**, 3769–3788 (2021).
- ⁶⁵P. C. D. Leoni, T. A. Zaki, G. Karniadakis, and C. Meneveau, “Two-point stress–strain-rate correlation structure and non-local eddy viscosity in turbulent flows,” *Journal of Fluid Mechanics* **914** (2021), 10.1017/jfm.2020.977.
- ⁶⁶Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, “Fourier neural operator for parametric partial differential equations,” (2021), arXiv:2010.08895 [cs.LG].
- ⁶⁷D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” (2017), arXiv:1412.6980 [cs.LG].
- ⁶⁸P. Izmailov, D. Podoprikin, T. Garipov, D. Vetrov, and A. G. Wilson, “Averaging weights leads to wider optima and better generalization,” arXiv preprint arXiv:1803.05407 (2018).