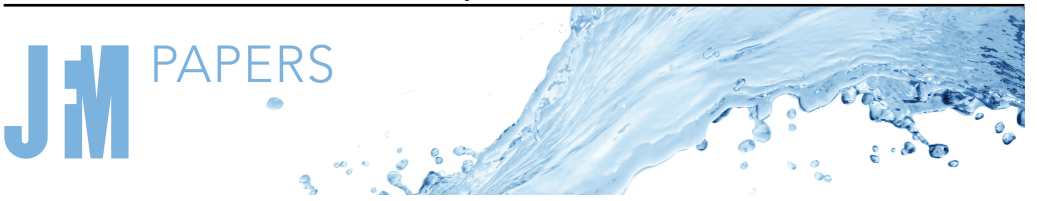
J. Fluid Mech. (2026), *vol.* 1041, A22, doi:10.1017/jfm.2026.11929



Subgrid stress modelling with multidimensional state space sequence models

Andy Wu[1] **and Sanjiva Lele**[1,2]

[1]Department of Aeronautics and Astronautics, Stanford University, Stanford, CA, USA

[2]Department of Mechanical Engineering, Stanford University, Stanford, CA, USA

Corresponding author: Andy Wu, awu1018@stanford.edu

(Received 14 November 2025 UTC; revised 10 June 2026 UTC; accepted 14 July 2026 UTC)

Large eddy simulations (LES) are becoming increasingly viable due to the growth in computational power over the last few decades, and subgrid stress modelling plays a large role in the accuracy of LES. A new class of neural network models, S4 and S4ND models, allow for learning a continuous representation of the discrete dataset, which facilitates a principled approach to incorporating grid dependence in neural network subgrid stress modelling. A S4ND U-Net neural network architecture is proposed and trained on both forced homogeneous isotropic turbulence (HIT) and channel flow. *A priori*, this architecture is able to extrapolate to grid spacings that are coarser than the training set grid spacings, while simpler artificial neural network (ANN) models fail. *A posteriori* tests on both forced HIT and channel flow indicate that the S4ND model is more accurate than traditional models and ANN-based models on grid sizes that are in the training set. The S4ND model is also able to extrapolate robustly to grid sizes that are coarser than the training set *a posteriori*, and is more accurate than many traditional subgrid stress models. Finally, the proposed model is also evaluated on flows at increasing Reynolds numbers, where it is seen that the proposed neural network remains stable even at a Reynolds number that is 500 000 times larger than what was seen in the training set.

Key words: turbulence modelling, machine learning

1. Introduction

Large eddy simulations (LES) provide an efficient paradigm to economically simulate turbulent flows and potentially flows of engineering interest by simulating the large-scale,

© The Author(s), 2026. Published by Cambridge University Press. This is an Open Access article, distributed under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives licence (https://creativecommons.org/licenses/by-nc-nd/4.0/), which permits non-commercial re-use, distribution, and reproduction in any medium, provided that no alterations are made and the original article is properly cited. The written permission of Cambridge University Press or the rights holder(s) must be obtained prior to any commercial use and/or adaptation of the article.

https://doi.org/10.1017/jfm.2026.11929 Published online by Cambridge University Press

predominantly energy containing turbulent eddies while modelling the small-scale turbulent structures (Goc *et al.* 2021). Thus, in order to close the governing equations, subgrid-scale (SGS) modelling, the act of modelling the effect of small-scale turbulent structures, is employed. SGS modelling has a significant impact on the accuracy and stability of LES (Pope 2000; Sagaut 2006). In general, SGS models can be split into two major categories, models being driven by statistical approximations inspired by turbulence theory, or derived from data (with additional constraints that may or may not be driven by turbulence theory). In each of these two major categories, one can additionally make the characterization of the model being functional, structural or a mix of both (Sagaut 2006; Prakash *et al.* 2022). Functional models aim to approximate the effect of smaller-scale structures of turbulence on the larger scales, contributing to the stability of the LES simulation (Sagaut 2006). Oftentimes, these models do not have high correlation with the subgrid stress tensor, τ_{ij}. Meanwhile, structural models aim to capture the subgrid stress tensor more accurately, but often do not capture the true energy dissipation associated with the small scales, and can be destabilizing to LES (Sagaut 2006).

Traditionally, SGS models are statistical approximations inspired by turbulence theory, whether they are functional or structural. One of the first SGS models for LES, the static Smagorinsky model (Smagorinsky 1963), assumes a perfect alignment between the resolved rate of strain tensor, $\bar{S}_{ij}$ and τ_{ij} (Pope 2000). Both the static Smagorinsky and dynamic Smagorinsky models (Germano *et al.* 1991) are based on inertial range energetics, where the rate of energy transfer from large-scale to small-scale structures is equivalent to the viscous dissipation at small scales. Other models, such as the sigma model (Nicoud *et al.* 2011), compute the eddy-viscosity coefficient differently as compared with the static and dynamic Smagorinsky models to incorporate more turbulence theory (subgrid stress must vanish in laminar flows), but are still rooted in the same theory of inertial range energetics (Vreman 2004). The sigma model represents a significant advance in traditional subgrid stress modelling, as by using the singular values of the resolved velocity gradient tensor, the model predicts zero subgrid stress if the resolved field is two-dimensional (2-D) or two-component, vanishes in laminar flow, and has cubic behaviour at the wall (Nicoud *et al.* 2011). Even then, all the models listed thus far have been functional models. A different subset of models, structural models, do not focus on the energy dissipation and instead are predicated on capturing higher correlation with the subgrid stress tensor. For example, Clark's gradient model (Clark *et al.* 1979) for the subgrid stress tensor is derived from a Taylor series expansion assuming an isotropic filter. The scale-similarity model (Bardina 1983; Berselli *et al.* 2006), assumes inertial range scale-similarity, suggesting that the subgrid stress tensor can be inferred from doubly filtered and filtered velocities. Since these models do not incorporate insights associated with turbulence energetics explicitly, often, they are destabilizing to a LES, even if the predicted subgrid stress tensor has higher correlation to the actual subgrid stress tensor (Beck & Kurz 2021; Prakash *et al.* 2022).

Recently, with the advance of deep learning and the availability of high quality direct numerical simulation (DNS) data online, learning a closure to the governing equations has become tractable instead of approaching subgrid stress modelling from a purely theoretical perspective (Beck *et al.* 2019; Xie *et al.* 2020*a*; Park & Choi 2021; Stoffer *et al.* 2021; Maejimam & Kawai 2024). Some of the early work done using machine learning involved using the velocity gradient and stresses to predict an eddy-viscosity coefficient (Sarghini *et al.* 2003). Models along this line of thought can be considered data-driven functional models. Another approach involves predicting the subgrid forcing term using neural networks, which directly approximates the term seen in the governing equations, or directly approximating τ_{ij} (Xie *et al.* 2020*a*,*b*). How τ_{ij} or the divergence of τ_{ij} is found

https://doi.org/10.1017/jfm.2026.11929 Published online by Cambridge University Press

varies, as Stolz *et al.* (2001) and Yuan *et al.* (2020) adopt a deconvolution approach, while others such as Xie *et al.* (2020*a*) use a tensor basis neural network (TBNN) with associated invariants as inputs (Wu & Lele 2025). Others use the velocity gradient tensor as an input to a neural network to predict τ_{ij} (Park & Choi 2021; Kang *et al.* 2023), or incorporate second derivatives into the neural network input (Wang *et al.* 2018; Pawar *et al.* 2020). Another approach is to use mixed models, where functional and structural models are mixed together to form one model, with the neural network learning the balance between the functional and structural models (Beck *et al.* 2019; Prakash *et al.* 2022). One can also adopt a reinforcement learning approach, where the neural network is given feedback at every 'simulation time step' and learns according to a reward function (Bae & Koumoutsakos 2022; Kurz *et al.* 2023). In the end, all these methods are just a way to either learn an eddy viscosity coefficient (functional), some part of the subgrid stress tensor (structural) or a mix of both.

Despite all this progress, neural network models are still not as flexible as traditional models, and the above works have all mainly focused on grid sizes that are relatively fine (only 4–8 times that of the DNS grid spacing, at most 16 times). Furthermore, it is well known that neural network models have trouble generalizing to grid spacings that are coarser than the training data (Kang *et al.* 2023; Cho *et al.* 2024). As such, Cho *et al.* (2024) adopt a recursive approach, where a neural network is continually augmented with higher Reynolds number (Re) data trained on LES integrated with a neural network subgrid stress model. This approach is able to generalize to a wide range of Reynolds numbers and grid spacing, and is one of the few investigations at a coarse LES grid spacing (when compared with the DNS grid spacing). Predominantly, neural network subgrid stress modelling only uses simple architectures such as artificial neural networks (ANNs) (Sarghini *et al.* 2003; Xie *et al.* 2020*a*,*b*; Yuan *et al.* 2020, 2021; Park & Choi 2021; Stoffer *et al.* 2021; Cheng *et al.* 2022; Kang *et al.* 2023; Sirignano & MacArt 2023; Cho *et al.* 2024) or simple convolutional neural networks (CNNs) (Beck *et al.* 2019; Guan *et al.* 2022). Given the advance of deep learning, it stands to reason that using more advanced neural network architectures could gain the generalization capability necessary for neural network models to be more flexible. As seen, Wu & Lele (2025) use a two neural network convolutional U-Net architecture trained on both channel flow and forced homogeneous isotropic turbulence (HIT), showing that it improves upon traditional models and machine learning models using more simple architectures at larger grid spacings (the LES grid spacings are 16 and 32 times the DNS grid spacing). However, generalization to coarser grid spacings than the training set is still lacking.

In deep learning, a new class of models that have been gaining traction involve learning continuous convolution kernels (Li *et al.* 2021; Nguyen *et al.* 2022). Since typical CNN architectures are discrete, they are not sensitized to the grid spacing. Therefore, it is not surprising that they are unable to generalize to coarser grid spacings for subgrid stress modelling. However, continuous convolution kernels explicitly account for the grid spacing, potentially allowing for generalization to coarser grid spacing than the training data. One class of models from the computer vision community, the S4ND model (Nguyen *et al.* 2022), allows for continuous convolution kernels to be learned by parameterizing each spatial dimension with a state space model (Gu *et al.* 2021). By using the S4ND model in place of traditional convolutions, while enforcing the subgrid stress tensor symmetries through a TBNN, a new neural network for subgrid stress modelling is created.

The main contributions of this paper include introducing a new neural network architecture that can extrapolate robustly to coarser grid spacings not seen during training without any additional retraining, first evaluated in an *a priori* setting. This paper also contributes *a posteriori* analysis of the neural network. After the neural network is trained

https://doi.org/10.1017/jfm.2026.11929 Published online by Cambridge University Press

on both forced HIT and channel flow data, it is integrated into LES simulations of both forced HIT and channel flow. The neural network is evaluated on a coarser grid spacing unseen during training while being more accurate than traditional and other data-driven SGS models. Further analysis also shows that the neural network model can extrapolate to *Re* up to 500 times the *Re* in the training set, while also extrapolating in grid spacing without the need to train on any additional LES data or undergo transfer learning. This neural network subgrid stress model is also able to remain stable even when the *Re* is over 500 000 times larger than what it was trained on.

2. Numerical and neural network details

This section provides background on the LES framework used, as well as the neural network model and training details.

2.1. *The LES governing equations*

The governing LES equations solved are the filtered incompressible Navier–Stokes equations. The filtering operation is defined as follows:

$$\bar{\chi}(\boldsymbol{x}, t) = \int G(\boldsymbol{r})\chi(\boldsymbol{x} - \boldsymbol{r}, t)\mathrm{d}r, \tag{2.1}$$

$$\int G(\boldsymbol{r})\mathrm{d}r = 1, \tag{2.2}$$

where χ is the flow variable to be filtered, $G(\boldsymbol{r})$ is the filtering kernel and $\bar{\chi}$ is the filtered flow variable. In all sections, the overbar operator is used to denote filtered quantities.

Now that the filter is defined, the governing equations are written below (in Einstein's summation notation)

$$\begin{gathered}\frac{\partial \bar{u}_i}{\partial x_i} = 0, \\ \frac{\partial \bar{u}_i}{\partial t} + \frac{\partial \bar{u}_i \bar{u}_j}{\partial x_j} = -\frac{1}{\rho}\frac{\partial \bar{p}}{\partial x_i} + \nu\frac{\partial^2 \bar{u}_i}{\partial x_j \partial x_j} - \frac{\partial \tau_{ij}}{\partial x_j}, \\ \tau_{ij} = \overline{u_i u_j} - \bar{u}_i \bar{u}_j,\end{gathered} \tag{2.3}$$

where $\overline{p}$ is the filtered pressure, ρ is the density, $\overline{u}$ is the filtered velocity and ν is the kinematic viscosity. The velocity field is solenoidal as seen from the continuity equation, and filtering the momentum equation results in obtaining the unclosed term, τ_{ij}. This tensor is computed by obtaining the filtered nonlinear product of the velocities before filtering, and then subtracting by the product of the filtered velocities. In LES only the filtered values of the dependent variables are available, thus, τ_{ij} cannot be computed and has to be modelled. In the present work, all simulations are in the incompressible regime, and the neural network will learn to approximate τ_{ij} by learning coefficients of a predefined tensor basis expansion.

2.2. *State space models and S4ND*

Recently, a new class of models, structured state space sequence (S4) models has emerged from the natural language processing community (Gu *et al.* 2021). For a one-dimensional (1-D) problem, the neural network attempts to learn a state space representation,

$$\begin{aligned}\dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}, \\ \boldsymbol{y} &= \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u},\end{aligned} \tag{2.4}$$

https://doi.org/10.1017/jfm.2026.11929 Published online by Cambridge University Press

where $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$, $\boldsymbol{D}$, are learnable matrices, $\boldsymbol{u}$ represents a 1-D input (function of time), $\boldsymbol{x}$ represents the state vector (dimension of this state vector is specified by the user), while $\boldsymbol{y}$ represents the output vector. In general, $\boldsymbol{D}$ is omitted as it can be viewed as a skip connection, which can be added in the neural network architecture (Gu *et al.* 2021). As seen, the neural network learns an underlying continuous representation from the discrete data. However, since the data is discrete, there needs to be a connection between the continuous and discrete setting. This connection is derived for the bilinear method (Gu *et al.* 2021). One can write the discretized state space as (the tilde operation signifies the discretized matrices)

$$\begin{aligned}
\boldsymbol{x}_k &= \tilde{\boldsymbol{A}}\boldsymbol{x}_{k-1} + \tilde{\boldsymbol{B}}\boldsymbol{u}_k, \\
\boldsymbol{y}_k &= \tilde{\boldsymbol{C}}\boldsymbol{x}_k, \\
\tilde{\boldsymbol{A}} &= \left(\boldsymbol{I} - \frac{\Delta}{2}\cdot\boldsymbol{A}\right)^{-1}\left(\boldsymbol{I} + \frac{\Delta}{2}\cdot\boldsymbol{A}\right), \\
\tilde{\boldsymbol{B}} &= \left(\boldsymbol{I} - \frac{\Delta}{2}\cdot\boldsymbol{A}\right)^{-1}\Delta\boldsymbol{B}, \\
\tilde{\boldsymbol{C}} &= \boldsymbol{C}.
\end{aligned} \tag{2.5}$$

As seen, when discrete values (such as during training) are needed since the training data is discrete, the state space can be discretized with a spacing parameter Δ and a discretization method such as the bilinear method or zero order hold (Gu *et al.* 2021). This ensures that the neural network, while learning from discrete data, has an underlying continuous representation, which is enabled by the additional knowledge of the spacing parameter Δ. The underlying continuous representation it learns can be found by writing the non-tilde matrices as a function of the tilde matrices. By learning this underlying continuous representation and then being able to discretize using a specified grid spacing, the S4 model has been found to generalize very well to different Δ not in the training set (Gu *et al.* 2021). This approach was extended to problems involving two dimensions in the computer vision community with the introduction of the S4ND model (Nguyen *et al.* 2022). The core idea behind the S4ND model, illustrated in figure 1, is to learn global, continuous convolution kernels by initializing a S4 model in each spatial dimension, and then taking the tensor outer product in order to obtain a continuous convolution kernel. Now, the Δ spacing parameter is the grid spacing. In other words, $\Delta = \Delta_g$ where Δ_g denotes the spatial spacing. While the grid spacing in each cardinal direction must be the same, this formulation allows for the grid spacing in various directions to be different (for example different constant grid spacing in x, y, z directions). When convolving the continuous convolution kernel with the discrete data, the spatial spacing is used to discretize the convolution kernel into a discrete convolution kernel before applying the convolution operation. Thus, the S4ND model can be used to replace traditional convolutions as a grid spacing aware operator, while traditional convolutions are agnostic to grid spacing. Overall, the S4ND model allows the neural network to learn continuous multidimensional representations of the data, which is more in line with fluid dynamics (continuum approximation), as the governing equations are continuous and then discretized onto a grid for LES.

In addition, since the state space is now discretized with a specific spatial spacing Δ_g, S4ND introduces a novel idea called bandlimiting (Nguyen *et al.* 2022), which seeks to remove spurious frequencies in the convolution kernel that cannot be supported by the grid spacing implied by Δ_g. If $\boldsymbol{A}$ is diagonal, then the state space convolution kernel

https://doi.org/10.1017/jfm.2026.11929 Published online by Cambridge University Press

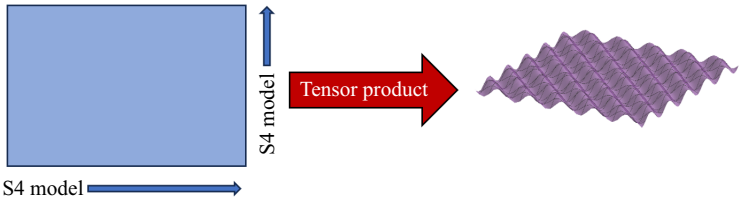


Figure 1. The S4ND extension of the S4 model. A S4 model is instantiated in each direction, and then an outer product is taken to produce a continuous convolution kernel. When training and during inference, the continuous convolution kernel is discretized.

'basis' can be represented by $e^{a_n t} B_n$, where a_n denotes the nth diagonal element of $\boldsymbol{A}$. Therefore, a_n is a parameter that controls the frequency of the basis function associated with the convolution kernel. In order to mask out spurious frequencies in the convolution kernel, S4ND introduces a bandlimiting parameter, α, which is a hyperparameter. If $a_n \Delta_g > (1/2)\alpha$, then this convolution kernel basis function is masked out, as this means that the frequency of the basis function is larger than what the grid can support (it is spurious). When $\alpha = 1$, bandlimiting corresponds to masking out everything above the Nyquist wavenumber. However, due to the finite state representation and also the decay associated with the real part of a_n, α is set lower empirically (between 0.1 and 0.6), which creates smoother convolution kernels (Nguyen *et al.* 2022). In this work, $\alpha = 0.1$ is used. Appendix A provides a justification for using this specific bandlimit parameter.

These mechanisms make S4ND more suitable for subgrid stress modelling in LES. First, the idea of bandlimiting functions as a rigorous filter for aliasing. In LES, the grid spacing Δ_g functions as a low-pass filter, where beyond the Nyquist wavenumber, $k_c = \pi/\Delta_g$, the grid can no longer support higher wavenumber oscillations. Standard discrete CNNs learn fixed weights that ignore this limitation set by the grid. Therefore, after training on a set of resolutions, when applied to coarser grids, they may predict high-frequency oscillations that alias back into the resolved scales, which is seen as a failure to extrapolate to larger grid spacings. In contrast, S4ND's bandlimiting framework ensures that the continuous convolution kernel (due to the state space representation) suppresses the frequencies that exceed the spatial grid's spectral resolution, enforcing the consistency between the data-driven subgrid stress model and the resolution of the mesh.

Specifically, this is directly connected to the notion of LES filter consistency. In LES, the subgrid stress tensor must be mathematically consistent with the implied filter that is a consequence of both the grid width and the numerical method. Since standard discrete convolutions learn fixed, grid-dependent weights that treat the LES simulation as an array of pixels, they have no inductive biases to enforce this consistency outside of the training set grid resolutions. Thus, the learned closure is oftentimes only valid for filter width ranges seen during training. Meanwhile, S4ND's continuous formulation with bandlimiting provides an inductive bias, ensuring that the subgrid stress model adapts its spectral content as the filter width changes, even at untrained grid resolutions. This ensures that the S4ND model can robustly produce a more filter-consistent closure even on grid resolutions outside of the training set, instead of injecting unphysical frequency content at untrained resolutions. For S4ND, this comes as part of the architectural framework instead of only as a learned consequence of the data.

Furthermore, S4ND allows for a continuous representation instead of a discrete representation, which allows it to better learn scale similarity. By learning the underlying subgrid stress field as a continuous quantity, it allows the neural network model to

https://doi.org/10.1017/jfm.2026.11929 Published online by Cambridge University Press

extrapolate physically consistent predictions to coarser grid resolutions where scale-invariance is important but where discrete correlations learned by a CNN would break down.

2.3. *Neural network architecture*

The neural network involves replacing discrete convolutions with the S4ND block in a U-Net architecture, followed by a backend TBNN.

2.3.1. *Tensor basis neural network*

The TBNN, originally introduced by Ling *et al.* (2016), involves computing the invariants and the tensor basis assuming that the subgrid stress tensor is a function of the resolved rate of strain tensor, $\bar{\boldsymbol{S}}_{ij}$, and the resolved rotation rate tensor, $\bar{\boldsymbol{R}}_{ij}$. A detailed derivation of the tensor basis and invariants for the subgrid stress tensor can be found in works such as Stallcup *et al.* (2022) and Wu & Lele (2025). Below, the invariants and tensor basis are listed for reference.

If one assumes that

$$\tau_{ij} = F(\bar{\boldsymbol{S}}_{ij}, \bar{\boldsymbol{R}}_{ij}), \tag{2.6}$$

$$\bar{\boldsymbol{S}} \equiv \bar{\boldsymbol{S}}_{ij} = \frac{1}{2}\left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i}\right), \tag{2.7}$$

$$\bar{\boldsymbol{R}} \equiv \bar{\boldsymbol{R}}_{ij} = \frac{1}{2}\left(\frac{\partial \bar{u}_i}{\partial x_j} - \frac{\partial \bar{u}_j}{\partial x_i}\right), \tag{2.8}$$

then the invariants can be written as

$$\left\{tr\left(\bar{\boldsymbol{S}}\right), tr\left(\bar{\boldsymbol{S}}^{\boldsymbol{2}}\right), tr\left(\bar{\boldsymbol{S}}^{\boldsymbol{3}}\right), tr\left(\bar{\boldsymbol{R}}^{\boldsymbol{2}}\right), tr\left(\bar{\boldsymbol{S}}\bar{\boldsymbol{R}}^{\boldsymbol{2}}\right), tr\left(\bar{\boldsymbol{S}}^{\boldsymbol{2}}\bar{\boldsymbol{R}}^{\boldsymbol{2}}\right), tr\left(\bar{\boldsymbol{S}}^{\boldsymbol{2}}\bar{\boldsymbol{R}}^{\boldsymbol{2}}\bar{\boldsymbol{S}}\bar{\boldsymbol{R}}\right)\right\}, \tag{2.9}$$

and the tensor basis can be written as

$$\left\{\boldsymbol{I}, \bar{\boldsymbol{S}}, \bar{\boldsymbol{S}}^{\boldsymbol{2}}, \bar{\boldsymbol{R}}^{\boldsymbol{2}}, \bar{\boldsymbol{S}}\bar{\boldsymbol{R}} - \bar{\boldsymbol{R}}\bar{\boldsymbol{S}}, \bar{\boldsymbol{R}}\bar{\boldsymbol{S}}\bar{\boldsymbol{R}}, \bar{\boldsymbol{S}}^{\boldsymbol{2}}\bar{\boldsymbol{R}} - \bar{\boldsymbol{R}}\bar{\boldsymbol{S}}^{\boldsymbol{2}}, \bar{\boldsymbol{R}}\bar{\boldsymbol{S}}\bar{\boldsymbol{R}}^{\boldsymbol{2}} - \bar{\boldsymbol{R}}^{\boldsymbol{2}}\bar{\boldsymbol{S}}\bar{\boldsymbol{R}}\right\}. \tag{2.10}$$

As seen, the invariants involve extremely high powers of the resolved strain rate and rotation rate tensors, which are not ideal due to aliasing errors, which increase with increasing power (Roberts & Mullis 1987). Thus, instead of using the invariants, which are mathematically complete, one can simply use the velocity gradient tensor in place of the invariants, which is an input commonly used for neural network subgrid stress models (Park & Choi 2021; Kang *et al.* 2023; Cho *et al.* 2024). It is noted that there is no performance degradation using the velocity gradient tensor instead of the invariants as inputs to the neural network *a priori*. Even if the input to the neural network is changed, the backend of the neural network remains the same. The neural network will predict coefficients of the tensor basis expansion, listed here as $c_1 - c_8$ (eight scalar values are predicted given the filtered velocity gradient tensor as an input)

$$\begin{aligned}\tau_{ij} = {} & c_1\boldsymbol{I} + c_2\bar{\boldsymbol{S}} + c_3\bar{\boldsymbol{S}}^{\boldsymbol{2}} + c_4\bar{\boldsymbol{R}}^{\boldsymbol{2}} + c_5\left(\bar{\boldsymbol{S}}\bar{\boldsymbol{R}} - \bar{\boldsymbol{R}}\bar{\boldsymbol{S}}\right) \\ & + c_6\bar{\boldsymbol{R}}\bar{\boldsymbol{S}}\bar{\boldsymbol{R}} + c_7\left(\bar{\boldsymbol{S}}^{\boldsymbol{2}}\bar{\boldsymbol{R}} - \bar{\boldsymbol{R}}\bar{\boldsymbol{S}}^{\boldsymbol{2}}\right) + c_8\left(\bar{\boldsymbol{R}}\bar{\boldsymbol{S}}\bar{\boldsymbol{R}}^{\boldsymbol{2}} - \bar{\boldsymbol{R}}^{\boldsymbol{2}}\bar{\boldsymbol{S}}\bar{\boldsymbol{R}}\right).\end{aligned} \tag{2.11}$$

2.3.2. *Overall architecture*

Now that the backend TBNN is introduced, the overall neural network architecture can be seen in figure 2.

https://doi.org/10.1017/jfm.2026.11929 Published online by Cambridge University Press

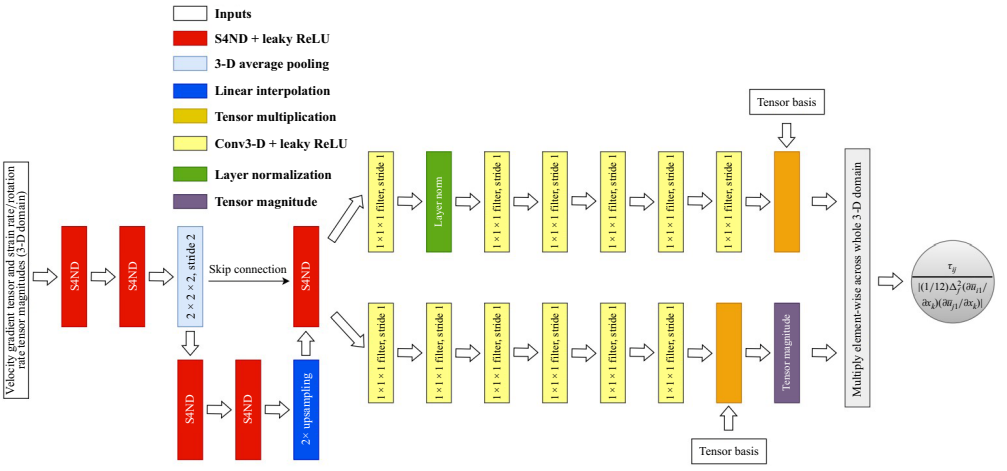


Figure 2. The overall neural network architecture. The whole three-dimensional (3-D) domain is inputted into the neural network, and the output is the corresponding subgrid stress at each point. For each S4ND layer, a state space dimension of eight was chosen to keep the trainable parameters comparable to existing models in the literature. The average pooling operation will downsample all spatial dimensions by a factor of two. As seen, after the final S4ND layer, the neural network architecture splits into two 'prediction heads' to predict the 'magnitude' and 'structure' of the subgrid stress tensor, and both prediction heads use the tensor basis (each prediction head predicts its own set of coefficients to multiply the tensor basis components). See Wu (2025) for downloadable version of the model and an example input/output.

The inputs to the neural network are the 3-D, spatially varying velocity gradient tensor, as well as the strain rate and rotation rate tensor magnitudes. These inputs are passed into a U-Net S4ND architecture that allows for multiscale fusion of different spatial scales of turbulence. After the U-Net architecture, the neural network splits into two streams. Wu & Lele (2025) found that using two neural networks, one to predict the magnitude of τ_{ij} while the other predicts the structure (normalized) τ_{ij}, results in better performance. This neural network architecture retains the idea of predicting a 'structure' component and a 'magnitude' component, but is able to do it in one single neural network instead of splitting the task of predicting a 'structure' or 'magnitude' component to separate neural networks. As seen, the neural network splits into two paths after the U-Net architecture, with both paths using the TBNN (predicting eight coefficients that multiply the tensor basis). However, instead of using fully connected layers, which are unable to handle different domain sizes, $1 \times 1 \times 1$ convolutions are used, which can emulate fully connected layers while still being able to generalize to various domain sizes.

Non-dimensionalizing the model and scaling the inputs and outputs are crucial to training a neural network model. Here, the inputs (velocity gradient tensor, strain rate and rotation rate tensor magnitudes, so a total of 11 quantities at each point in the domain) are normalized with a max–min scaling,

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} 2 - 1 \tag{2.12}$$

where x corresponds to every component of the velocity gradient tensor (or the strain rate/rotation rate tensor magnitudes). The maximum and minimum correspond to the 3-D domain's maximum and minimum. This input normalization ensures that the neural network model inputs are always scaled between $[-1, 1]$. The tensor basis are also scaled,

https://doi.org/10.1017/jfm.2026.11929 Published online by Cambridge University Press

although with the Frobenius norm instead,

$$W_{ij}^{norm} = \frac{W_{ij}}{|W_{ij}|}, \tag{2.13}$$

where W_{ij} is a tensor corresponding to a tensor in the tensor basis expansion. Note that in (2.13), repeated indices do not imply summation. Furthermore, the output is also normalized accordingly to ensure that the model is non-dimensional. Following observations made by Kang *et al.* (2023), the Clark's gradient model term is taken as a normalization for τ_{ij}, as seen in figure 2. However, one small modification is made. Work done by Scotti *et al.* (1993) introduced the anisotropic mesh length factor, f, which multiplies the effective grid width Δ_e as grid anisotropy may play a role in determining τ_{ij}. Therefore, the normalization used for τ_{ij} becomes

$$\frac{\tau_{ij}}{\left|\frac{1}{12}\Delta_f^2 \frac{\partial \bar{u}_l}{\partial x_k}\frac{\partial \bar{u}_m}{\partial x_k}\right|} \tag{2.14}$$

where $\Delta_f = f\Delta_e$, $f = 1 + (2/27)[\ln(a_1)^2 - \ln(a_1)\ln(a_2) + \ln(a_2)^2]$ and $\Delta_e = (\Delta_1\Delta_2\Delta_3)^{1/3}$. Here Δ_i denotes the grid spacing in the ith dimension, while a_1 and a_2 denote the grid spacing ratios (both less than 1).

2.4. *Neural network training details*

This section goes over the process of querying the training data, as well as introducing the loss function and some metrics that can be used to quantify the performance of the neural networks in a LES *a posteriori*.

2.4.1. *Training data*

Both filtered HIT and channel flow data are queried from the Johns Hopkins Turbulence Database (JHTDB), a service that provides high-quality DNS data for research purposes (Perlman *et al.* 2007; Li *et al.* 2008; Graham *et al.* 2016). From Park & Choi (2021), training on two different filter widths provides the neural network with more generalizability to different grid spacings. Thus, since this paper is focused on applications to aggressive, coarse grid spacing settings, after querying the DNS data, the DNS data is filtered to a filter width corresponding to $\Delta_g = 16\Delta_{DNS}$ and $\Delta_g = 32\Delta_{DNS}$, where Δ_{DNS} denotes the DNS grid spacing. The filtering operation involves using a box filter in physical space. Following Wu & Lele (2025), two different domain sizes are also queried in order to help the neural network generalize to various domain sizes. The querying procedure used here is identical to that used in Wu & Lele (2025), and is also listed here for completeness.

The training data is split into a 80 : 10 : 10 split, where 80 % of the total training data is used for training the neural network, 10 % of the data is used as a validation set during training for early stopping and 10 % of the data is used as the hold-out test set in the *a priori* testing section. To produce the training data for forced HIT, five snapshots are queried from the JHTDB database. To assemble the channel flow training data, four snapshots are queried. Each snapshot is queried at equal time intervals that are statically decorrelated from each other (time intervals different for HIT and channel flow). The DNS HIT resolution queried is 1024^3, with a Taylor Reynolds number (Re_λ) of 433 and a domain size of $[2\pi, 2\pi, 2\pi]$. The channel flow resolution queried is $2048 \times 512 \times 1536$, with a friction Reynolds number (Re_τ) of 1000 and a domain size of $[8\pi, 2, 3\pi]$ when non-dimensionalized by the channel half-height. After the training data is queried, it is filtered. For example, filtering at a $16\Delta_{DNS}$ filter width would result in the filtered HIT

https://doi.org/10.1017/jfm.2026.11929 Published online by Cambridge University Press

Number of realizations	Filter width	Domain size	Flow type
320	$16\Delta_{DNS}$	16^3	HIT
320	$32\Delta_{DNS}$	8^3	HIT
40	$16\Delta_{DNS}$	32^3	HIT
40	$32\Delta_{DNS}$	16^3	HIT
320	$16\Delta_{DNS}$	16^3	Channel
320	$32\Delta_{DNS}$	8^3	Channel
40	$16\Delta_{DNS}$	32^3	Channel
40	$32\Delta_{DNS}$	16^3	Channel

Table 1. Training data.

resolution being 64^3 and the filtered channel flow resolution being $128 \times 32 \times 96$. Then, the full domains are split into 'subdomains' of sizes 16^3 and 32^3. This results in the total training data content in table 1.

As stated before, the training data is divided into 80 per cent for training, 10 per cent for validation and 10 per cent as a held-out test set. Each of the categories of training data is split according to the percentages above and then recombined into the total training, total validation, and total test set. In addition, filter width $\Delta_g = 24\Delta_{DNS}$ and filter width $\Delta_g = 64\Delta_{DNS}$ data are also queried, but are not included in the training set and are instead held out as part of the test set to test the neural network generalization capabilities.

2.4.2. *Loss function and metrics*

The loss function used for training follows from the loss function introduced in Wu & Lele (2025), involving a two-term loss function that enforces both the root mean squared error (RMSE) between the predicted and actual subgrid stress tensor, as well as the RMSE between the predicted and actual interscale energy transfer field, $\epsilon = -\tau_{ij}\bar{S}_{ij}$. As such, the loss function involves

$$L = \boldsymbol{RMSE}(\tau_{ij,pred} - \tau_{ij})/\boldsymbol{RMS}(\tau_{ij}) + \lambda \boldsymbol{RMSE}(\epsilon_{pred} - \epsilon)/\boldsymbol{RMS}(\epsilon), \tag{2.15}$$

where λ is a constant that can bias the neural network towards emphasizing the energy dissipation or the predicted values of τ_{ij}. In the present work, following Wu & Lele (2025), $\lambda = 1$ is used, which indicates an equal bias between both loss function terms. The first term of this loss function is termed 'RMSE' (unbolded) and the second term of this loss function is termed 'DRMSE' (unbolded). Meanwhile, the bolded terms signify the operation of taking the RMSE or RMS of the particular quantity in the parenthesis. During training, the neural networks are trained with a batch size of eight, and the denominators of the loss function are computed per batch. As seen, this loss function directly enforces that the predicted and actual subgrid stress tensors are close, while simultaneously ensuring that the predicted and actual interscale energy transfer fields are also close. This ensures that the overall energy dissipation is accurate, meaning that the subgrid stress model is dissipating the correct energy from the resolved field.

For *a posteriori* tests, both qualitative and quantitative evaluations are important. However, quantitative metrics are often case dependent. Therefore, a few metrics are introduced that can be used to compare the various subgrid stress models with the filtered DNS solution that work for both forced HIT and channel flow. Note that these are necessary but not sufficient metrics to evaluating a subgrid stress model's quality. The first metric involves computing a sum-absolute error (SAE) between the filtered DNS spectra and the

https://doi.org/10.1017/jfm.2026.11929 Published online by Cambridge University Press

LES spectra,

$$SAE_E = \int_k \left|\left(\log(E_{DNS}(k)) - \log(E_{LES}(k))\right)\right| \mathrm{d}k \approx \sum_k \left|\log\left(E_{DNS_k}\right) - \log\left(E_{LES_k}\right)\right|, \tag{2.16}$$

where the energy spectra is summed over a specific wavenumber band, with k denoting the wavenumber. The logarithm ensures that differences in the spectra at large wavenumbers can still meaningfully increase this metric. Another metric involves computing the SAE between the filtered DNS and LES correlations (spatial or temporal),

$$SAE_{corr} = \int_\delta \left|C_{DNS}(\delta) - C_{LES}(\delta)\right| \mathrm{d}\delta \approx \sum_\delta \left|C_{DNS_\delta} - C_{LES_\delta}\right|, \tag{2.17}$$

where C denotes the temporal or spatial correlation and δ represents the spacing (temporal or spatial) used to calculate the value of the correlation function. For channel flow, additional metrics involving the Reynolds stress profiles can also be developed. One involves computing the SAE over the various Reynolds stress profiles summed in the wall-normal direction,

$$SAE_{R_{xx}} = \int_y \left|R_{xx_{DNS}}(y) - R_{xx_{LES}}(y)\right| \mathrm{d}y \approx \sum_y \left|R_{xx_{DNS_y}} - R_{xx_{LES_y}}\right|, \tag{2.18}$$

where R_{xx} represents the Reynolds stress over the 'xx' direction (i.e. 11, 22, 33), while y denotes the wall-normal direction. Another quantitative metric for channel flow involves computing the SAE for the mean velocity profiles, summed over the wall-normal direction,

$$SAE_u = \int_y \left|\overline{u(y)}_{DNS} - \overline{u(y)}_{LES}\right| \mathrm{d}y \approx \sum_y \left|\overline{u(y)}_{DNS} - \overline{u(y)}_{LES}\right|, \tag{2.19}$$

where y once again is the wall-normal direction and $\overline{u(y)}$ is the LES or filtered DNS mean velocity profile (averaged over the homogenous directions, in this case x and z). The above four quantitative metrics do not represent a complete characterization of turbulence, but serve as 'first-order' metrics in order to compare various subgrid stress models quantitatively in an *a posteriori* setting.

3. *A priori* results

Here, neural network training hyperparameters are covered, along with information on various other neural network architectures before discussing *a priori* results.

3.1. *Neural network training details*

All neural networks are trained using the Adam optimizer using three learning rates of 0.01, 0.001 and 0.0001, and the best performing model is selected. We note that in training all the neural network models, each neural network performed the best using a learning rate of 0.001. In order to facilitate comparisons between the neural network proposed and other neural network models in the literature, a TBNN using an ANN frontend was also trained (labelled as 'TBNN'), as well as a two neural network convolutional U-Net architecture introduced by Wu & Lele (2025), which is labelled as 'NN'. The 'TBNN', which uses an ANN as a frontend, has around 13 200 trainable parameters and uses normalized invariants and a normalized tensor basis. Meanwhile, the 'NN' is a CNN architecture following Wu & Lele (2025), and also uses normalized invariants and a normalized tensor basis

https://doi.org/10.1017/jfm.2026.11929 Published online by Cambridge University Press

NN model	Inputs	Trainable parameters
S4ND	$\boldsymbol{V}_{norm}$, $\|\boldsymbol{S}\|_{norm}$, $\|\boldsymbol{R}\|_{norm}$	9256
NN-V	$\boldsymbol{V}_{norm}$, $\|\boldsymbol{S}\|_{norm}$, $\|\boldsymbol{R}\|_{norm}$	18 526
NN	Normalized invariants and Normalized tensor basis	12 523
TBNN	Normalized invariants and Normalized tensor basis	13 210

Table 2. Various neural network model inputs and trainable parameters.

Δ_g	Average energy filtered	Maximum observed energy filtered
$1\Delta_{DNS}$	0 %	0 %
$2\Delta_{DNS}$	1 %	2.1 %
$4\Delta_{DNS}$	2 %	5.3 %
$8\Delta_{DNS}$	4 %	12.2 %
$16\Delta_{DNS}$	9.6 %	23.4 %
$32\Delta_{DNS}$	14.3 %	40.4 %
$64\Delta_{DNS}$	22.6 %	56.4 %

Table 3. Filter widths and energy filtered for HIT.

as inputs. The S4ND model was trained with the bandlimit parameter set to 0.1. This neural network has inputs that are simplified from the invariants, and is seen in table 2. Appendix A provides a more detailed hyperparameter sweep of the bandlimit parameter, which provides justification for setting the bandlimit parameter to 0.1. Table 2 also shows the total number of trainable parameters for each neural network model. As seen, all neural networks are roughly the same size (orders of magnitude for the parameters are similar), allowing for a fair comparison between the various neural network models. To isolate the architectural effects of the S4ND model, a second CNN, mirroring the exact same architecture as the S4ND model with only the S4ND layers replaced with convolutions is also trained, which is labelled as NN-V. This neural network is also trained on the exact same inputs and outputs and trained the same way as the S4ND model. Here NN-V is evaluated *a posteriori* and will be compared with all the other models to isolate the architectural effects of S4ND *a posteriori*. For all neural networks, early stopping was employed to ensure that the neural networks do not overfit, with the criteria being that if the validation loss did not decrease for 300 epochs, the neural network weights would be restored to the instance of the lowest validation loss and training would terminate.

3.2. *HIT filtering*

The *a priori* results will be demonstrated on forced HIT. To gain intuition on the amount of filtering introduced by different filter widths, table 3 illustrates the amount of energy filtered as a function of filter width.

The values are computed by evaluating the average turbulent kinetic energy filtered out as compared with the reference DNS simulation over multiple ensembles, as well as the maximum turbulent kinetic energy filtered out that was observed over the ensembles. As seen, for small filter widths, only a small amount of energy has been filtered out as compared with the DNS simulation. Therefore, the subgrid stress models do not need to model much of the physics, as most of the flow has been resolved. However, from a filter width of 16 Δ_{DNS} onward, the energy filtered out becomes increasingly significant. In these

https://doi.org/10.1017/jfm.2026.11929 Published online by Cambridge University Press

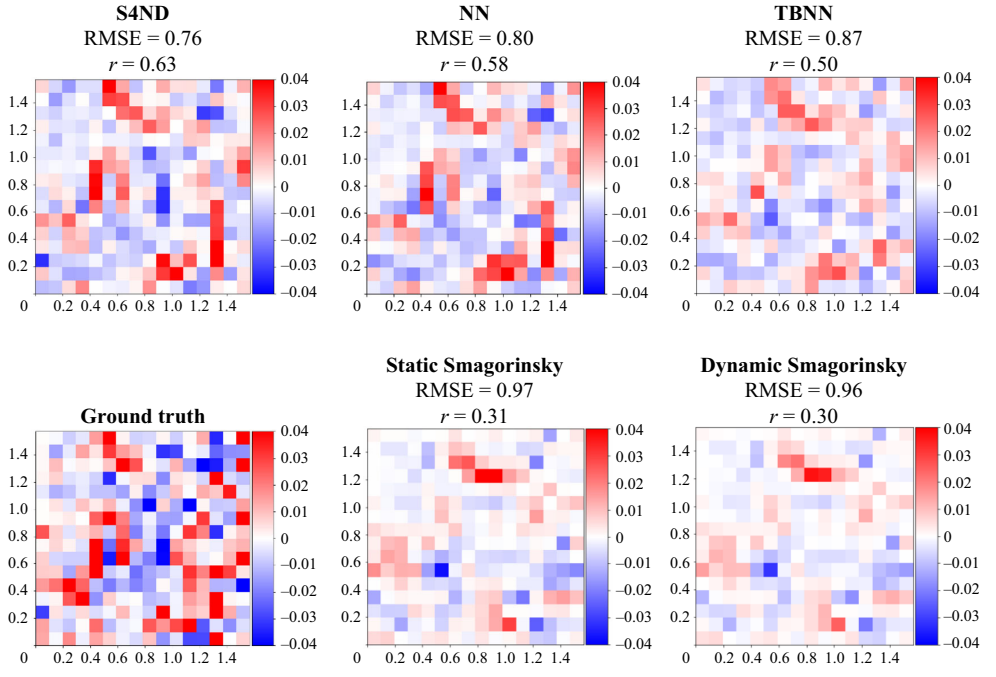


Figure 3. Spatial distribution of τ_{11} for a 2-D slice of the 3-D domain for a filter width of 16 Δ_{DNS} for the forced HIT case. All numerical values are ensemble averaged over 10 ensembles, while the spatial fields are chosen randomly for visual reference. The letter r denotes the correlation coefficient. The correlation coefficients are highest for S4ND, while the other, less complex neural networks have lower correlation coefficients. The x and y axis correspond to the coordinates in the physical domain.

situations, the subgrid stress models contribute directly to the accuracy of the simulation, and inaccuracies in the subgrid stress model have a larger effect on the resolved flow. Therefore, the *a priori* results are evaluated on filter widths 16 Δ_{DNS} or larger.

3.3. *A priori results*

For *a priori* tests, the S4ND model is compared with the TBNN, NN model, static Smagorinsky and dynamic Smagorinsky models. As defined in the previous sections, the TBNN model refers to a TBNN with an ANN frontend, while the NN model refers to the CNN-based model. First, the spatial distribution of the subgrid stress can be compared for a filter width of 16 Δ_{DNS}, which is shown in figure 3.

From figure 3, all the neural network models are far better than the Smagorinsky models in capturing the overall correlation with the filtered DNS subgrid stress, as evidenced by both the higher correlation coefficient and the lower RMSE. The trend among neural networks is also of note, as the simplest neural network (i.e. TBNN) has the worse RMSE and correlation coefficient out of all the neural network models, yet it has the most trainable parameters. Meanwhile, the more complex neural network models (such as the NN model and the S4ND model) perform far better. Due to the coarse-grid regime that the models are evaluated in, the traditional models, as well as the TBNN model perform poorly, and the more complex models show improvement on top of these models. However, even the more advanced models are not perfect when compared with the ground truth, which is expected in the coarse-grid regime, where a significant portion of the energy has

https://doi.org/10.1017/jfm.2026.11929 Published online by Cambridge University Press

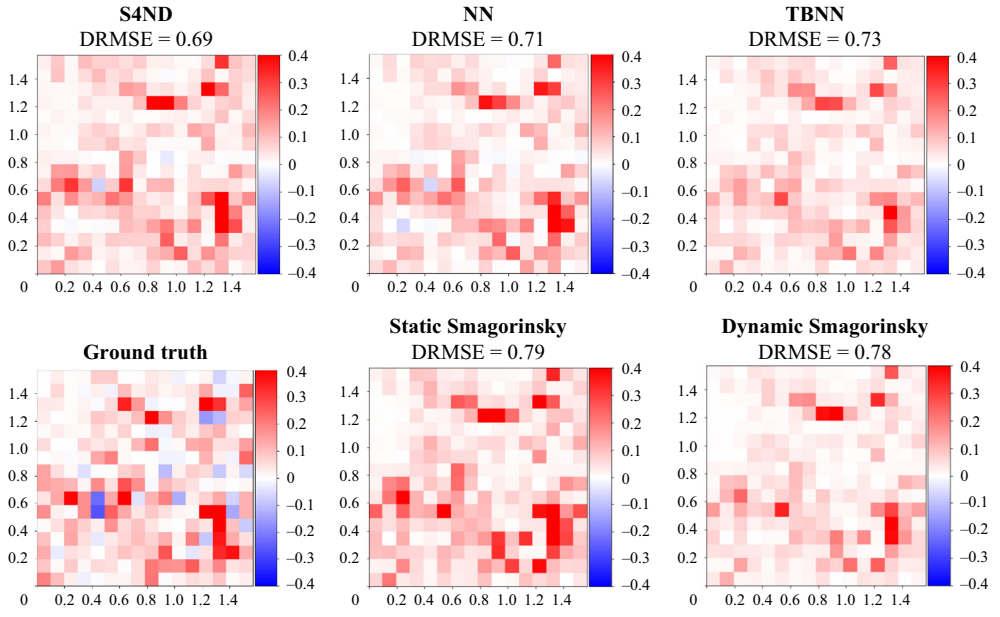


Figure 4. Spatial distribution of $-\tau_{ij}\bar{S}_{ij}$ for a 2-D slice of the 3-D domain for a filter width of 16 Δ_{DNS}. All numerical values are ensemble averaged over 10 ensembles, while the spatial fields are chosen randomly for visual reference. The x and y axis correspond to the coordinates in the physical domain.

been filtered out (Wu & Lele 2025). In addition, the interscale energy transfer field, $-\tau_{ij}\bar{S}_{ij}$ is also shown.

Overall, as seen from figures 3 and 4, visually, the TBNN field is more diffuse as compared with the S4ND and NN fields, but they are quite similar. In fact, all the interscale energy transfer fields look relatively similar visually, and the DRMSE are not too far off from one another. This suggests that the neural network models are able to represent the energetics of turbulence faithfully *a priori*, and therefore have a good chance of being stable *a posteriori*, as the interscale energy transfer field is similar to the static and dynamic Smagorinsky models, which are known to be stabilizing towards a LES simulation. Despite the spatial distribution of $-\tau_{ij}\bar{S}_{ij}$ looking visually similar to the Smagorinsky models, the neural network models are still able to predict actual subgrid stress tensor spatial distributions that are more in line with the ground truth seen in figure 3. This suggests that the neural network model performance *a priori*, when coupled with the two-term loss function, are not behaving like 'structural models', which oftentimes sacrifice stability (prediction of energy dissipation) for higher correlation with the subgrid stress tensor.

In addition, probability density functions (PDFs) showing the overall distribution associated with various models are shown in figures 5(*a*) and 5(*b*). From figure 4, the overall spatial distribution of $-\tau_{ij}\bar{S}_{ij}$ is similar across most models, and this is corroborated when evaluating the PDF of ϵ. From figure 5(*a*), one can see that all the models do a good job representing the forward scatter (energy transfer to smaller scales) while they do worse on the backscatter (energy transfer from small to large scales). However, both the NN and S4ND models have more backscatter than the traditional models, which barely even have any values smaller than zero. From figure 5(*b*), the overall conclusions drawn from evaluating the spatial distribution are confirmed. Overall, the S4ND model is the closest to

https://doi.org/10.1017/jfm.2026.11929 Published online by Cambridge University Press

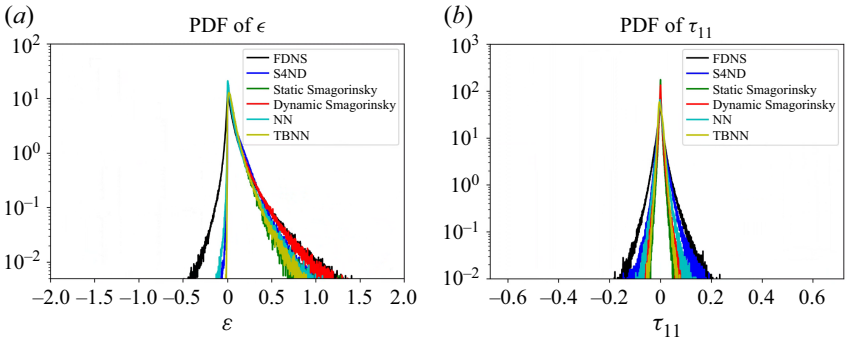


Figure 5. The PDFs for 16 Δ_{DNS} forced HIT. As seen, the S4ND model is more able to accurately capture the distribution of τ_{11} as compared with other models. The S4ND model and NN model (convolution-based) both capture more backscatter than traditional models, although it is still less than filtered DNS (FDNS). Here (*a*) PDF of ϵ; (*b*) PDF of τ_{11}.

the filtered DNS PDF, followed by the CNN-based neural network model (NN), and then followed by the TBNN and the traditional models. Note that in the aggressive filtering regime, the TBNN behaves quite poorly, oftentimes being no better than the dynamic Smagorinsky model when comparing PDFs. As seen from others in the literature, in general, simple ANN-based neural network models can still perform well when compared with traditional models when the filtering is less aggressive (Xie *et al.* 2020*b*; Cho *et al.* 2024).

From all these results, while adding complexity to the neural network model can help increase performance, as seen since the S4ND and NN models have better performance as compared with the TBNN model, in a setting where the filter width is the same as the training data, it is hard to distinguish between whether the S4ND or NN model is better. This is to be expected, as the continuous convolution kernels that S4ND uses should perform better in extrapolative settings, not necessarily in situations that are part of the training data.

3.4. *Interpolative and extrapolative a priori results*

While the performance when the filter width is the same as the training set is good, one also needs to evaluate the neural network on different filter widths. First, a less stringent test is applied, where the neural network models, trained on $16\Delta_{DNS}$ and $32\Delta_{DNS}$ data, is evaluated on a filter width of $24\Delta_{DNS}$.

From figure 6, the overall trends hold, where the ANN-based model (i.e. TBNN) is the worst-performing, followed then by the NN and then the S4ND models for both RMSE and DRMSE. Overall, all the neural network models are able to interpolate to this 'inbetween' filter width, since the training data consists of $16\Delta_{DNS}$ and $32\Delta_{DNS}$ data. A more challenging case is evaluating the neural network models on $64\Delta_{DNS}$ data, as even more energy is filtered out at this aggressive filter width, meaning that the neural network models must learn scale-similarity in order to extrapolate to this more aggressive filter width. These results are shown in figure 7.

Figure 7 shows that the only neural network model that can successfully extrapolate to this filter width is the S4ND model. The TBNN model does extremely poorly, as the RMSE loss is now multiple times higher than the test set loss at an in-training set filter width, completely failing to generalize. Even the NN model is unable to generalize, as the loss is around twice as high as compared with the in-training set filter width loss.

https://doi.org/10.1017/jfm.2026.11929 Published online by Cambridge University Press

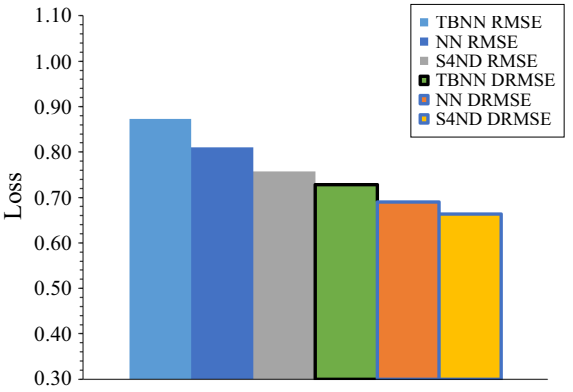


Figure 6. The RMSE and DRMSE for various neural network models evaluated on $24\Delta_{DNS}$ HIT data. As seen, the S4ND is able to handle this case the best (no increase in the RMSE and DRMSE as compared with the test set at other filter widths), followed by the NN model and then the traditional TBNN, which shows a noticeable increase in both the RMSE and DRMSE when compared with the S4ND model.

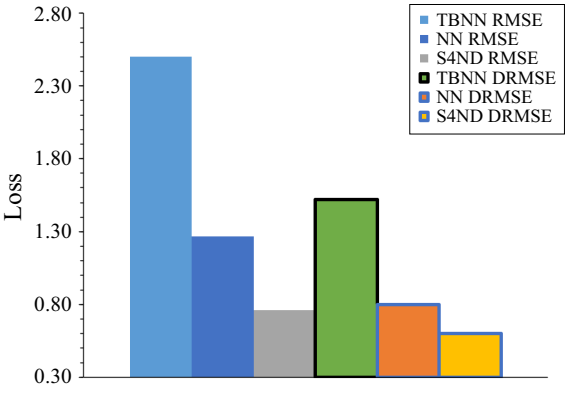


Figure 7. The RMSE and DRMSE for various neural network models evaluated on $64\Delta_{DNS}$ HIT data. As seen, the S4ND model is able to extrapolate to this more aggressive filter width without any significant increase in the RMSE and DRMSE, while the NN model has around a factor of two increase in the RMSE and a factor of 1.3 in DRMSE. Meanwhile, the traditional TBNN has a loss increase of almost a factor of four for RMSE and a loss increase of a factor of 2.5 for DRMSE.

Meanwhile, the S4ND model shows no increase in the loss as compared with filter widths that are within the training set, being the only model that can extrapolate successfully to a setting where the filter width is more aggressive than the training set it was trained on. Referring to table 3, this extrapolative case corresponds to a setting where on average, 22.6 per cent of the energy has been filtered out as compared with DNS. In particular, the DRMSE is an important quantity. While the RMSE is important, as it is an indirect way to ensure that the momentum equation is closed properly (since the divergence of the subgrid stress tensor is taken in the momentum equation), if the energy equation for the flow were written, the term $\tau_{ij}S_{ij}$ arises directly in the equation. The DRMSE is a direct measure of how well the neural network represents this as compared with the ground truth data. Therefore, when the S4ND model is able to produce DRMSE values on a coarser grid that has no performance degradation as compared with the training set, this suggests that the

https://doi.org/10.1017/jfm.2026.11929 Published online by Cambridge University Press

model is correctly representing the interscale energy transfer for various grid resolutions (filter cutoffs). Capturing this rate of energy transfer correctly is a necessary condition for the model to remain stable and accurate *a posteriori*, since incorrect energy transfer rates can lead to energy pile-up at large wavenumbers or over-dissipation of resolved structures.

4. *A posteriori* results

All neural network models are integrated into an in-house code, PadeOps (Ghate & Lele 2017), to be evaluated in an *a posteriori* setting. In these sections, two traditional models (the static Smagorinsky and sigma models) are compared with four neural network models (S4ND, TBNN, NN and NN-V as defined in the earlier section). These two traditional models were chosen as they represent one of the earliest subgrid stress models and a modern subgrid stress model that has many desirable properties such as correct behaviour near-wall. The static Smagorinsky model is used with a coefficient set to $C = 0.18$, while the sigma model is also run with a coefficient set to $C = 1.3$. These are all within the ranges commonly used for these two models (Smagorinsky 1963; Meyers & Sagaut 2006; Nicoud *et al.* 2011). The sigma model was chosen as the coefficient range works for many different flows and does not require clipping for channel flow, whereas models such as the dynamic Smagorinsky model may need clipping (Meneveau *et al.* 1996). Four neural network models are compared to highlight the benefits gained from increasing the complexity of neural network models, as well as to highlight the architectural effects of the S4ND model. For all cases, using no subgrid stress model results in the simulations producing NaNs (not a number).

4.1. *In-training set simulation results*

First, all models are evaluated on LES simulations with grid resolutions corresponding to in-training set filter width cases.

4.1.1. *Forced HIT at* $16\Delta_{DNS}$

The forced HIT simulation is run with a RK4–5 time stepping scheme (Bogacki & Shampine 1996) on a 64^3 grid, which corresponds to a $16\Delta_{DNS}$ filter width. The spatial discretization scheme is Fourier collocation, possible since the boundary conditions are triply periodic, with a domain length of 2π in each direction. The forcing scheme forces the energy in the wavenumber band below 2 to be constant, and the non-dimensional dissipation, ϵ is set to 0.103 for all simulations. Due to Fourier collocation, 2/3 dealiasing is applied. For this simulation, all cases have a Taylor Reynolds number (Re_λ) of approximately 433. As seen in figure 8, all the LES simulations with various models have reasonable spectra. The eddy viscosity models, as expected, are more dissipative as compared with filtered DNS, while the neural network models better approximate the overall shape of the spectra but are slightly under-dissipative. From the 'bump' in the spectra, the TBNN subgrid stress model is the most under-dissipative, followed by the NN model, then the S4ND and NN-V model. The SAE_E table validates this, as this metric suggests that the S4ND model and the NN-V model are the closest to the filtered DNS spectra, as lower SAE_E values are closer to filtered DNS statistical quantities. Overall, the NN-V and S4ND models have spectra that are extremely similar, suggesting that S4ND, even with bandlimiting, is not significantly reducing the performance of the neural network in forced HIT.

From figure 9, all neural network models are fairly good at predicting the temporal decorrelation, as they all follow the overall expected decay shape with the S4ND, NN-V and NN neural network models being slightly more accurate. When evaluating the spatial

https://doi.org/10.1017/jfm.2026.11929 Published online by Cambridge University Press

Model	SAE_E
S4ND	**2.55**
NN	**3.11**
TBNN	3.41
Sigma	10.59
Static	12.97
NN-V	**2.41**

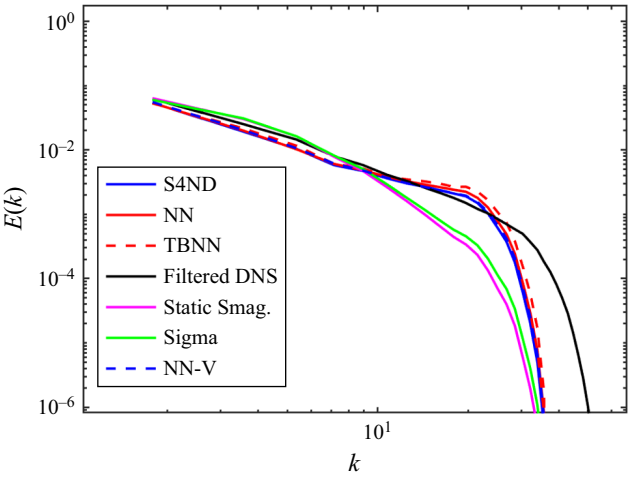
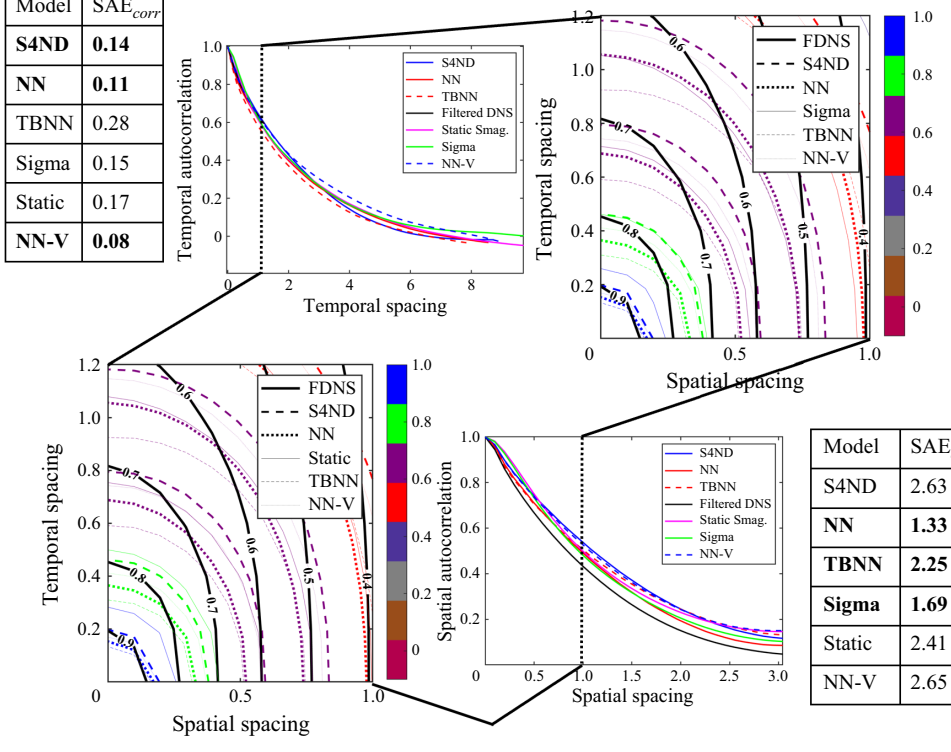


Figure 8. Forced HIT spectra for various SGS models *a posteriori* on a grid size corresponding to a filter width of 16 Δ_{DNS}.

Model	SAE_{corr}
S4ND	**0.14**
NN	**0.11**
TBNN	0.28
Sigma	0.15
Static	0.17
NN-V	**0.08**



Model	SAE_{corr}
S4ND	2.63
NN	**1.33**
TBNN	**2.25**
Sigma	**1.69**
Static	2.41
NN-V	2.65

Figure 9. Forced HIT spatial, temporal, and space–time correlations for various SGS models *a posteriori* on a grid size corresponding to a filter width of 16 Δ_{DNS}.

https://doi.org/10.1017/jfm.2026.11929 Published online by Cambridge University Press

Model	SAE_E
S4ND	**2.01**
NN	2.54
TBNN	10.5
Sigma	**2.75**
Static	6.63
NN-V	**1.76**

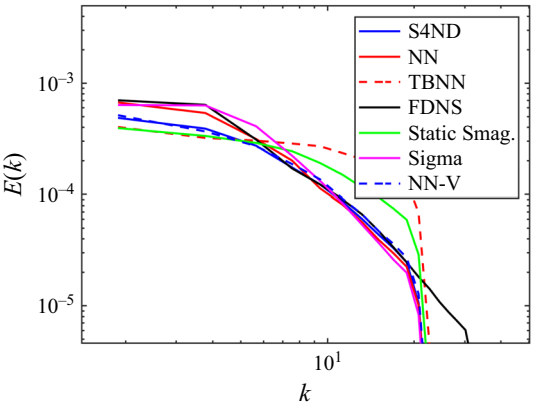


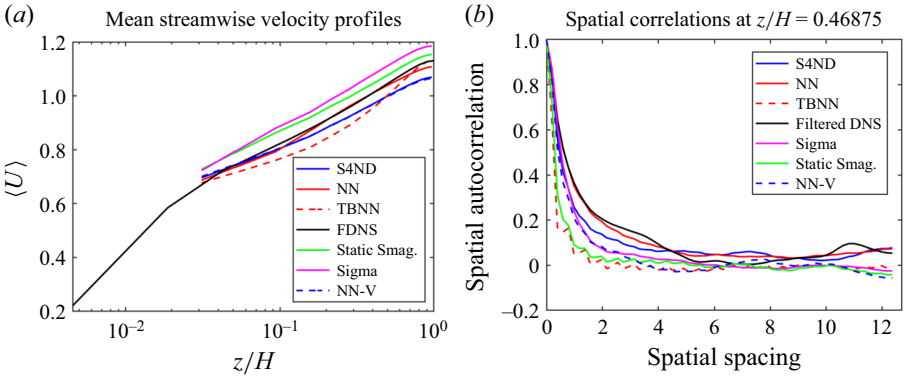
Figure 10. Channel flow streamwise spectra at $z/H = 0.46875$ for various SGS models compared with filtered DNS. Both the static Smagorinsky model and TBNN fail at predicting the overall spectra shape.

correlations, all models are fairly accurate for small spacings, but the S4ND and NN-V models are less accurate towards the larger spacings compared with both the sigma and NN models. While the overall spatial correlation shapes are preserved for all models, as seen from the SAE_{corr} for the spatial decorrelation, the S4ND and NN-V models have the largest error compared with the filtered DNS decorrelation. This suggests that the performance degradation for S4ND is not necessarily due to the state space model, but is likely a product of the normalization and inputs. From the space–time correlations, one can conclude that all the neural network models are far more accurate than either of the traditional models, especially for small spatial and temporal spacings. It is hard to determine which neural network model is more superior when it comes to space–time correlations. Overall, the benefits of neural network complexity is not clear in this set of simulations, as sometimes the S4ND model is better, while other times (such as when evaluating spatial correlations) it is worse. However, this simulation is run with an in-training set filter width of $16\Delta_{DNS}$, where it has already been established that one does not expect much difference between the S4ND and the CNN-based models. This is reflected in this set of simulations as the S4ND model is sometimes better, and sometimes worse than the NN model (although it is quite consistent with the NN-V model due to the same input/output normalizations). The S4ND model's strength, as seen *a priori*, is in extrapolative scenarios.

4.1.2. *Channel flow at* $16\Delta_{DNS}$

Similarly to the HIT cases, the channel flow simulation is also discretized with Fourier collocation in the homogeneous directions, while a sixth-order compact scheme is used in the wall-normal direction. The wall model used is not data-driven, and instead follows from the wall model used by Bou-Zeid *et al.* (2005). The simulation grid size is $128 \times 96 \times 32$, while the domain size is $[8\pi, 3\pi, 2]$, with the streamwise velocity pointing in the 'first' direction and the wall-parallel direction being the 'second' direction. All simulations run in this section correspond to a filter width of $16\Delta_{DNS}$ with a friction Reynolds number of $Re_\tau = 1000$, with $Re_\tau = u_\tau H/\nu$, where u_τ is the friction velocity, H is the channel half-height and ν is the kinematic viscosity. As seen in figure 10, in a more complex flow (as compared with HIT), the utility of more complicated neural network models is starting to show. The TBNN is unable to predict even the spectra properly, and is even worse than static Smagorinsky. Meanwhile the overall shapes associated with the NN and S4ND model is significantly closer to that of the filtered DNS profile. While the

https://doi.org/10.1017/jfm.2026.11929 Published online by Cambridge University Press



Model	SAE_u
S4ND	**0.68**
NN	**0.18**
TBNN	0.56
Sigma	0.98
Static	**0.53**
NN-V	0.72

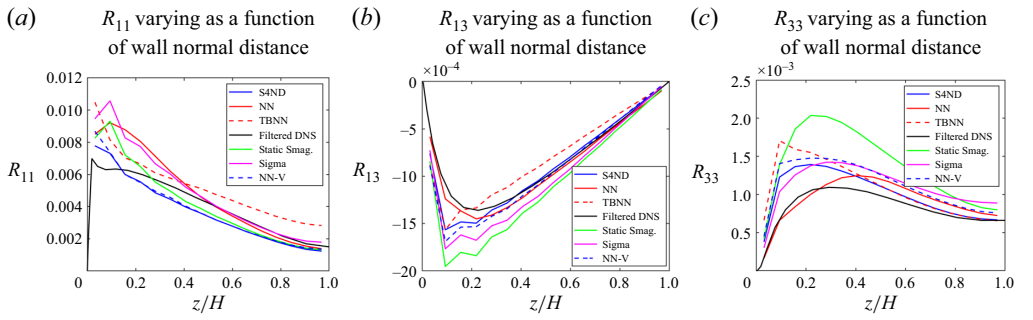
Model	SAE_{corr}
S4ND	**1.14**
NN	**0.24**
TBNN	3.63
Sigma	**1.62**
Static	3.07
NN-V	1.89

Figure 11. Channel flow macroscale quantities analysed for various SGS models. The mean velocity profiles and spatial correlations for the S4ND and NN model look the closest compared with filtered DNS. As seen, the TBNN completely fails in this setting, predicting jagged spatial correlations. (*a*) Mean streamwise velocity profiles. (*b*) Spatial correlations at $z/H = 0.46875$.

S4ND model is slightly over-dissipative at small scales, it follows the inertial range of the spectra better than the NN model when comparing these two profiles with the filtered DNS profile. However, as seen with NN-V, the S4ND model being over-dissipative is not due to the state space model, as the NN-V model also follows a similar spectra, suggesting that the differences in the spectra are more due to normalization and different inputs into the neural network. Overall, from the SAE_E table, one can see that the S4ND model, NN-V and the sigma model are more similar to filtered DNS as compared with the other models.

From figure 11(*a*) to 11(*b*), one can see a similar story when comparing the TBNN model with its more complex counterparts. The mean profiles associated with the TBNN model are inaccurate as compared with filtered DNS, not even predicting the shape properly, while the other three neural network models perform far better. In fact, the spatial correlations associated with the TBNN model at aggressive grid spacings highlights that it is completely inaccurate, as the spatial correlations become jagged. Meanwhile, the mean profiles predicted by all the other models that are not the TBNN are reasonable, with the NN model being more accurate as compared with the S4ND model. The NN-V and S4ND models have very similar mean profiles, which highlights that the S4ND architecture and the CNN architectures have relatively similar performance when the training set is still in filter-width. For the spatial correlations, both the S4ND and the NN model are far better than the traditional models, with the NN model being more accurate. When comparing NN and NN-V, one can isolate the effects of the normalization scheme and inputs. The

https://doi.org/10.1017/jfm.2026.11929 Published online by Cambridge University Press



Model	SAE_{R11}
S4ND	**0.0098**
NN	0.0127
TBNN	0.0164
Sigma	0.0127
Static	**0.0102**
NN-V	**0.0102**

Model	SAE_{R13}
S4ND	**0.0013**
NN	**0.0008**
TBNN	0.0023
Sigma	0.0026
Static	0.0039
NN-V	**0.0018**

Model	SAE_{R33}
S4ND	**0.0027**
NN	**0.0019**
TBNN	**0.0041**
Sigma	0.0044
Static	0.0087
NN-V	0.0047

Figure 12. Channel flow Reynolds stress profiles for various SGS models at a filter width corresponding to 16 Δ_{DNS}. (*a*) Here R_{11} varying as a function of wall normal distance. (*b*) Here R_{13} varying as a function of wall normal distance. (*c*) Here R_{33} varying as a function of wall normal distance.

improvements associated with the NN model is due to the inputs and normalization scheme used, not the architecture as NN-V has the same exact architecture as the NN model. From figure 11(*b*), it is seen that the NN-V model has worse spatial correlations as compared with the S4ND model, which highlights that even for an in-training set filter width, when comparing across the same normalizations, the convolutional models can start to lose performance. This is because channel flow operates on anisotropic grids (different $\mathrm{d}x$, $\mathrm{d}y$, $\mathrm{d}z$ spacings, but constant spacing in each direction), while convolutional-based models implicitly assume isotropic grids (HIT grids). This means that for anisotropic grids, the convolutional models will not perform as well when all things are considered equal (such as same normalization and same inputs).

Reynolds stress profiles are also plotted, which are seen in figure 12(*a*) to 12(*c*). Overall, both the S4ND model and NN model are more accurate compared with the other models, with the S4ND model and NN model trading off in accuracy. When comparing the NN model with the S4ND model, it would seem like for channel flow at this resolution, the NN model is slightly better than the S4ND model. This is seen in the R_{13} profile, as the overall shape of the Reynolds stress profile is better captured by this model (although S4ND is better at capturing the R_{11} profile). A different trend emerges when comparing the S4ND and NN-V model. In all cases, the NN-V model is either slightly worse (R_{11}) or worse than the S4ND model (R_{13}, R_{33}). This suggests that when accounting only for architectural effects, the S4ND model provides more accuracy on anisotropic grids than the NN-V model, which uses discrete convolutions. Since discrete convolutions assume isotropic grids, they will suffer a decrease in accuracy in flows with anisotropic grids such as channel flow when compared with the S4ND model.

While the NN-based model (with different inputs and normalization) is indeed slightly better than the S4ND model at this resolution, LES grid spacing is ultimately determined

https://doi.org/10.1017/jfm.2026.11929 Published online by Cambridge University Press

Model	SAE_E
S4ND	**0.67**
NN	**1.71**
Sigma	2.71
Static	3.31
NN-V	**0.72**

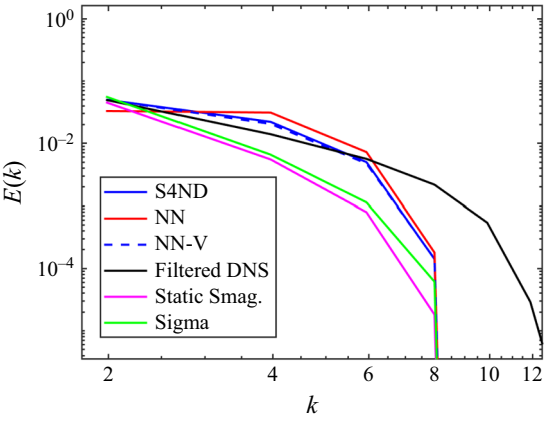


Figure 13. Spectra for various SGS models on a grid corresponding to an extrapolative filter width of $64\Delta_{DNS}$.

by the user, and therefore the models must also be evaluated on an out-of-training set filter width. As seen in Nguyen *et al.* (2022), the S4ND model, through the use of bandlimiting, explicitly learns smoother kernels that would generalize better. However, this is an implicit form of regularization that will slightly decrease the performance of the S4ND model on filter widths that it was trained on. In the in-training set filter width *a posteriori* results for HIT and channel flow, the bandlimiting used actually does not decrease the performance of the S4ND model. For HIT, the S4ND model is on par with NN-V and for channel flow, since it accepts the different grid spacings in each dimension, the S4ND model is actually better than NN-V. This suggests that even though the S4ND model is biasing more towards extrapolation by regularizing the continuous convolution kernel to be smoother, the larger scales of the flow that are captured with the regularized convolution kernel are still enough to infer the subgrid stress tensor *a posteriori*.

4.2. *Filter width extrapolation results*

Now, all the models are evaluated on an extrapolative filter width that is out of the training set. This will test the generality of the neural network models to see if true 'scale-similarity' has been learned. For true scale-similarity to be learned, the neural network must produce results on coarser grids that are reasonable for both HIT and channel flow. Given that the TBNN model was completely inaccurate for channel flow already (shown in figure 11*b*), this section focuses on evaluating the more complex neural network models and comparing them with the traditional models.

4.2.1. *Forced HIT at* $64\Delta_{DNS}$

Now, fixing all the previous simulation parameters to be identical to the $16\Delta_{DNS}$ HIT simulation except for grid spacing, all models are run on a significantly coarser LES where the number of points is 16^3, corresponding to a filter width of $64\Delta_{DNS}$. The spectra is shown in figure 13.

From the spectra, one can conclude that the sigma and static Smagorinsky models, consistent with the in-training set filter widths, are still over-dissipative. In addition, the S4ND model produces more accurate spectra as compared with the NN model. Furthermore, the S4ND and NN-V models produce similar spectra. Overall, the spectra suggest that the data-driven models are better at predicting the filtered DNS spectra, even

https://doi.org/10.1017/jfm.2026.11929 Published online by Cambridge University Press

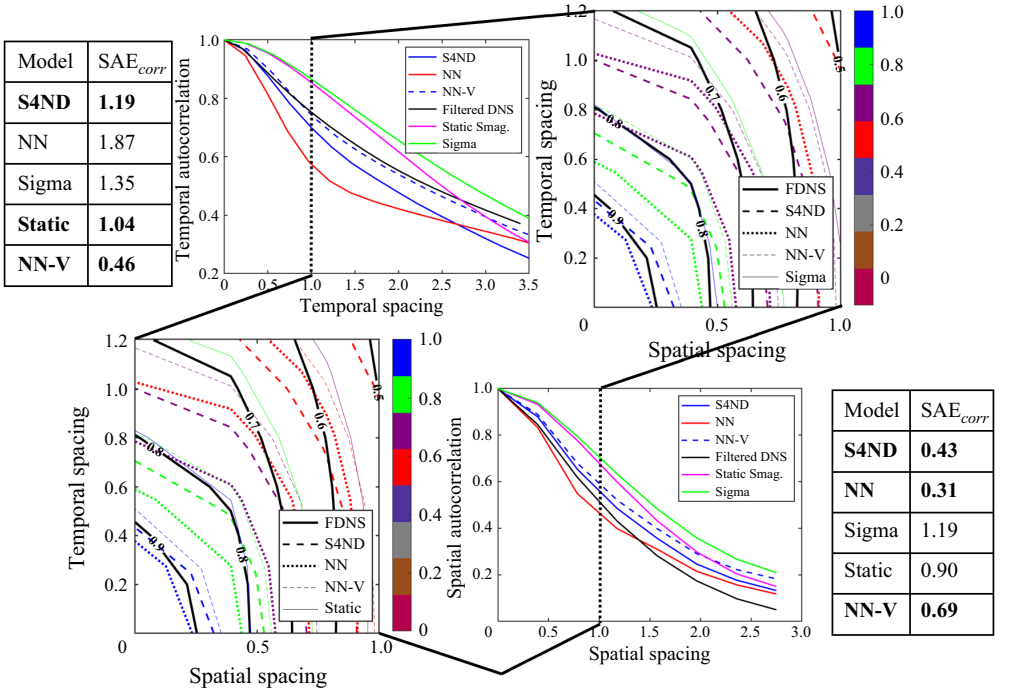


Figure 14. Forced HIT spatial, temporal and space–time correlations for various SGS models on a grid corresponding to an extrapolative filter width of $64\Delta_{DNS}$.

when extrapolating, when compared with the traditional models. Furthermore, additional quantities such as correlations are shown in figure 14.

At this more aggressive grid spacing, the temporal autocorrelations suggest that the S4ND model is far better at extrapolating than the NN-based model, and outperforms both traditional models. First, the traditional models are unable to predict the initial temporal decay properly, significantly overpredicting the temporal autocorrelation. Meanwhile, the NN model decorrelates far too quickly, and then decorrelates far too slowly as the temporal spacing increases. When comparing NN-V with S4ND, the temporal correlations for NN-V are surprisingly better than S4ND. This suggests that with this output normalization, some sense of scale-similarity is 'baked-in'. In addition, since the HIT grid is isotropic, and the models are trained on two grid resolutions, the discrete convolutions in NN-V are able to learn a sense of scale similarity. This also highlights the importance of output normalizations and inputs, as the NN model, with different inputs and output normalization, was unable to extrapolate while the NN-V model was able to, despite both using CNNs. In general, the top three lowest SAE_{corr} include the S4ND, NN-V and static Smagorinsky models. However, both quantitative and qualitative analysis must be used, as SAE_{corr} for the static Smagorinsky model is lower than the S4ND model, but this can be attributed to the static Smagorinsky model crossing the filtered DNS line and thus the integrated distance between the two curves is smaller due to this crossover. As seen from the temporal decorrelation graph, the overall shape of the static Smagorinsky correlation is quite different from filtered DNS.

For the spatial correlations, when comparing the S4ND model with the NN model and traditional models, one can see that the S4ND model produces the most similar decorrelation shape to the filtered DNS profile. Meanwhile, the SAE_{corr} for the NN model

https://doi.org/10.1017/jfm.2026.11929 Published online by Cambridge University Press

is artificially lowered due to the line crossing the filtered DNS line. When comparing the S4ND model and the NN-V model, one can see that the S4ND model is more accurate than the NN-V model. For the spatial correlations, the S4ND model is the most accurate out of all the models. From the space–time correlations, one can conclude that the traditional models completely fail in predicting the correlations, as the black lines denote the predicted contour for filtered DNS, and the coloured lines denote the contours associated with a particular SGS model. For example, the blue contours denote the 0.9 contours for all the SGS models. Meanwhile, the traditional model 'thin blue contours' in the plots are extremely off, showing a significant overprediction of the correlation as compared with both the neural network models. While both the NN and S4ND model do well, the NN-V model performs a little worse, especially for small decorrelation scales. This is seen as it is slower to decorrelate than the S4ND model, as the blue contour is farther away from the 0.9 filtered DNS contour. Thus, one can see that the S4ND model, with the continuous kernel and bandlimiting, overall produces more accurate space–time correlations and spatial decorrelations as compared with all the other SGS models evaluated. Meanwhile, while the NN-V model produces good temporal decorrelations, it still suffers in accuracy when comparing the small-scale space–time decorrelations and the overall spatial autocorrelations, suggesting that S4ND's architectural effects are directly responsible for this increase in accuracy. By comparing NN and NN-V, one can also conclude that correct inputs and output normalization is essential for scaling to coarser grid spacings.

4.2.2. *Channel flow at* $64\Delta_{DNS}$

Channel flow is also evaluated at a more aggressive, extrapolative grid spacing. Keeping all other simulation parameters fixed, the grid spacing is changed to $32 \times 24 \times 8$, corresponding to a filter width of $64\Delta_{DNS}$. Otherwise, the simulation parameters are kept the same as the finer grid simulation, corresponding to a Re_τ of 1000. From the mean profiles, shown in figure 15, the S4ND model is able to preserve the slope of the mean streamwise velocity profile as compared with filtered DNS, while the NN model fails. Meanwhile, the two traditional models are more inaccurate as they are farther from the filtered DNS profile. When comparing the S4ND model with the NN-V model, one can see that while the NN-V model is able to improve upon the NN model form by using different inputs and output normalization, the overall slope is still incorrect when compared with the S4ND model. This is also seen in the SAE_u tables, with the S4ND being the lowest (and the NN model being not far behind only because it crosses the filtered DNS profile).

The Reynolds stress profiles are also plotted, as seen in figure 16. From these profiles, the accuracy of the S4ND model in an extrapolative setting is highlighted. Comparing the S4ND model with the NN model, the S4ND model is far more accurate as the NN model has significant trouble extrapolating. The NN model profile shapes are extremely off and the magnitudes are incorrect even when compared with the traditional models. As seen from every single profile, the S4ND is far better at preserving the overall shape and magnitude as compared with both the NN model and all the traditional models. However, this could also potentially be because the NN model's output scaling, which was different from the S4ND's output scaling, was just not as optimized for extrapolation to coarse grids. This hypothesis can be ruled out by looking at the NN-V model, which has the same exact inputs and output scaling as the S4ND model. Here, the NN-V model also is significancy worse than the S4ND model. While it does not overpredict as much as the NN model, suggesting that the output scaling used for the NN-V model and S4ND model are better at scaling, the overall profile shapes are oftentimes incorrect, especially R_{33}, where

https://doi.org/10.1017/jfm.2026.11929 Published online by Cambridge University Press

Model	SAE_u
S4ND	**0.25**
NN	0.31
Sigma	0.32
Static	**0.28**
NN-V	0.30

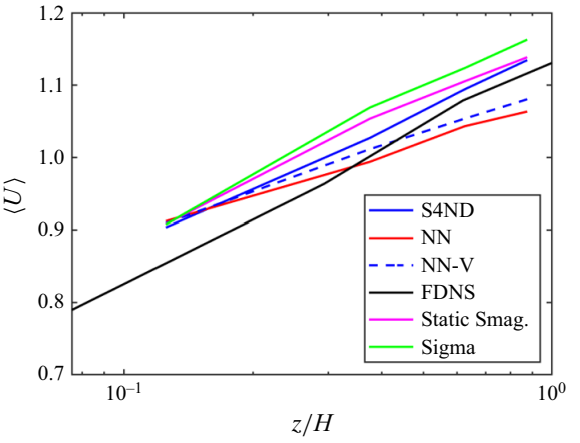
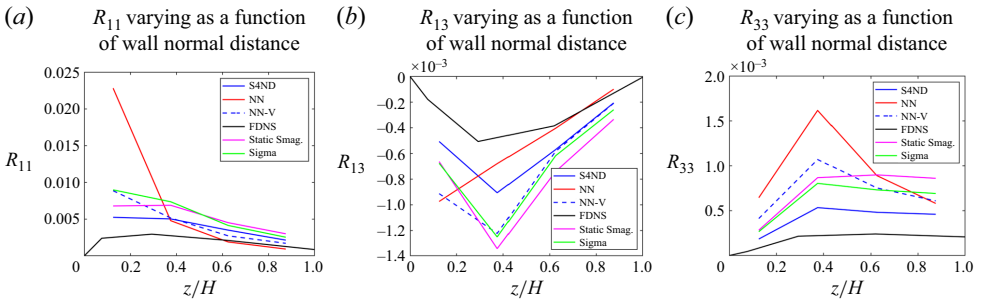


Figure 15. Channel flow mean profiles for various SGS models on a grid corresponding to an extrapolative filter width of $64\Delta_{DNS}$.

it overpredicts the Reynolds stress peak such that it is over five times that of filtered DNS. This suggests that it is not the output scaling that is causing the performance improvement of S4ND for channel flow extrapolation, but solely due to the S4ND architecture. This highlights the extrapolatory capability of the S4ND model as it outperforms the NN and NN-V model while also being significantly better than the traditional models. In all situations for coarse-grid channel flow, the S4ND-based model, when integrated into a LES simulation, gives results closest to the filtered DNS profiles.

Overall, combined with the HIT results from the previous section, the S4ND model is a more flexible model since it is more accurate in extrapolating than any of the other neural network models tested (especially for channel flow), and is more accurate than the traditional SGS models even when extrapolating in grid resolution. Furthermore, by comparing with NN-V, one can highlight that while the output scaling does play a role in helping to improve extrapolation, the S4ND architecture is still key in extrapolating, especially in channel flow. This also suggests that the in-training set filter width results, where oftentimes the NN model was better, highlight that the normalization used for the NN model helped contribute to the NN model 'overfitting' to the in-training set filter width cases, producing better results *a posteriori* on these cases and then completely failing to extrapolate to HIT or channel flow on coarser grids. This can be contrasted with NN-V and S4ND, where the inputs and output normalization allowed for less *a posteriori* performance for the in-training set filter width cases, but then allowed for better extrapolation to coarser grids. In the simpler HIT case with isotropic grids, the normalization allowed the NN-V model, which uses CNNs, to extrapolate well to HIT, but it still failed in the more complex channel flow case with anisotropic grids. The S4ND meanwhile, was able to extrapolate to both coarser grid cases reasonably well, as a consequence of its architecture. This validates what was brought up in the previous section, where the use of bandlimiting is effectively a regularizer that smooths out the convolution kernel, allowing the S4ND model to be more accurate in out of training set filter widths, while the models with CNNs are more likely to 'overfit' to the specific grid resolutions that it has been trained on, particularly for channel flow, where due to anisotropic grids, the primary assumption of CNNs is already violated (isotropic grids). Physically, the S4ND coarse grid extrapolation cases highlight that the continuous convolution kernels with bandlimiting allow the model to respect the scale-dependent structure of turbulence across

https://doi.org/10.1017/jfm.2026.11929 Published online by Cambridge University Press



Model	SAE_{R11}
S4ND	**0.0069**
NN	0.0223
Sigma	0.0140
Static	0.0122
NN-V	**0.0095**

Model	SAE_{R13}
S4ND	**8.6×10^{-4}**
NN	**9.1×10^{-4}**
Sigma	1.5×10^{-3}
Static	1.8×10^{-3}
NN-V	1.6×10^{-3}

Model	SAE_{R33}
S4ND	**8.8×10^{-4}**
NN	3.0×10^{-3}
Sigma	**1.7×10^{-3}**
Static	2.1×10^{-3}
NN-V	2.1×10^{-3}

Figure 16. Channel flow Reynolds stress profiles for various SGS models on a grid corresponding to an extrapolative filter width of $64\Delta_{DNS}$. (*a*) Here R_{11} varying as a function of wall normal distance. (*b*) Here R_{13} varying as a function of wall normal distance. (c) Here R_{33} varying as a function of wall normal distance.

filter widths. When the filter width increases (grid gets coarser), more energy is filtered out, and the subgrid stress model must account for a larger fraction of the unresolved energy. Particularly for channel flow, since the S4ND model produces more accurate Reynolds stress profiles and mean profiles on coarser grids when compared with the NN-V (which has identical inputs, outputs and normalization), this suggests that the continuous formulation adjusts to the balance between resolved and modelled scales more naturally than discrete convolutions. The improved performance of the S4ND model on coarser grid resolutions is therefore attributed to the continuous formulation with bandlimiting, which ensures that the learned closure adapts to changing filter width, maintaining filter-consistent spectral content while capturing the interscale energy transfer (which can be seen directly in the *a priori* results).

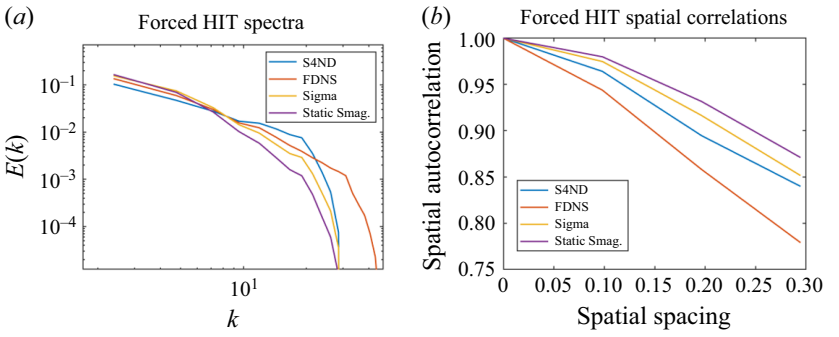
5. Reynolds number scaling

The Reynolds number scaling capabilities of the S4ND model are now investigated. First, the S4ND model, which is trained on forced HIT with $Re_\lambda = 433$ and channel flow with $Re_\tau = 1000$ data, is tested *a posteriori* on both forced HIT with $Re_\lambda = 2400$ and channel flow with $Re_\tau = 5200$. The reference data are also queried from the JHTDB, with the DNS HIT resolution being 4096^3 on the same $[2\pi, 2\pi, 2\pi]$ domain, while the DNS channel flow resolution is $10240 \times 1536 \times 7680$ on the same $[8\pi, 2, 3\pi]$ domain when normalized by channel half-height. Next, the S4ND model is then tested on cases where not only the grid width, but the Reynolds number is also extrapolated.

5.1. *Reynolds number scaling with DNS data*

Both HIT and channel flow Reynolds number scaling is investigated. The HIT simulation is conducted on a 64^3 grid on a triply periodic domain of length 2π in each direction, with the dimensional dissipation, ϵ, set to 1.4144. The channel flow simulation is conducted on a grid of size $128 \times 96 \times 32$ on a domain of size $8\pi \times 3\pi \times 2$. Due to the increase

https://doi.org/10.1017/jfm.2026.11929 Published online by Cambridge University Press



Model	SAE_E
S4ND	**4.37**
Sigma	**6.22**
Static	12.8

Model	SAE_{corr}
S4ND	**0.12**
Sigma	**0.16**
Static	0.20

Figure 17. Macroscale quantities analysed for forced HIT at a higher Reynolds number not seen in the training set. (*a*) Forced HIT spectra. (*b*) Forced HIT spatial correlations.

in Reynolds number, while the $\Delta_{LES} = 64\Delta_{DNS}$ for HIT (reference DNS is now four times finer than the previous DNS) and $\Delta_{LES} = 80\Delta_{DNS}$ for the channel flow simulation (reference DNS is now five times finer than the previous DNS), the overall grid spacing is still considered an in-training set. This is because the grid spacing $\mathrm{d}x$, $\mathrm{d}y$, $\mathrm{d}z$ are grid spacings that the neural network has been trained on in the training data. As seen from Cho *et al.* (2024), neural networks extrapolate well to higher Reynolds number as long as $\mathrm{d}x$, $\mathrm{d}y$, $\mathrm{d}z$ is the same as the grid spacing of the training data, even if Δ_{LES}/η is larger than the training data, where η is the Kolmogorov length scale. As such, we will still consider this an 'in-training set' filter width.

5.1.1. *The HIT*

The Reynolds number scaling results for forced HIT are shown in figures 17(*a*) and 17(*b*). In these figures, the S4ND model is compared with both the static Smagorinsky and sigma models, as well as filtered DNS to evaluate the extrapolatory capability of the S4ND model.

As seen, when the Reynolds number is larger than the training set Reynolds number, the S4ND model still produces reasonable results for both spectra and spatial correlations. Overall, the same trends hold when comparing the S4ND simulations that were run with an in-training set Reynolds number versus an extrapolatory Reynolds number. For example, the energy spectra for S4ND is still slightly underdissipative, while the spatial correlations are more accurate for the S4ND model compared with the spatial correlations produced by the traditional subgrid stress models. In this case, the spatial correlations are truncated since there is only one ensemble at this Reynolds number for the filtered DNS profile (JHTDB provides one snapshot), and thus the spatial correlation is truncated at a point where it is still reasonably converged. When taking a look at the SAE for both the energy spectra and the correlations, the S4ND model has the lowest error out of all the models compared. The compensated spectra, which premultiplies the energy spectra with a $k^{5/3}$ factor, are shown in figure 18. As seen, the filtered DNS reference has an inertial range,

https://doi.org/10.1017/jfm.2026.11929 Published online by Cambridge University Press

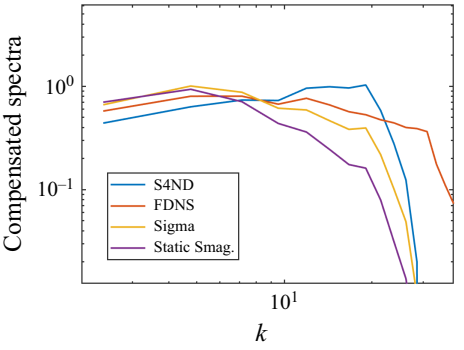
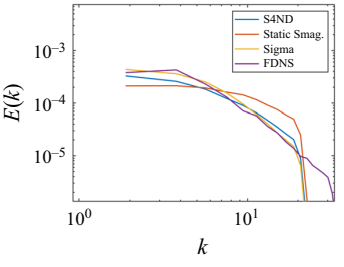


Figure 18. Compensated spectra for forced HIT at a higher Reynolds number not seen in the training set.

(*a*) Streamwise spectra at $z/H = 0.46875$

Model	SAE_E
S4ND	**2.32**
Sigma	**1.41**
Static	5.79



(*b*) Spatial correlations at $z/H = 0.46875$

Model	SAE_{corr}
S4ND	**1.59**
Sigma	**2.16**
Static	5.79

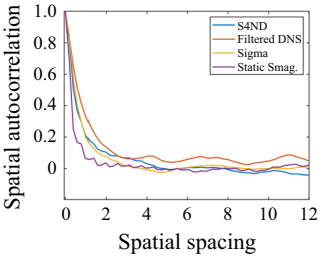


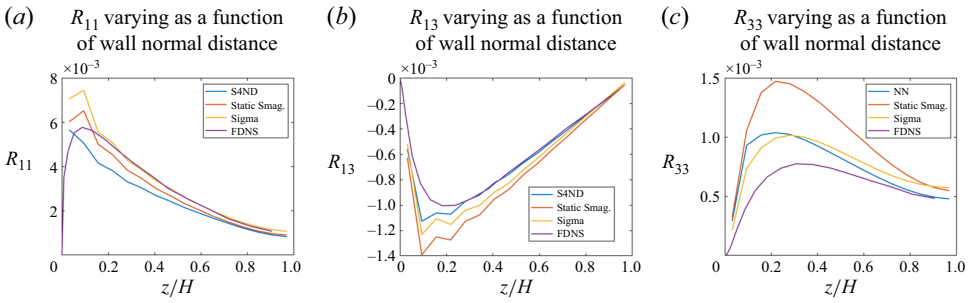
Figure 19. Macroscale quantities analysed for channel flow at a higher Reynolds number not seen in the training set. (*a*) Streamwise spectra at $z/H = 0.46875$. (*b*) Spatial correlations at $z/H = 0.46875$.

but the inertial range varies from being slightly less than the $k^{5/3}$ scaling to slightly more than the $k^{5/3}$ scaling. This is also seen in the SGS models, with the S4ND model being consistent with previous results (slightly under-dissipative) while the traditional SGS models are also consistent with previous results (slight over-dissipative). Overall, the S4ND model is able to successfully extrapolate to a higher Reynolds number for forced HIT.

5.1.2. *Channel flow*

For channel flow Reynolds number scaling, first the spectra at a specific wall normal location and the spatial correlations are shown in figures 19(*a*) and 19(*b*). As seen in the plots, the S4ND model produces spectra and spatial correlations that match filtered DNS profiles when the Reynolds number is out of the training set. There is no obvious performance degradation when compared with the in-training set filter width, and the overall profiles match the filtered DNS profiles. Furthermore, the S4ND Reynolds stress profiles are seen in figures 20(*a*), 20(*b*) and 20(*c*). Overall, the S4ND model is able to capture the overall shape and magnitude of the Reynolds stress profiles associated with channel flow. From all these plots, the S4ND model is in general, consistent with the in-training set Reynolds number, with no obvious performance degradation when the Reynolds number is out of the training set. This is corroborated by looking at the SAE, where the S4ND model produces reasonable SAE for all quantities. Combining both the channel flow and forced HIT results, one can conclude that the S4ND model is able to

https://doi.org/10.1017/jfm.2026.11929 Published online by Cambridge University Press



Model	SAE_{R11}
S4ND	0.0092
Sigma	**0.0050**
Static	0.0059

Model	SAE_{R13}
S4ND	**7.05×10^{-4}**
Sigma	1.3×10^{-3}
Static	2.3×10^{-3}

Model	SAE_{R33}
S4ND	**0.0025**
Sigma	0.0027
Static	0.0062

Figure 20. Channel flow Reynolds stress profiles for various SGS models at a higher Reynolds number not seen in the training set. (*a*) Here R_{11} varying as a function of wall normal distance. (*b*) Here R_{13} varying as a function of wall normal distance. (*c*) Here R_{33} varying as a function of wall normal distance.

extrapolate to higher Reynolds number flow cases and still match filtered DNS reasonably well.

5.2. *Extreme Reynolds number scaling without DNS data*

Now, true 'out of training set' filter width and Reynolds number extrapolation is run, where the grid spacings $\mathrm{d}x$, $\mathrm{d}y$, $\mathrm{d}z$ are larger than the training set, and where Δ_{LES}/η is also far larger than the training set. While there are no DNS results at these larger Reynolds numbers, one can evaluate the stability of the subgrid stress models in extreme Reynolds number scaling scenarios.

5.2.1. *HIT*

The HIT simulation is evaluated on a 16^3 grid, corresponding to a grid spacing in each direction that is twice as large as the training set. Furthermore, five Reynolds numbers based on the Taylor microscale are tested: $Re_1 = 4\,800$, $Re_2 = 19\,500$, $Re_3 = 56\,900$, $Re_4 = 515\,000$, $Re_5 = 508\,900\,000$. The S4ND model is only trained on $Re = 433$. As seen in figures 21(*a*) and 21(*b*), the neural network is able to ensure that the LES simulation is stable, even when extrapolating in both grid spacing and Re. Overall, the spectra and spatial correlations look reasonable for all Re, showing that the S4ND model produces reasonable results even if the Reynolds number is over 1 000 000 times that of the training set. Given the coarseness of the grid, the results should not change drastically with Reynolds number, which is demonstrated in these plots. In addition, up to Re_3, the trends are monotonic, suggesting that the S4ND model can capture trends up to around 70 times the Reynolds number in the training set. From Re_1 to Re_5, at no point is a 'tail-up' in the spectra seen, which is expected for ANN-based subgrid stress models that are trained on only one Reynolds number, as seen in Cho *et al.* (2024).

5.2.2. *Channel flow*

The same neural network is also integrated into a channel flow simulation with varying Reynolds numbers to test the extrapolatory capabilities of the neural network, on a grid

https://doi.org/10.1017/jfm.2026.11929 Published online by Cambridge University Press

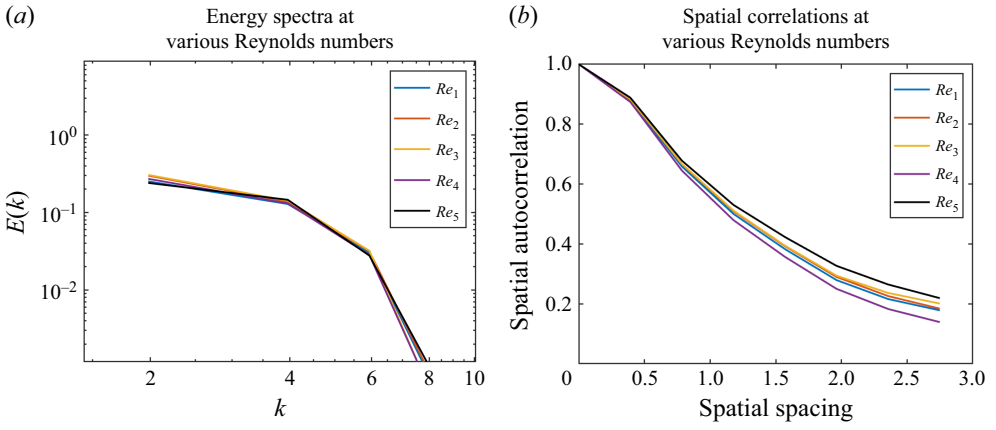


Figure 21. Extreme Reynolds number scaling for HIT. (*a*) Energy spectra at various Reynolds numbers. (*b*) Spatial correlations at various Reynolds numbers.

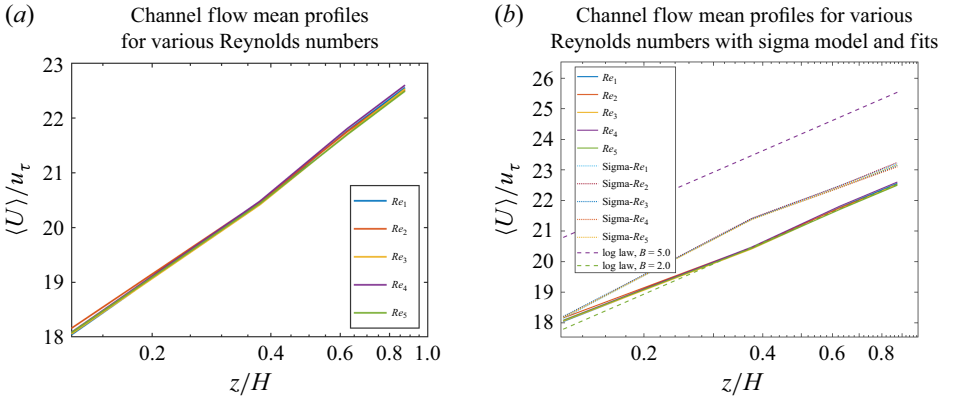


Figure 22. Various mean profiles for channel flow extrapolation to extreme Reynolds numbers. (*a*) Channel flow mean profiles for various Reynolds numbers. (*b*) Channel flow mean profiles for various Reynolds numbers with sigma model and fits.

of size $32 \times 42 \times 8$ on a domain of size $8\pi \times 3\pi \times 2$ (grid spacing is coarser than the training set). The Reynolds number based on friction velocity tested involve $Re_1 = 5200$, $Re_2 = 20\,000$, $Re_3 = 50\,000$, $Re_4 = 500\,000$ and $Re_5 = 500\,000\,000$. The neural network is trained only on Reynolds number based on friction velocity of $Re = 1000$.

From figure 22(*a*), when scaled with friction velocity, the channel mean streamwise velocity profiles collapse nicely, suggesting that these profiles are reasonable for all Reynolds numbers. To fully verify this, the log-law profiles are plotted, along with the sigma model mean profile results in figure 22(*b*). Due to the large offset from the S4ND model, the sigma model was also run as verification to isolate if this offset was associated with the subgrid stress model or the wall model. As seen, using the classic offset of $B = 5.0$ is insufficient, likely due to the error in the wall model. This error can be attributed to the wall model since two different SGS models (sigma model and data-driven S4ND model) produce extremely similar results at the first off-wall grid point, which is dominated by the wall model error. In this extremely coarse and extremely

https://doi.org/10.1017/jfm.2026.11929 Published online by Cambridge University Press

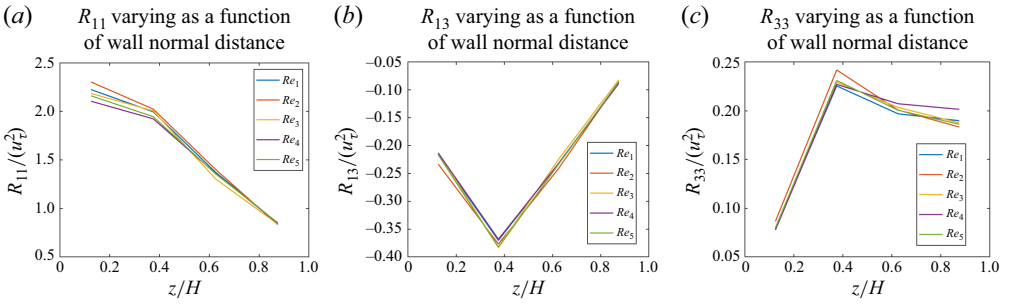


Figure 23. Reynolds stress profiles for various extreme Reynolds numbers for the S4ND model. (*a*) Here R_{11} varying as a function of wall normal distance. (*b*) Here R_{13} varying as a function of wall normal distance. (*c*) Here R_{33} varying as a function of wall normal distance.

aggressive Reynolds number extrapolation case, the S4ND model is able to produce a profile that is closer to the log-law profile than the sigma model. From all this, the neural network is seen to provide reasonable results even when the Reynolds number is extrapolated by over 500 000 times the Reynolds number of the training set when considering mean profiles. This is further corroborated in figures 23(*a*), 23(*b*) and 23(*c*), where the Reynolds stresses also collapse reasonably well when scaled with friction velocity. There is no significant performance degradation visible at first glance as the Reynolds number increases, suggesting that the S4ND model remains stable even at Reynolds numbers significantly larger than the training set Reynolds number. While exact comparisons cannot be made as there is no DNS at these higher Reynolds number cases ($Re_2 - Re_5$), if the neural network is unable to extrapolate, as seen in Cho *et al.* (2024), the change in shape of the profiles of turbulent statistical quantities is quite obvious to see.

In order to provide first-order analysis on the reasonableness of the channel flow scaling results, the implied skin friction coefficient, C_f is plotted, as well as the expected inverse $log(Re)$ scaling that is supported by Schultz & Flack (2013). These results are shown in figure 24. As seen, the implied C_f is decreasing for the first four Reynolds numbers, and is still reasonably following the overall trend. However, the last Reynolds number suggests an increase in C_f, suggesting that the results are less trustworthy at the highest Reynolds number. However, it is very likely this is more of a wall modelling error as opposed to a subgrid stress model error, as in these really coarse simulations there are eight points across the channel and the wall model's accuracy is suspect at these extreme Reynolds number cases (Re_τ of 500 000 000). This argument is further enhanced when the sigma model results are taken into account, as it exhibits similar trends as compared with the S4ND model. Therefore, the S4ND model likely extrapolates well to these extremely out of training set Reynolds numbers, and the error that is seen can be attributed to the non-data-driven wall model. Even if this largest Reynolds number data point is discarded, the S4ND model still obtains a 500 times friction Reynolds number scaling when the model is only trained on one friction Reynolds number. The largest Reynolds number data point is mainly kept to show that even if the results are less trustworthy, the subgrid stress model still remains stable even in this sort of highly extrapolatory situation.

While another way to ensure that the neural networks are stable during Reynolds number scaling is to use clipping, where values of the SGS tensor are set to zero if the local dissipation is negative. While this is an effective method, it is ultimately unphysical. In all these cases, no clipping is utilized and the S4ND model is stable without clipping.

https://doi.org/10.1017/jfm.2026.11929 Published online by Cambridge University Press

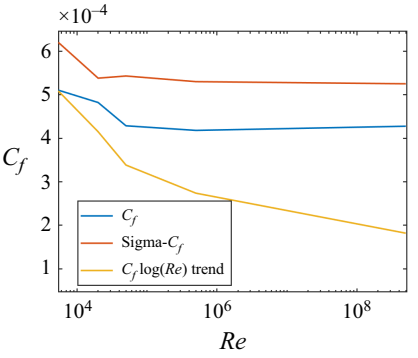


Figure 24. Implied wall skin friction coefficient versus a first-order trend, with the x-axis being friction Reynolds number. The trends show that the error is likely due to the wall model, as both the S4ND model (plotted in blue) and the sigma model (plotted in red) show similar trends and deviation from the expected trend (plotted in yellow (Schultz & Flack 2013)).

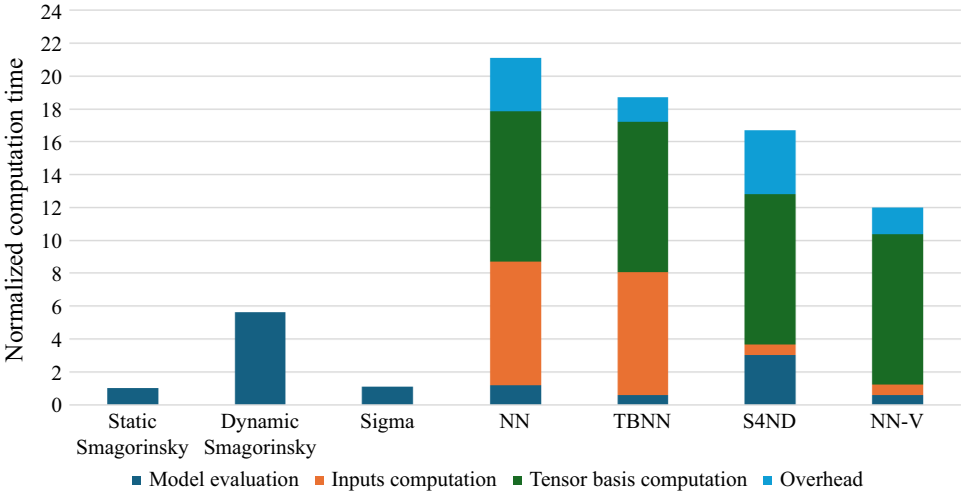


Figure 25. Computational cost of various subgrid stress models. Overhead refers to the time it takes to load the neural network from disk and send the inputs from the LES to the neural network for computation, and then returning the neural network subgrid stress output back to the LES.

5.3. *Computational cost*

The computational cost of various models is listed, normalized by the static Smagorinsky computation time. In general, using the TBNN backend is more expensive due to the computation of the invariants and the tensor basis, which are the dominant computational costs of the model, as seen in figure 25. Overall, the actual cost to evaluate the neural network models is cheaper than a dynamic Smagorinsky evaluation, but the computation of the invariants and associated tensor basis significantly dominate the time. Overall, the neural network models are around three to four times more expensive than a dynamic Smagorinsky evaluation. Due to the use of the velocity gradient tensor instead of the invariants, the S4ND model is less costly than the TBNN and the NN model. However, even with the increase in computation cost, the simulations are still fairly computationally

https://doi.org/10.1017/jfm.2026.11929 Published online by Cambridge University Press

inexpensive, requiring less than a day to complete even when evaluating the most expensive neural network models.

Due to the architectural sophistication of the S4ND model, the model is approximately 2.5 times more expensive than the NN model when evaluating the neural network. However, the S4ND model is able to gain extrapolatory capability by being grid-spacing aware as opposed to the discrete convolutions that the NN model uses. In addition, the trade off between directly enforcing physical constraints and additional runtime is clear. If directly enforcing physical constraints (for example through a TBNN backend), the cost to compute the tensor basis can be as much as two dynamic Smagorinsky evaluations. If the invariants are used as inputs, then the cost to compute the invariants would cost around another dynamic Smagorinsky evaluation. Therefore, removing the invariants allows the computational time to be allocated to the more sophisticated S4ND model. Overall, this results in the overall compute time of the S4ND model to be less than the NN and TBNN models. If one did not care about directly enforcing physical constraints through a TBNN, then the overall compute time could potentially be around the same as 1–1.5 times that of a dynamic Smagorinsky evaluation. While the S4ND model provides a step in the right direction over CNNs for scaling to more complex flows with anisotropic grids and different grid resolutions (such as channel flow), it is unable to handle stretched, curvilinear or unstructured grids natively. The S4ND architecture would be a good choice for many canonical flows in a wall-modelled context.

6. Conclusions

A new neural network subgrid stress model based on state space sequence models (S4ND) was introduced, and bandlimiting was used to increase the generalization capabilities of the model to untrained grid spacings. *A priori*, the use of bandlimiting does not restrict the expressivity of the learned representation as compared with other traditional and neural network models, while learning a continuous convolution kernel and using bandlimiting allows the neural network to extrapolate to more aggressive, untrained grid spacings. *A posteriori*, when testing on untrained grid widths, the S4ND model is more accurate than traditional models and is able to generalize where other neural network models fail on both forced HIT and channel flow LES. In addition, even when extrapolating in grid width and Reynolds number at the same time, the S4ND model is stable for both forced HIT and channel flow even when evaluated on Reynolds numbers that are over 500 000 times larger than the training set Reynolds numbers, achieving one-shot Reynolds number extrapolation capability. Overall, this suggests that the S4ND model produces more filter-consistent behaviour over a range of grid resolutions and is able to respect the scale-dependent structure of turbulence to create a more robust subgrid stress model.

Acknowledgements. This work used CPU and GPU resources via Darwin at the University of Delaware and via Bridges2 at the Pittsburgh Supercomputing Center through allocation PHY230025 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by National Science Foundation grants #2138259, #2138286, #2138307, #2137603 and #2138296. The DNS data is provided by the JHTDB. We appreciate the comments of the anonymous referees whose comments have made the manuscript better.

Funding. A.W. is partially supported by NASA Cooperative Agreement number 80NSSC22M0108 and Northrop Grumman, as well as the NDSEG Fellowship.

Declaration of interests. The authors report no conflict of interest.

https://doi.org/10.1017/jfm.2026.11929 Published online by Cambridge University Press

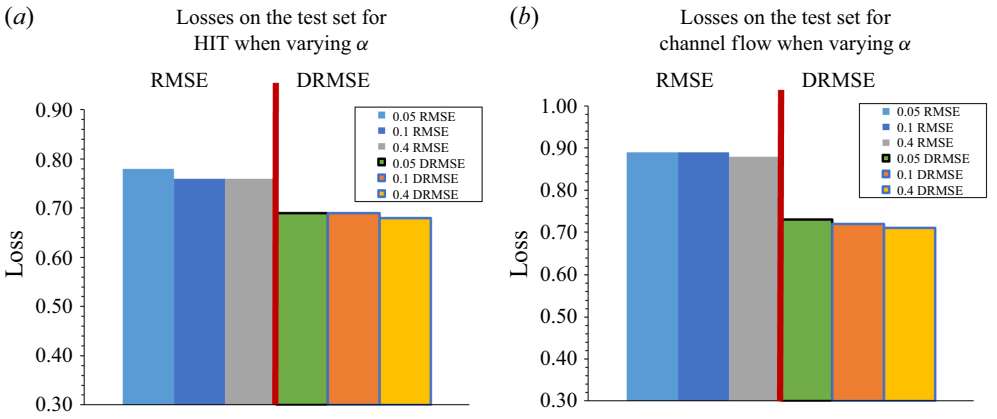


Figure 26. Bandlimit parameter sensitivity sweeps for HIT and channel flow on the hold out test set. The bandlimit value of $\alpha = 0.05$ has a slight performance degradation as compared with the other two larger values. (*a*) Losses on the test set for HIT when varying α. (*b*) Losses on the test set for channel flow when varying α.

Data availability statement. All DNS data that was filtered to produce the training data for the neural network is provided publicly on JHTDB. A public repository of the S4ND model used can be found in the references section, linking to the author's implementation.

Appendix A. The bandlimit parameter for S4ND

As seen from the main text, the interpretation of the bandlimit parameter is that it masks out frequencies in the convolution kernel that are said to be spurious. Theoretically, a bandlimit parameter of 1 corresponds to masking out all frequencies above the Nyquist wavenumber. However, due to the finite state representation and also the decay associated with the real part of a_n, the bandlimit parameter α is often set lower numerically as seen in Nguyen *et al.* (2022). Nguyen *et al.* (2022) study the effect of the bandlimit parameter and show that the accuracy of the extrapolation to different resolutions decreases quite significantly when α is set to be larger than $\alpha = 0.5$. For LES, one of the issues with *a priori* versus *a posteriori* evaluation is that the training data can be different than the neural network inputs that are seen in actual LES, partially due to the numerical method. Numerical discretization errors disproportionately contaminate the highest resolved wavenumbers near the grid cutoff (Sagaut 2006), which suggests that for robustness, one would like to use as small as a α parameter as possible that still preserves accuracy. This can be evaluated with *a priori* testing, as really low bandlimit values should decrease the expressiveness of the continuous convolution kernel, reducing its accuracy on the test set. Therefore, a parameter sweep of three $\alpha \in [0.05, 0.1, 0.4]$ is shown, with the *a priori* results on the test set seen in figure 26(*a*) to 26(*b*) for both the in-training-set HIT and channel flow results.

As seen, the results are not very sensitive to the bandlimit parameter, although there is a slight performance improvement when comparing the $\alpha = 0.05$ bandlimit cases and the $\alpha = 0.1$ bandlimit cases, where the $\alpha = 0.1$ case either does just as well or slightly better than the $\alpha = 0.05$ bandlimit cases. Meanwhile, the $\alpha = 0.4$ cases are comparable to $\alpha = 0.1$ cases. Therefore, this suggests that the bandlimit parameter of $\alpha = 0.05$ is too small as it produces too much regularization to the convolution kernel. This is seen as there is a slight performance decrease (higher errors) for this value of the bandlimit parameter. As such, $\alpha = 0.1$ is chosen as the bandlimit parameter as it is the smallest value

https://doi.org/10.1017/jfm.2026.11929 Published online by Cambridge University Press

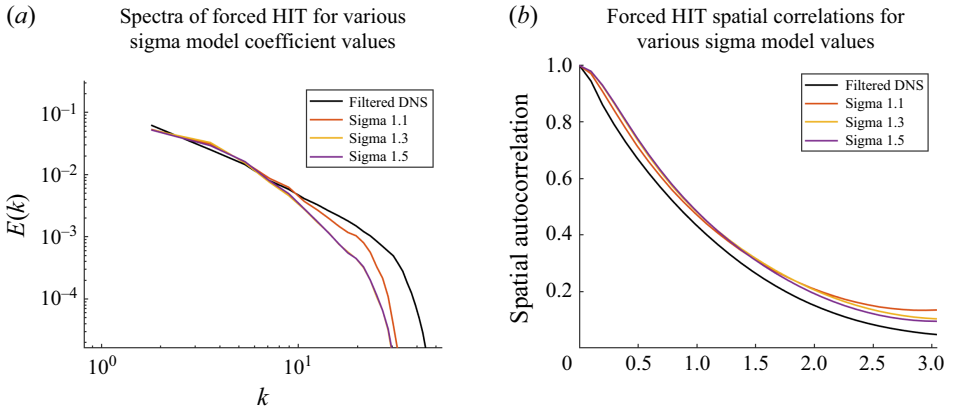


Figure 27. Sensitivity study of the sigma model to coefficients in the recommended range and also non-standard coefficients. The values in the legend correspond to the coefficient used. (*a*) Spectra of forced HIT for various sigma model coefficient values. (*b*) Forced HIT spatial correlations for various sigma model values.

(when compared with $\alpha = 0.4$) where there is no performance degradation, since $\alpha = 0.05$ has a slight performance degradation.

Appendix B. Sigma model coefficient sweep for HIT

Here, the sensitivity of the sigma model to the coefficient is investigated for coarse grid forced HIT. The sigma model is run for forced HIT using three different model coefficients. This case would correspond to the same simulation parameters as § 4.1.1. The smallest model coefficient is a non-standard model coefficient, and was used since the sigma model was found to be over-dissipative. The other two are within the standard range. In total, the three coefficients are 1.1, 1.3 and 1.5, respectively. Figure 27 shows the spectra and spatial correlation results for forced HIT at these three coefficient values.

As seen, when in the recommended range, the spectra for the sigma model with coefficients of 1.3 and 1.5 are very similar, while the non-standard coefficient of 1.1 is less dissipative. This is expected as it is outside the recommended range, and reduces the magnitude of the SGS tensor, resulting in less dissipation. While this means that it is closer to filtered DNS, there is a corresponding trade-off in the spatial correlations, as the sigma model with the non-standard coefficient performs the worst at larger spatial spacings. The spatial correlations indicate that at smaller spacings, all the model coefficients are relatively similar, but at larger grid spacings, the smaller coefficients perform worse than the larger coefficients when compared with filtered DNS. Therefore, even for coarser grids, adopting the 'middle' coefficient of 1.3 is reasonable for subgrid stress model comparisons.

References

BAE, H.J. & KOUMOUTSAKOS, P. 2022 Scientific multi-agent reinforcement learning for wall-models of turbulent flows. *Nat. Commun.* **13** (1), 1443.

BARDINA, J. 1983 *Improved Turbulence Models Based on Large Eddy Simulation of Homogeneous, Incompressible, Turbulent Flows*. Stanford University.

BECK, A., FLAD, D. & MUNZ, C.-D. 2019 Deep neural networks for data-driven LES closure models. *J. Comput. Phys.* **398**, 108910.

BECK, A. & KURZ, M. 2021 A perspective on machine learning methods in turbulence modelling. *Surv. Appl. Maths Mech.* **44** (1), e202100002.

https://doi.org/10.1017/jfm.2026.11929 Published online by Cambridge University Press

BERSELLI, L.C., ILIESCU, T. & LAYTON, W.J. 2006 *Mathematics of Large Eddy Simulation of Turbulent Flows*, 1st edn. Springer.

BOGACKI, P. & SHAMPINE, L.F. 1996 An efficient Runge–Kutta (4,5) pair. *Comput. Maths Applics* **32** (6), 15–28.

BOU-ZEID, E., MENEVEAU, C. & PARLANGE, M. 2005 A scale-dependent Lagrangian dynamic model for large eddy simulation of complex turbulent flows. *Phys. Fluids* **17** (2), 025105.

CHENG, Y. *et al.* 2022 Deep learning for subgrid-scale turbulence modeling in large-eddy simulations of the convective atmospheric boundary layer. *J. Adv. Model. Earth Syst.* **14** (5), e2021MS002847.

CHO, C., PARK, J. & CHOI, H. 2024 A recursive neural-network-based subgrid-scale model for large eddy simulation: application to homogeneous isotropic turbulence. *J. Fluid Mech.* **1000**, A76.

CLARK, R.A., FERZIGER, J.H. & REYNOLDS, W.C. 1979 Evaluation of subgrid-scale models using an accurately simulated turbulent flow. *J. Fluid Mech.* **91** (1), 1–16.

GERMANO, M., PIOMELLI, U., MOIN, P. & CABOT, W.H. 1991 A dynamic subgrid-scale eddy viscosity model. *Phys. Fluids A: Fluid Dyn.* **3** (7), 1760–1765.

GHATE, A.S. & LELE, S.K. 2017 Subfilter-scale enrichment of planetary boundary layer large eddy simulation using discrete Fourier–Gabor modes. *J. Fluid Mech.* **819**, 494–539.

GOC, K.A., LEHMKUHL, O., PARK, G.I., BOSE, S.T. & MOIN, P. 2021 Large eddy simulation of aircraft at affordable cost: a milestone in computational fluid dynamics. *Flow* **1**, E14.

GRAHAM, J. *et al.* 2016 A web services accessible database of turbulent channel flow and its use for testing a new integral wall model for LES. *J. Turbul.* **17** (2), 181–215.

GU, A., GOEL, K. & RÉ, C. 2021 Efficiently modeling long sequences with structured state spaces. arXiv preprint arXiv:2111.00396.

GUAN, Y., CHATTOPADHYAY, A., SUBEL, A. & HASSANZADEH, P. 2022 Stable a posteriori LES of 2-D turbulence using convolutional neural networks: backscattering analysis and generalization to higher *Re* via transfer learning. *J. Comput. Phys.* **458**, 111090.

KANG, M., JEON, Y. & YOU, D. 2023 Neural-network-based mixed subgrid-scale model for turbulent flow. *J. Fluid Mech.* **962**, A38.

KURZ, M., OFFENHÄUSER, P. & BECK, A. 2023 Deep reinforcement learning for turbulence modeling in large eddy simulations. *Intl J. Heat Fluid Flow* **99**, 109094.

LI, Y., PERLMAN, E., WAN, M., YANG, Y., MENEVEAU, C., BURNS, R., CHEN, S., SZALAY, A. & EYINK, G. 2008 A public turbulence database cluster and applications to study Lagrangian evolution of velocity increments in turbulence. *J. Turbul.* **9** (31), N31.

LI, Z., KOVACHKI, N., AZIZZADENESHELI, K., LIU, B., BHATTACHARYA, K., STUART, A. & ANANDKUMAR, A. 2021, Fourier neural operator for parametric partial differential equations. arXiv:2010.08895.

LING, J., KURZAWSKI, A. & TEMPLETON, J. 2016 Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *J. Fluid Mech.* **807**, 155–166.

MAEJIMAM, S. & KAWAI, S. 2024 Coarse-grid large-eddy simulation by unsupervised-learning-based sub-grid scale modeling. In *AIAA SciTech 2024 Forum*. AIAA 2024–1361. AIAA.

MENEVEAU, C., LUND, T.S. & CABOT, W.H. 1996 A Lagrangian dynamic subgrid-scale model of turbulence. *J. Fluid Mech.* **319**, 353–385.

MEYERS, J. & SAGAUT, P. 2006 On the model coefficients for the standard and the variational multi-scale Smagorinsky model. *J. Fluid Mech.* **569**, 287–319.

NGUYEN, E., GOEL, K., GU, A., DOWNS, G., SHAH, P., DAO, T., BACCUS, S. & RÉ, C. 2022 S4ND: modeling images and videos as multidimensional signals with state spaces. *Adv. Neural Inform. Proc. Syst.* **35**, 2846–2861.

NICOUD, F., TODA, H.B., CABRIT, O., BOSE, S. & LEE, J. 2011 Using singular values to build a subgrid-scale model for large eddy simulations. *Phys. Fluids* **23** (8), 085106.

PARK, J. & CHOI, H. 2021 Toward neural-network-based large eddy simulation: application to turbulent channel flow. *J. Fluid Mech.* **914**, A16.

PAWAR, S., SAN, O., RASHEED, A. & VEDULA, P. 2020 A priori analysis on deep learning of subgrid-scale parameterizations for Kraichnan turbulence. *Theor. Comput. Fluid Dyn.* **34** (4), 429–455.

PERLMAN, E., BURNS, R., LI, Y. & MENEVEAU, C. 2007 Data exploration of turbulence simulations using a database cluster. In *Supercomputing SC07, ACM*. IEEE.

POPE, S.B. 2000 *Turbulent Flows*, 1st edn. Cambridge University.

PRAKASH, P., JANSEN, K.E. & EVANS, J.A. 2022 Invariant data-driven subgrid stress modeling in the strain-rate eigenframe for large eddy simulation. *Comput. Meth. Appl. Mech. Engng* **399**, 115457.

ROBERTS, R.A. & MULLIS, C.T. 1987 *Digital Signal Processing*. Addison-Wesley Longman Publishing Co., Inc.

https://doi.org/10.1017/jfm.2026.11929 Published online by Cambridge University Press

SAGAUT, P. 2006 *Large Eddy Simulation for Incompressible Flows: An Introduction*, 3rd edn.Springer Science & Business Media.

SARGHINI, F., DE FELICE, G. & SANTINI, S. 2003 Neural networks based subgrid scale modeling in large eddy simulations. *Comput. Fluids* **32** (1), 97–108.

SCHULTZ, M.P. & FLACK, K.A. 2013 Reynolds-number scaling of turbulent channel flow. *Phys. Fluids* **25** (2), 025104.

SCOTTI, A., MENEVEAU, C. & LILLY, D.K. 1993 Generalized Smagorinsky model for anisotropic grids. *Phys. Fluids* **5**, 2306–2308.

SIRIGNANO, J. & MACART, J.F. 2023 Deep learning closure models for large-eddy simulation of flows around bluff bodies. *J. Fluid Mech.* **966**, A26.

SMAGORINSKY, J. 1963 General circulation experiments with the primitive equations: I. the basic experiment. *Mon. Weath. Rev.* **91** (3), 99–164.

STALLCUP, E.W., KSHITIJ, A. & DAHM, W.J. 2022 Adaptive scale-similar closure for large eddy simulations. Part 1: subgrid stress closure. In *AIAA SciTech*. AIAA. AIAA 2022-0595.

STOFFER, R., LEEUWEN, C.M., PODAREANU, D., CODREANU, V., VEERMAN, M.A., JANSSENS, M., HARTOGENSIS, O.K. & HEERWAARDEN, C.C. 2021 Development of a large-eddy simulation subgrid model based on artificial neural networks: a case study of turbulent channel flow. *Geosci. Model Develop.* **14** (6), 3769–3788.

STOLZ, S., ADAMS, N.A. & KLEISER, L. 2001 An approximate deconvolution model for large-eddy simulation with application to incompressible wall-bounded flows. *Phys. Fluids* **13** (4), 997–1015.

VREMAN, A.W. 2004 An eddy-viscosity subgrid-scale model for turbulent shear flow: algebraic theory and applications. *Phys. Fluids* **16** (10), 3670–3681.

WANG, Z., LUO, K., LI, D., TAN, J. & FAN, J. 2018 Investigations of data-driven closure for subgrid-scale stress in large-eddy simulation. *Phys. Fluids* **30** (12), 125101.

WU, A. 2025 S4ND_SGS_Model. Available at: https://github.com/awu1018/S4ND_SGS_Model/tree/main.

WU, A. & LELE, S.K. 2025 Two neural network Unet architecture for subfilter stress modeling. *Phys. Rev. Fluids* **10** (1), 014601.

XIE, C., WANG, J. & WEINAN, E. 2020*a* Modeling subgrid-scale forces by spatial artificial neural networks in large eddy simulation of turbulence. *Phys. Rev. Fluids* **5** (5), 054606.

XIE, C., YUAN, Z. & WANG, J. 2020*b* Artificial neural network-based nonlinear algebraic models for large eddy simulation of turbulence. *Phys. Fluids* **32** (11), 115101.

YUAN, Z., WANG, Y., XIE, C. & WANG, J. 2021 Dynamic iterative approximate deconvolution models for large-eddy simulation of turbulence. *Phys. Fluids* **33** (8), 085125.

YUAN, Z., XIE, C. & WANG, J. 2020 Deconvolutional artificial neural network models for large eddy simulation of turbulence. *Phys. Fluids* **32** (11), 115106.

https://doi.org/10.1017/jfm.2026.11929 Published online by Cambridge University Press