
EddyFormer: Accelerated Neural Simulations of Three-Dimensional Turbulence at Scale

Yiheng Du

UC Berkeley

yihengdu@berkeley.edu

Aditi S. Krishnapriyan

UC Berkeley, LBNL

aditik1@berkeley.edu

Abstract

Computationally resolving turbulence remains a central challenge in fluid dynamics due to its multi-scale interactions. Fully resolving large-scale turbulence through direct numerical simulation (DNS) is computationally prohibitive, motivating data-driven machine learning alternatives. In this work, we propose *EddyFormer*, a Transformer-based spectral-element (SEM) architecture for large-scale turbulence simulation that combines the accuracy of spectral methods with the scalability of the attention mechanism. We introduce an *SEM tokenization* that decomposes the flow into grid-scale and subgrid-scale components, enabling capture of both local and global features. We create a new three-dimensional isotropic turbulence dataset and train EddyFormer to achieve DNS-level accuracy at 256^3 resolution, providing a 30x speedup over DNS. When applied to unseen domains up to 4x larger than in training, EddyFormer preserves accuracy on physics-invariant metrics—energy spectra, correlation functions, and structure functions—showing domain generalization. On *The Well* benchmark suite of diverse turbulent flows, EddyFormer resolves cases where prior ML models fail to converge, accurately reproducing complex dynamics across a wide range of physical conditions. Code and project page are available at: <https://mrlazy1708.github.io/eddyformer>.

1 Introduction

Numerical simulation of fluid systems is essential for understanding and predicting complex flow phenomena [Van Dyke, 1982] in engineering and the natural sciences. Many critical applications, such as aerodynamics and weather forecast, rely on computational fluid dynamics (CFD) to simulate these flows. However, capturing flow patterns in turbulent flows [Pope, 2001] remains a fundamental challenge. In turbulence, *eddies*—coherent swirling structures—continuously interact and transfer energy across a wide range of length scales. The extent of these scales is determined by the Reynolds number, Re , a dimensionless quantity which represents the ratio of inertial to viscous forces. Fully resolving a turbulent flow requires $Re^{9/4}$ resolution, making DNS prohibitively expensive at high Reynolds number [Moin and Mahesh, 1998], even with supercomputers [Yeung et al., 2012, 2015].

Turbulence exhibits a characteristic cascade: energy is injected at large scales, transferred through the inertial range, and dissipated at the smallest scales where viscosity dominates. Statistical descriptions of turbulence assumes that the smallest scales become independent of the large-scale flow structures, which is known as the Kolmogorov’s similarity hypothesis [Pope, 2001]:

At sufficiently high Reynolds number, the statistics of small-scale motions have a universal form that is uniquely determined by viscosity and energy dissipation rate.

Based on this principle, large-eddy simulation [Smagorinsky, 1963] (LES) resolves only the large-scale structures while using theoretical models on smaller, subgrid scales (SGS). By filtering the governing equations, LES captures dominant flow features while relying on “closure” models to approximate unresolved dynamics at SGS. Despite its successes, LES faces challenges in wall-bounded and anisotropic turbulence [Piomelli, 1999], particularly in balancing accuracy and efficiency.

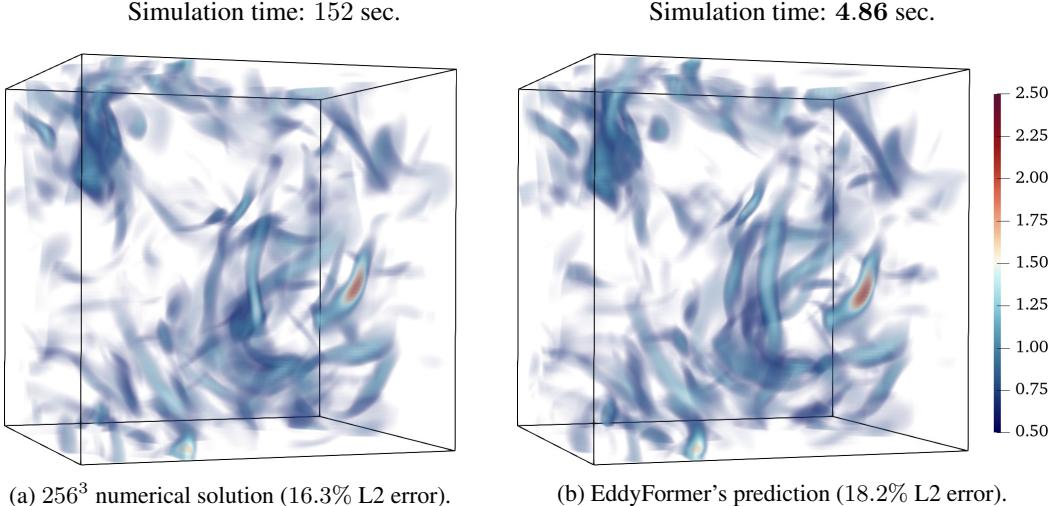


Figure 1: Vorticity magnitude of forced turbulence with Reynolds no. $Re \approx 94$ at $t = 5$. EddyFormer reproduces the accuracy level and invariant flow statistics of a direct numerical simulation (DNS) at 256^3 resolution, computed with highly efficient pseudo-spectral schemes [Canuto, 2007]. The reported L2 error is measured against a reference DNS at 384^3 resolution. When benchmarked on a single NVIDIA A100 GPU, EddyFormer is 30x faster, achieving essentially real-time simulations.

Recent trends in ML have explored data-driven turbulence simulation [Li et al., 2020, Kochkov et al., 2021], with spectral-based neural network surrogates showing promising accuracy. Similar to spectral methods [Canuto, 2007], they approximate the solution using spectral bases and parameterize the dynamics through spectral transformations. For example, Fourier neural operators (FNOs) [Li et al., 2020] provide an expressive model for learning mappings on structured grids. However, scaling them to large problems with a wide range of fine-scale eddy dynamics remains challenging, as parameterizing large Fourier kernels is inefficient [Qin et al., 2024]. Likewise, Transformer architectures [Vaswani et al., 2017] face certain obstacles in fluid systems, as the quadratic growth of attention with mesh resolution may constrain their scalability in practice. These limitations suggest exploring architectures that balance efficiency with scalability to handle large-scale turbulence.

Our contributions. We propose *EddyFormer*, a machine learning model for large-scale fluid flow simulation that captures multi-scale interactions through learned representations of the flow. The input is tokenized using the spectral/hp element method (SEM) [Karniadakis and Sherwin, 2005], which partitions the domain into coarse elements with spectral expansions in each subdomain. This tokenization combines the strengths of spectral modeling and scalability: local spectral expansions provide highly expressive representations, while each coarse element, i.e., a *SEM token*, yields a compact sequence that makes the attention mechanism efficient for capturing long-range interactions.

EddyFormer consists of two complementary streams: an *SGS stream* (§3.1), which captures localized eddy interactions; and an *LES stream* (§3.2), which models global, coherent structures. The LES stream is parameterized using multi-head attention applied to the filtered SEM tokens, significantly reducing computational complexity compared to self-attention on the full mesh. For the SGS stream, the Kolmogorov hypothesis suggests that small-scale dynamics can be modeled universally. We use spectral convolutions with a localized kernel to model the subgrid-scale dynamics.

EddyFormer demonstrates strong performance across a wide range of fluid flow problems, outperforming state-of-the-art ML models, as well as efficient numerical solvers at the same level of accuracy. For three-dimensional isotropic turbulence (§4.1, Fig. 1), EddyFormer achieves the accuracy of direct numerical simulation (DNS) at 256^3 resolution while running 30 \times faster, reducing error by 30% compared to neural operator and Transformer baselines. When trained on two-dimensional turbulence (§4.2), it generalizes to domains up to four times larger than those seen in training by applying attention masking at test time, suggesting it learns domain-independent flow dynamics that can be adapted to general domain geometries. Benchmarks on other turbulent flows from The Well Ohana et al. [2025] (§4.3) show that EddyFormer accurately resolves turbulence driven by a wide range of physical conditions, including cases where other ML baselines fail to converge.

Related works. We briefly review ML approaches for turbulence here, with a more comprehensive survey of Transformer-based models and application domains presented in §A. Early attempts focused on learning data-driven turbulence closures for Reynolds-averaged Navier-Stokes (RANS)[Tracey et al., 2015, Ling et al., 2016] and large-eddy simulation (LES) [Beck et al., 2018, Font et al., 2021, List et al., 2022]. Wang et al. [2020] adopted the LES decomposition and modeled the mean and fluctuating fields separately. More recent works instead focus on direct simulation of turbulent flows using advanced neural networks architectures [Li et al., 2020, Tran et al., 2021, Wen et al., 2022, Kossaifi et al., 2023], showing promise but not yet consistently matching the accuracy of numerical solvers [Spalart, 2023, McGreivy and Hakim, 2024]. A promising alternative, “learned correction” [Kochkov et al., 2021, Shankar et al., 2025], uses low-accuracy solvers to improve accuracy. However, most previous works remain limited to small-scale two-dimensional flows.

2 Background

The Navier-Stokes (NS) equations [Kundu et al., 2024] describe the conservation of mass, momentum, and energy in fluid systems, capturing flows in a wide range of physical conditions. We focus on the incompressible flow described by the Boussinesq approximation for negligible density variations:

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} &= \nu \nabla^2 \mathbf{u} - \frac{1}{\rho_0} \nabla p, \\ \nabla \cdot \mathbf{u} &= 0, \end{aligned} \quad (1)$$

where $\mathbf{u}(t; x, y, z) \in \mathbb{R}^3$ is the velocity field on a domain Ω , ρ is the fluid density, ∇p is the pressure gradient, and ν is the kinematic viscosity. Given the initial velocity field $\mathbf{u}(0; \cdot)$ and certain boundary conditions on \mathbf{u} and p , the task is to simulate the field $\mathbf{u}(t; \cdot)$ at a later time step t .

Large-eddy simulation. We adopt the methodology of large-eddy simulation (LES) in our model architecture. Classical LES aims to directly simulate the large-scale unsteady turbulent motions, while modeling the effects of small motions in theory. Given a (spatial) filter $G(\cdot, \cdot)$ on Ω , the velocity field $\mathbf{u}(t; \cdot)$ is decomposed into its filtered field $\bar{\mathbf{u}}(t; \cdot) := G * \mathbf{u}(t; \cdot)$, and the corresponding subgrid-scale field (SGS) $\mathbf{u}' = \mathbf{u} - \bar{\mathbf{u}}$. The filtered velocity field $\bar{\mathbf{u}}$ is governed by the filtered momentum equation,

$$\frac{\partial \bar{\mathbf{u}}}{\partial t} + \bar{\mathbf{u}} \cdot \nabla \bar{\mathbf{u}} = \nu \nabla^2 \bar{\mathbf{u}} - \nabla \cdot \tau - \frac{1}{\rho_0} \nabla \bar{p}, \quad (2)$$

where $\tau_{ij} := \bar{u}_i \bar{u}_j - \bar{u}_i \bar{u}_j$ is the residual stress (or SGS) tensor. This filtered equation is unclosed, in terms of the SGS tensor being approximated based on the filter G and the filtered field $\bar{\mathbf{u}}$. Many classical turbulence models express τ as an eddy viscosity $\tau_{ij} \approx -2\nu_t \bar{S}_{ij}$, where \bar{S} is the resolved strain rate tensor and ν_t is the eddy viscosity to be modeled by the corresponding SGS model.

Spectral element method. We use the standard spectral element method (SEM) to parameterize the solution field $\mathbf{u}(t; \cdot)$. The physical domain Ω is discretized into H structured elements $\Omega = \cup_{h \leq H} \Omega_h$, where each element Ω_h is mapped to an unit cube through a geometric transformation. We assume a uniform mesh with cell size Δ and denote the local coordinates on Ω_h as $0 \leq \xi_h^x, \xi_h^y, \xi_h^z \leq 1$.

The local field $\mathbf{u}_h(t; \xi_h)$ on Ω_h is expanded using higher-order bases. Let $\{\phi_m\}_m$ be a set of one-dimensional basis on $[0, 1]$, their full tensor product forms a multi-dimensional basis set $\Phi_p(\xi) := \Pi_i \phi_{p_i}(\xi_i)$. The local field $\mathbf{u}_h(t; \cdot)$ is then approximated as,

$$\mathbf{u}_h(t; \xi_h) = \sum_{|p| \leq M} \tilde{\mathbf{u}}_{hp}^t \Phi_p(\xi_h), \quad (3)$$

where $\tilde{\mathbf{u}}_h^t \in \mathbb{R}^{3M^3}$ are the expansion coefficients of the h 'th element. In contrast to spectral methods, the standard spectral basis must be modified to enforce a certain order of continuity across the element boundaries. In this work, we consider C^0 -continuous boundary-interior decomposition. Given a set of orthogonal polynomials¹ $\{p_m\}_m$, their corresponding modal p -type basis is defined as,

$$\phi_m(\xi) = \begin{cases} 1 - \xi & m = 0, \\ (\xi - \xi^2)p_{m-1}(\xi) & 0 < m < M, \\ \xi & m = M, \end{cases} \quad (4)$$

where ϕ_0 and ϕ_M are the shared left and right boundary modes, respectively.

¹In this work, we implement the (shifted) Chebyshev polynomials $T_m(x) = \cos(m \arccos(2x - 1))$, and the Legendre polynomials $P_m(x) = \frac{1}{2^m m!} \frac{d^m}{dx^m} (x^2 - 2x)^m$, as the orthogonal polynomial set p_m .

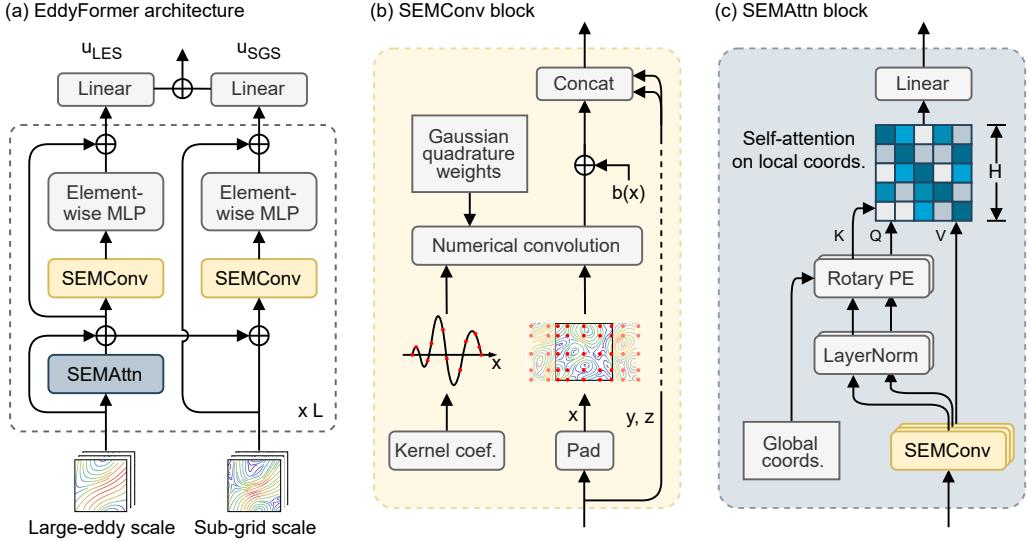


Figure 2: Our proposed architecture: *EddyFormer*. (a) The PDE initial conditions are interpolated using spectral element methods (SEM). The large-eddy simulation (LES) field and subgrid-scale (SGS) field are transformed through the LES stream (§3.2) and SGS stream (§3.1), respectively. The SGS stream consists of (b) SEM-based convolution blocks to model local eddy dynamics, while the LES stream consists of (c) SEM-based attention blocks to capture long-range dependencies.

3 Model design choices for EddyFormer

We introduce the EddyFormer architecture (Fig. 2). EddyFormer parameterizes the solution field using SEM bases defined in Eqn. (3). For a fixed domain discretization $\Omega = \cup_h \Omega_h$ with $H = N^3$ elements and $P = M^3$ modes per element, EddyFormer maps the spectral representation of the initial condition \mathbf{u}_0 to the prediction $\mathbf{u}(t; \cdot)$ at a fixed time t , based on a chosen SEM basis Φ :

$$\mathbf{u}(0; \cdot) = \sum_{|p| \leq M} \tilde{\mathbf{u}}_{hp}^0 \Phi_p \mapsto \sum_{|p| \leq M} \tilde{\mathbf{u}}_{hp}^t \Phi_p = \mathbf{u}(t; \cdot) \quad \text{on } \Omega_h, \quad (5)$$

where $\tilde{\mathbf{u}}^t \in \mathbb{R}^{3N^3 M^3}$ is the output of the model.

EddyFormer decomposes the output as the sum of an LES-filtered part \mathbf{u}_{LES} and the residual field \mathbf{u}_{SGS} . The residual coefficients $\tilde{\mathbf{u}}_{\text{SGS}}$ are unfiltered and parameterized by an *SGS stream* (§3.1). For the LES filter, we simply use a spectral cutoff filter with $k_{\max} < M$ modes on each axis, parameterizing the LES-filtered coefficients $\tilde{\mathbf{u}}_{\text{LES}} \in \mathbb{R}^{3N^3 k_{\max}^3}$ by an *LES stream* (§3.2). In §C.2.1, we perform an ablation study on the LES-SGS splitting with different k_{\max} to verify its effectiveness.

$$\left. \begin{array}{l} \text{SGS stream: } \mathbf{u}^{(0)} \rightarrow \mathbf{u}^{(1)} \rightarrow \dots \rightarrow \mathbf{u}^{(L)} \rightarrow \mathbf{u}_{\text{SGS}} \\ \text{LES stream: } \bar{\mathbf{u}}^{(0)} \rightarrow \bar{\mathbf{u}}^{(1)} \rightarrow \dots \rightarrow \bar{\mathbf{u}}^{(L)} \rightarrow \mathbf{u}_{\text{LES}} \end{array} \right\} \rightarrow \mathbf{u} = \mathbf{u}_{\text{LES}} + \mathbf{u}_{\text{SGS}}. \quad (6)$$

3.1 SGS stream for local motions

The input is first interpolated onto the chosen SEM basis Φ , and then projected to the initial feature, $\mathbf{u}^{(0)} = W_{\text{in}} \mathbf{u}(0; \cdot)$, and the output is the projection of the final feature, $\mathbf{u}_{\text{SGS}} = W_{\text{SGS}} \mathbf{u}^{(L)}$. In each layer l , the locals fields $\mathbf{u}_h^{(l)}$ receive information from the LES stream and then perform convolutions:

$$\mathbf{u}_h^{(l+1)} = \mathbf{u}_h^{(l)} + \text{FFN}(\text{SEMConv}_h(\mathbf{u}_h^{(l)} + \epsilon \bar{\mathbf{u}}_h^{(l)})), \quad (7)$$

where FFN is a two-layer feed-forward network applied on the collocation points independently, and ϵ is a learnable scalar. SEMConv is our SEM-based convolution block, as introduced below.

SEM-based convolution. We extend the powerful spectral convolution [Li et al., 2020] as the basic block in our architecture. The SEMConv block parameterizes convolution in SEM bases. For a given

field \mathbf{u} with d components, we parameterize its convolution with a compact kernel \mathbf{k} that is supported on a window of size s ,

$$\text{SEMConv}(\mathbf{u})(\mathbf{x}') = \int_{|\mathbf{x}| \leq s/2} \mathbf{k}(\mathbf{x}) \mathbf{u}(\mathbf{x}' - \mathbf{x})^T d\mathbf{x}. \quad (8)$$

Note that $s > 0$ is a hyperparameter to be determined. We found that a small value of s (compared to the domain length-scale) is usually expressive enough. Due to the success of Fourier factorization techniques [Tran et al., 2021], we also restrict the convolution kernel to axial form,

$$\mathbf{k}(x, y, z) = \sum_m \tilde{\mathbf{k}}_m [e^{i\pi mx/s} e^{i\pi my/s} e^{i\pi mz/s}]^T, \quad (9)$$

where $\tilde{\mathbf{k}}_m \in \mathbb{C}^{d \times d \times 3}$ are the learnable Fourier coefficients. The convolution is computed by directly evaluating Eqn. (8) using numerical quadrature rules (see Algorithm 1). Since the number of modes per element is small compared to the total number of modes, the computational cost of SEMConv remains comparable to that of FFT-based convolution with global kernels (see Tab. 1 for comparison).

Inclusion of LES features. The inclusion of LES features in the SGS stream, Eqn. (7), is justified by the turbulence cascade. Large eddies transport the turbulent kinetic energy downwards, while smaller eddies primarily dissipate that energy. Classical closures—such as Smagorinsky [1963] and Spalart and Allmaras [1992]—reflect this hierarchy by parameterizing subgrid stress τ in Eqn. (2) using the resolved large-scale flow field. Similarly, we also find that the inclusion of LES features into the SGS features is crucial for EddyFormer’s performance, but not vice versa.

3.2 LES stream for global motions

The initial LES feature $\bar{\mathbf{u}}^{(0)}$ is spectrally truncated to k_{\max} modes, and the output is the projection of the final feature, $\mathbf{u}_{\text{LES}} = W_{\text{LES}} \bar{\mathbf{u}}^{(L)}$. The local fields $\bar{\mathbf{u}}_h^{(l)}$ are regarded as *SEM tokens*. In each layer l , the SEM tokens are globally mixed with other tokens first, and then locally transformed. Specifically,

$$\begin{aligned} \bar{\mathbf{u}}_h^{(l)} &\leftarrow \bar{\mathbf{u}}_h^{(l)} + \text{SEMAttn}_h(\bar{\mathbf{u}}^{(l)}), \\ \bar{\mathbf{u}}_h^{(l+1)} &= \bar{\mathbf{u}}_h^{(l)} + \text{FFN}(\text{SEMConv}_h(\bar{\mathbf{u}}^{(l)})), \end{aligned} \quad (10)$$

where SEMAttn is our SEM-based self-attention block, which we introduce below.

SEM-based attention. We utilize the attention mechanism to model interactions between SEM tokens. In SEMAttn, the SEM tokens $\bar{\mathbf{u}}_h^{(l)}$ are mixed based on their *local* coordinates. Specifically, these features are projected using convolutions, $\bar{\mathbf{u}}^{Q/K/V} = \text{SEMConv}^{Q/K/V}(\bar{\mathbf{u}}^{(l)})$, and the SEMAttn is defined by mixing all features on the same *local* coordinate ξ using self-attention mechanism,

$$\text{SEMAttn}_h(\bar{\mathbf{u}}^{(l)})(\xi) = \sum_{h' \leq H} \text{SoftMax}_{h'}(\bar{\mathbf{u}}_h^Q(\xi) \cdot \bar{\mathbf{u}}_{h'}^K(\xi)) \bar{\mathbf{u}}_{h'}^V(\xi), \quad (11)$$

which is implemented by performing self-attention on each collocation point independently. Note that SEMAttn strictly preserves the continuity of the input features, regardless of the discretization scheme of SEM representation. Similar to standard Transformers, we also add learnable bias terms, apply layer normalization², and perform position encoding to the key and query features.

In contrast to mesh-based Transformers, the sequence length, or the number of SEM tokens, equals the number of spectral elements N^3 , which is orders of magnitude smaller than the number of resolved modes $N^3 M^3$. This avoids the unfavorable quadratic scaling of standard attention mechanism.

Rotary position encoding. While vision Transformers [Dosovitskiy et al., 2020] are insensitive to position encoding, we find that turbulence simulation relies on substantial positional information in SEMAttn. Inspired by Su et al. [2024], we rotate the key and query fields based on their *global* coordinates, namely, for the k 'th channel, the rotation along x axis is defined by,

$$\begin{aligned} \bar{\mathbf{u}}_{h,2k}^{Q/K}(x, y, z) &\leftarrow \cos(kx) \bar{\mathbf{u}}_{h,2k}^{Q/K}(x, y, z) - \sin(kx) \bar{\mathbf{u}}_{h,2k+1}^{Q/K}(x, y, z), \\ \bar{\mathbf{u}}_{h,2k+1}^{Q/K}(x, y, z) &\leftarrow \sin(kx) \bar{\mathbf{u}}_{h,2k}^{Q/K}(x, y, z) + \cos(kx) \bar{\mathbf{u}}_{h,2k+1}^{Q/K}(x, y, z). \end{aligned} \quad (12)$$

The rotation along y and z axes are similar, and the final results are concatenated together. Due to the properties of rotation, it’s apparent that the SEMAttn layer is also strictly translational invariant.

²We don’t apply normalization on any other hidden features, since it was found to hurt model performance. In order to regularize gradients, we initialize all convolution kernels with a random variable of magnitude 10^{-7} .

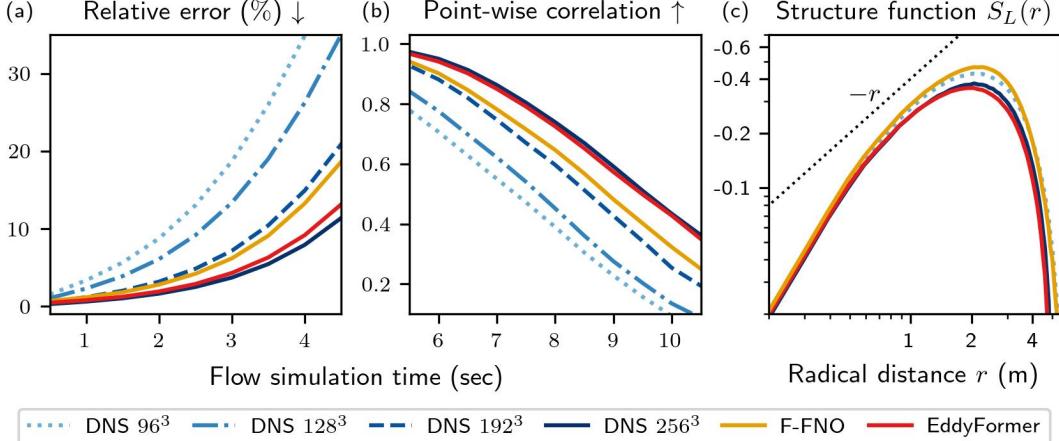


Figure 3: Performance of models and DNS on Re94 (§4.1), a 3D homogeneous isotropic turbulence fluid flow. The models learn to correct DNS at 96^3 resolution, and we report evaluation metrics relative to a reference DNS at 384^3 resolution. EddyFormer achieves the accuracy of DNS at 256^3 resolution in terms of (a) relative L2 error for $t \leq 5$ and (b) correlation for $5 \leq t \leq 10$ which measures field similarities; for (c) longer prediction rollouts up to 20 seconds, EddyFormer also captures the third-order structure function, Eqn. (18), which measures the velocity skewness and indicates the energy transfer across different scales. In contrast, the baseline F-FNO shows higher rollout error and a mismatched structure function.

4 Experiments

The JAX [Bradbury et al., 2018] implementation of EddyFormer is available at <https://github.com/ASK-Berkeley/EddyFormer>, and a PyTorch [Paszke et al., 2019] version is provided via NVIDIA PhysicsNeMo (<https://github.com/NVIDIA/physicsnemo>). In §4.1 and §4.2, we consider two- and three-dimensional turbulent flows that are maintained by an artificial forcing. We evaluate the models using a set of comprehensive metrics (see §B for details). In §4.3, we benchmark EddyFormer on The Well [Ohana et al., 2025], a large-scale fluid simulation dataset under diverse flow conditions.

4.1 Homogeneous isotropic turbulence

We consider three-dimensional homogeneous isotropic turbulence, Re94. We use an isotropic forcing scheme on the lowest Fourier modes, which has been extensively used in turbulence modeling [Langford and Moser, 1999, Bazilevs et al., 2007, Yeung et al., 2015]. Specifically, a constant power forcing $\mathbf{f}(\mathbf{x})$ is applied on the lowest velocity modes $|\mathbf{k}| \leq 1$, in order to maintain the turbulence:

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = \nu \nabla^2 \mathbf{u} + \mathbf{f}(\mathbf{x}), \quad \mathbf{f}(\mathbf{x}) = \sum_{|\mathbf{k}| \leq 1, \mathbf{k} \neq 0} \frac{P_{\text{in}}}{E_1} \hat{\mathbf{u}}_{\mathbf{k}} e^{i\mathbf{k} \cdot \mathbf{x}}, \quad (13)$$

where $\nu = 0.01$ is the kinematic viscosity, $P_{\text{in}} = 1.0$ is the input power, and $E_1 = \frac{1}{2} \sum_{|\mathbf{k}| \leq 1} \hat{\mathbf{u}}_{\mathbf{k}} \cdot \hat{\mathbf{u}}_{\mathbf{k}}$ is the kinetic energy in lowest forced modes where $|\mathbf{k}| \leq 1$. The task is to predict the velocity field \mathbf{u} after $\Delta t = 0.5$. This flow (see Fig. 1 for vorticity rendering) at equilibrium has Reynolds number $Re \approx 94$ at Taylor microscale. The dataset is recorded at 96^4 resolution. We generate the dataset using a high-accuracy pseudo-spectral direct numerical simulation (DNS) at 384^3 resolution, which corresponds to a Kolmogorov scale of 0.5η . Detailed dataset and flow statistics can be found in §B.

For EddyFormer, we use 8^3 elements and 13^3 modes per element, resulting in approximately $\sim 0.1\%$ interpolation error on Re94. Detailed model architecture is described in Tab. 7. The LES filter size is set to $k_{\max} = 5$ in order to balance accuracy and efficiency (see §C.2.1 for ablations on k_{\max}).

Direct prediction (“pure ML”) benchmark. We first benchmark EddyFormer in a purely learned setting, where the task is to directly predict the one-step velocity field \mathbf{u} without using any coarse initial guess from a numerical solver. Baseline models include the classic ResNet [He et al., 2016], TF-Net [Wang et al., 2020] which performs filter-based splitting, the popular neural operators (FNO [Li et al., 2020] and F-FNO Tran et al. [2021]), and competitive Transformers for partial differential equations (GNOT [Hao et al., 2023] and AViT [McCabe et al., 2023]).

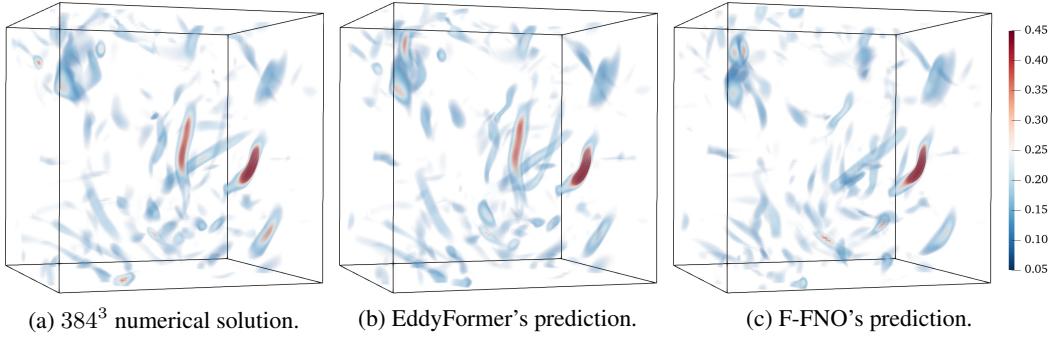


Figure 4: Visualization of the Q-criterion for a Re94 (§4.1) test sample at $t = 5$. The Q-criterion, Eqn. (19), identifies vortex structures where rotational motion dominates over strain, with the red volume in (a) reference DNS solution highlighting the vortex cores; (b) EddyFormer successfully captures both vortex cores, while (c) F-FNO fails to resolve the vortex at the center of the domain.

Every model is trained for 15k steps using the time-averaged one-step relative error,

$$L_\theta = \mathbb{E}_{t, \mathbf{u}_0} [\|\mathbf{u}_{t+\Delta t} - f_\theta(\mathbf{u}_t)\| / \|\mathbf{u}_{t+\Delta t}\|], \quad (14)$$

where f_θ is a neural network parameterized by θ . Training details and loss curves for different models can be found in §C.2. Tab. 1 compares the one-step error and the training speed of each model. EddyFormer achieves lower test errors, combined with a compact model size and competitive training speed, outperforming all other baseline models.

Table 1: Direct prediction on Re94 : model size, training time per iteration, and one-step relative error. EF (cheb) and EF (leg) stands for our model EddyFormer with Chebyshev and Legendre basis.

	ResNet	TF-Net	FNO	F-FNO	GNOT	AViT	EF (cheb)	EF (leg)
# of params.	2.3M	6.3M	17.6M	1.7M	6.2M	6.0M	2.3M	2.3M
Train speed	1.2s/it	3.0s/it	1.5s/it	1.5s/it	7.4s/it	1.0s/it	2.7s/it	1.8s/it
Test error	87.2%	27.7%	23.1%	22.5%	69.7%	26.6%	8.61%	8.52%

Learned correction from a coarse numerical solver. Next, we use the learned correction (“hybrid model”) approach [Kochkov et al., 2021], which uses the output of a low-resolution DNS as an efficient initial solution and trains the model to predict the missing fine-scale information. This provides a good starting point for ML. The model additionally receives the output, $\mathbf{u}_{t+\Delta t}^*$, of a coarse solver, and predicts its error with respect to the high-resolution reference data $\mathbf{u}_{t+\Delta t}$. Similarly, the loss function is also defined as the time-averaged relative L2 error,

$$L_\theta = \mathbb{E}_{t, \mathbf{u}_0} \|\mathbf{u}_{t+\Delta t} - \text{LC}_\theta(\mathbf{u}_t, \mathbf{u}_{t+\Delta t}^*)\| / \|\mathbf{u}_{t+\Delta t}\|, \quad (15)$$

where $\text{LC}_\theta(\mathbf{u}, \mathbf{u}^*) = \mathbf{u}^* + \alpha f_\theta(\mathbf{u}, \mathbf{u}^*)$ is the correction on the coarse solution \mathbf{u}^* . Here, we choose $\alpha = 10^{-5}$, which corresponds to an a priori estimate of the coarse solver’s error. In practice, we find that model performance is not very sensitive to the α value.

For baseline comparison, we only consider F-FNO since it was the best baseline in the direct prediction benchmark. All models learn to correct DNS at 96^3 resolution, which under-resolves the Kolmogorov scale η by a ratio of 2. Each model is trained for 15k steps on four NVIDIA A100 GPUs, taking approximately one day wall-clock time to converge.

Results. Fig.3 shows the errors and invariant measures for both models, alongside DNS results at different resolutions. On the test set, EddyFormer matches the accuracy of DNS at 256^3 resolution both the within trained window (Fig. 3a) and at later time steps (Fig. 3b), while accurately capturing the invariant structure functions over longer rollouts (Fig. 3c). EddyFormer also successfully predicts all characteristic vortices (Fig. 4), preserving both the shape and spatial coherence of highly rotational eddies.

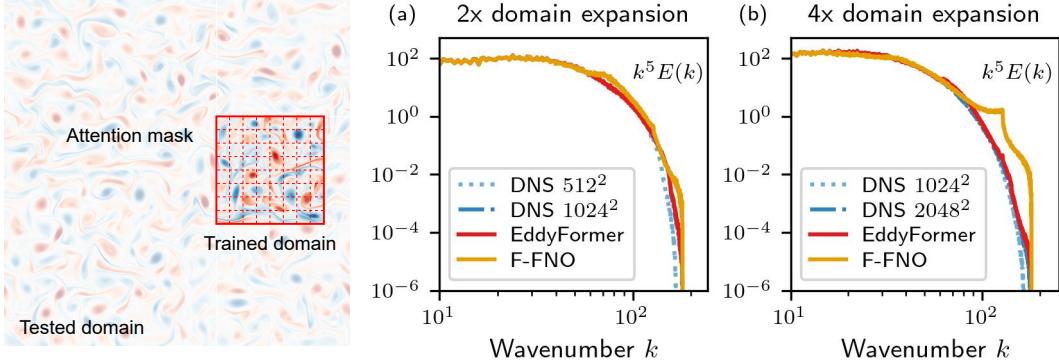


Figure 5: **(Left)** Illustration of how EddyFormer generalizes to larger domains. Trained on a smaller domain, EddyFormer predicts the flow field on a larger domain by applying an attention mask that ensures consistency with the training setup. The context window of each SEM token is constrained to a square of length 2π , maintaining consistency between training and test times. **(Right)** Model evaluation on both 2x and 4x larger domains of KF4 (§4.2). The scaled energy spectra, Eqn. (17), of the EddyFormer’s predictions align closely with the reference high-resolution DNS, while F-FNO does not capture this and shows deviation in high-frequency modes.

In addition to its accuracy, EddyFormer offers computational benefit. At test time, it only adds $\sim 50\%$ extra computational cost to the base DNS at 96^3 resolution, accelerating the simulation by 30-fold on a single GPU (including the cost of the coarse DNS, as shown in Fig. 1), and is comparable to a high-accuracy DNS at 256^3 resolution. This speed-up allows for almost real-time high-accuracy simulations on Re94, enabling long-time ensemble rollouts that would otherwise be computationally expensive with high-resolution DNS.

4.2 Kolmogorov flow

We also consider the “Kolmogorov” flow KF4, another very popular model for studying turbulence in two dimensions. We use the scalar vorticity form of the Navier-Stokes equation:

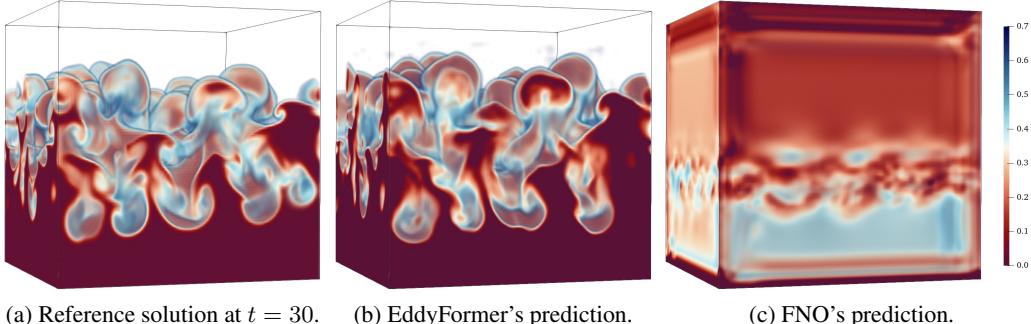
$$\frac{\partial \omega}{\partial t} + \mathbf{u} \cdot \nabla \omega = \nu \nabla^2 \omega + k \cos(kx) - 0.1\omega, \quad (16)$$

where ω is the vorticity on z -axis. The energy is injected by a sinusoidal forcing $\mathbf{f}(x, y) = \sin(kx)\hat{\mathbf{e}}_y$ on periodic domain of length 2π . Following Kochkov et al. [2021], we set the forcing mode $k = 4$, the viscosity $\nu = 10^{-3}$, and add a damping term $-0.1\mathbf{u}$ to prevent energy accumulation at the lowest modes [Chertkov et al., 2007]. The task is to predict the vorticity field ω after $\Delta t = 1$.

We set the forcing mode $k = 4$ and use the vorticity form of the equation. The resulting flow at equilibrium has a Reynolds number $Re \approx 800$ at Taylor microscale. The dataset is generated using a 4096^2 pseudo-spectral DNS, which is more accurate than Kochkov et al. [2021] and Dresdner et al. [2022]. See §B for dataset details and simulation statistics.

All models are trained to correct DNS at a 256^2 resolution. To control the propagation of errors over time, we incorporate a 5-step roll-out training approach, which minimizes accumulated error by differentiating through the DNS steps [Kochkov et al., 2021]. The roll-out loss function is applied during the fine-tuning stage, after the one-step loss has converged. See §C.1 for training details.

Results. In Fig. 8a and 8b, EddyFormer matches the accuracy of DNS at resolutions between 512^2 and 1024^2 , both within the training window and beyond, in terms of relative error and correlations with the reference DNS. Over long rollouts (Fig. 8c), EddyFormer also accurately captures the invariant structure function whereas the baseline F-FNO shows significant deviation. Tab. 10 reports the kinetic energy, Eqn. (23), showing that EddyFormer conserves energy over extended time horizons.



(a) Reference solution at $t = 30$. (b) EddyFormer’s prediction. (c) FNO’s prediction.

Figure 6: Density of a Rayleigh-Taylor instability test case. EddyFormer successfully predicts the structure of plumes due to buoyancy rise up to a long rollout of $t = 30$, while the baseline FNO fails to converge to a physical solution. See Fig. 16 for the full density evolution over time.

Domain generalization. We evaluate EddyFormer’s ability to generalize beyond the training domain by testing it on domains that are 4 times larger. EddyFormer requires minimal modification (Fig. 5): we tile more elements proportionally to the domain expansion rate and apply a fixed 16×16 attention window to the SEMAttn block. This ensures that each spectral element remains the same size between training and test time, while still exchanging information within a fixed $[0, 2\pi]^2$ neighborhood. This also ensures that the computation cost only scales linearly with domain size.

Fig. 5 shows the scaled energy spectra on both $2x$ and $4x$ larger domains. EddyFormer consistently matches the energy spectrum of high-resolution DNS across all length scales, generalizing to larger scales without degradation in accuracy. In contrast, F-FNO, while performing well on the original domain, shows noticeable deviations in the high-frequency modes on the larger domains, indicating a loss in consistency as the domain size increases.

4.3 Turbulence beyond homogeneous isotropy

The Well [Ohana et al., 2025] is a large-scale physics simulation dataset that contains fluid flows under a diverse set of conditions. We benchmark EddyFormer on turbulence-related subsets, which include magnetohydrodynamics, shear layer, thermal convection, and buoyancy-driven turbulence. We follow their training protocols and describe experiment details in §C.3.

Results. The VRMSE scores (Eqn. (25)) are reported in Tab. 3. Across all flow cases, EddyFormer achieves lower errors and consistently outperforms the baselines across a wide range of physically-driven turbulent flows. Fig. 6 visualizes the model predictions on the Rayleigh-Taylor instability problem. EddyFormer is able to accurately capture the development of turbulent plumes, while all baseline methods fail to converge.

Table 3: Performance on The Well dataset: time-averaged one-step VRMSE \downarrow scores on test sets. EF (cheb) and EF (leg) denote our EddyFormer model with Chebyshev and Legendre basis, respectively.

Dataset name	FNO	T-FNO	U-net	CNextU-net	EF (cheb)	EF (leg)
MHD_64	0.3605	0.3561	0.1798	0.1633	0.1160	<u>0.1240</u>
shear_flow	1.189	1.472	3.447	0.8080	<u>0.1123</u>	0.0837
rayleigh_benard	0.8395	0.6566	1.4860	0.6699	<u>0.2527</u>	0.1854
rayleigh_taylor_instab.	> 10	> 10	> 10	> 10	0.1115	0.1243

Conclusion. We introduced EddyFormer, an architecture inspired by the principles of large-eddy simulation, and designed for accurate and efficient turbulence simulation. EddyFormer demonstrates high accuracy with a significant acceleration on three-dimensional isotropic turbulence, as well as on flows driven by a wide range of conditions, ensuring stable rollouts over extended time periods. Future work will focus on extending EddyFormer to more complex flow domains and further scaling it for extreme-scale turbulence.

Acknowledgements. This work was supported by Laboratory Directed Research and Development (LDRD) funding under Contract Number DE-AC02-05CH11231. This research used resources of the National Energy Research Scientific Computing Center (NERSC), a Department of Energy User Facility. We thank Ningxiao Tao and Eric Qu for helpful discussions and feedback on earlier drafts.

Broader impacts. The development of EddyFormer has the potential to impact both scientific research and practical applications. By accelerating turbulence simulations, it can aid in solving complex fluid dynamics problems across fields such as climate modeling, aerospace engineering, and energy systems, where understanding turbulence is critical. Additionally, the use of machine learning to improve simulation efficiency opens new possibilities for real-time simulations, making it feasible to explore more dynamic and complex systems. However, as with any AI-driven technology, it is important to consider potential risks, such as over-reliance on model accuracy without sufficient validation or unintended biases in training data. Moreover, as simulation tools become more accessible, there is a responsibility to ensure their ethical and equitable use, particularly in areas like environmental modeling and resource management, where decisions based on inaccurate simulations could have widespread societal impacts.

References

- Benedikt Alkin, Andreas Fürst, Simon Schmid, Lukas Gruber, Markus Holzleitner, and Johannes Brandstetter. Universal physics transformers. *arXiv e-prints*, pages arXiv–2402, 2024.
- Yuri Bazilevs, Victor M Calo, John A Cottrell, Thomas JR Hughes, Alessandro Reali, and Guglielmo Scovazzi. Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows. *Computer methods in applied mechanics and engineering*, 197(1-4):173–201, 2007.
- Andrea D Beck, David G Flad, and Claus-Dieter Munz. Deep neural networks for data-driven turbulence models. *arXiv preprint arXiv:1806.04482*, 2018.
- Gal Berkooz, Philip Holmes, and John L Lumley. The proper orthogonal decomposition in the analysis of turbulent flows. *Annual review of fluid mechanics*, 25(1):539–575, 1993.
- Anudhyan Boral, Zhong Yi Wan, Leonardo Zepeda-Núñez, James Lottes, Qing Wang, Yi-fan Chen, John Anderson, and Fei Sha. Neural ideal large eddy simulation: Modeling turbulence with neural stochastic differential equations. *Advances in Neural Information Processing Systems*, 36:69270–69283, 2023.
- James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL <http://github.com/jax-ml/jax>.
- C Canuto. *Spectral Methods: Evolution to Complex Geometries and Applications to Fluid Dynamics*. Springer-Verlag, 2007.
- Shuhao Cao. Choose a transformer: Fourier or galerkin. *Advances in neural information processing systems*, 34: 24924–24940, 2021.
- Nithin Chalapathi, Yiheng Du, and Aditi Krishnapriyan. Scaling physics-informed hard constraints with mixture-of-experts. *arXiv preprint arXiv:2402.13412*, 2024.
- M Chertkov, C Connaughton, I Kolokolov, and V Lebedev. Dynamics of energy condensation in two-dimensional turbulence. *Physical review letters*, 99(8):084501, 2007.
- Yuchen Dang, Zheyuan Hu, Miles Cranmer, Michael Eickenberg, and Shirley Ho. Tnt: Vision transformer for turbulence simulations. *arXiv preprint arXiv:2207.04616*, 2022.
- Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
- Gideon Dresdner, Dmitrii Kochkov, Peter Norgaard, Leonardo Zepeda-Nunez, Jamie A Smith, Michael P Brenner, and Stephan Hoyer. Learning to correct spectral methods for simulating turbulent flows. *arXiv preprint arXiv:2207.00556*, 2022.
- Yiheng Du, Nithin Chalapathi, and Aditi Krishnapriyan. Neural spectral methods: Self-supervised learning in the spectral domain. *The Twelfth International Conference on Learning Representations*, 2024.

Vladimir Sergeevich Fanaskov and Ivan V Oseledets. Spectral neural operators. In *Doklady Mathematics*, volume 108, pages S226–S232. Springer, 2023.

Bernat Font, Gabriel D Weymouth, Vinh-Tan Nguyen, and Owen R Tutty. Deep learning of the spanwise-averaged navier–stokes equations. *Journal of Computational Physics*, 434:110199, 2021.

Zhongkai Hao, Zhengyi Wang, Hang Su, Chengyang Ying, Yinpeng Dong, Songming Liu, Ze Cheng, Jian Song, and Jun Zhu. Gnot: A general neural operator transformer for operator learning. In *International Conference on Machine Learning*, pages 12556–12569. PMLR, 2023.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Jacob Helwig, Xuan Zhang, Cong Fu, Jerry Kurtin, Stephan Wojtowytsh, and Shuiwang Ji. Group equivariant fourier neural operators for partial differential equations. *arXiv preprint arXiv:2306.05697*, 2023.

Maximilian Herde, Bogdan Raonic, Tobias Rohner, Roger Käppeli, Roberto Molinaro, Emmanuel de Bézenac, and Siddhartha Mishra. Poseidon: Efficient foundation models for pdes. *Advances in Neural Information Processing Systems*, 37:72525–72624, 2024.

Thomas Y Hou and Ruo Li. Computing nearly singular solutions using pseudo-spectral methods. *Journal of Computational Physics*, 226(1):379–397, 2007.

Jinhee Jeong and Fazle Hussain. On the identification of a vortex. *Journal of fluid mechanics*, 285:69–94, 1995.

George Karniadakis and Spencer J Sherwin. *Spectral/hp element methods for computational fluid dynamics*. Oxford University Press, 2005.

George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

Youngkyu Kim, Youngsoo Choi, David Widemann, and Tarek Zohdi. A fast and accurate physics-informed neural network reduced order model with shallow masked autoencoder. *Journal of Computational Physics*, 451:110841, 2022.

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

Jean Kossaifi, Nikola Kovachki, Kamyar Azizzadenesheli, and Anima Anandkumar. Multi-grid tensorized fourier neural operator for high-resolution pdes. *arXiv preprint arXiv:2310.00120*, 2023.

Robert H Kraichnan. Inertial ranges in two-dimensional turbulence. *Physics of fluids*, 10(7):1417, 1967.

Robert H Kraichnan and David Montgomery. Two-dimensional turbulence. *Reports on Progress in Physics*, 43(5):547, 1980.

Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. *Advances in neural information processing systems*, 34:26548–26560, 2021.

Pijush K Kundu, Ira M Cohen, David R Dowling, and Jesse Capecelatro. *Fluid mechanics*. Elsevier, 2024.

J Nathan Kutz, Steven L Brunton, Bingni W Brunton, and Joshua L Proctor. *Dynamic mode decomposition: data-driven modeling of complex systems*. SIAM, 2016.

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

Kookjin Lee and Kevin T Carlberg. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *Journal of Computational Physics*, 404:108973, 2020.

Zijie Li, Kazem Meidani, and Amir Barati Farimani. Transformer for partial differential equations’ operator learning. *arXiv preprint arXiv:2205.13671*, 2022.

Zijie Li, Dule Shu, and Amir Barati Farimani. Scalable transformer for pde surrogate modeling. *Advances in Neural Information Processing Systems*, 36:28010–28039, 2023a.

- Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.
- Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier neural operator with learned deformations for pdes on general geometries. *Journal of Machine Learning Research*, 24(388):1–26, 2023b.
- Zongyi Li, Nikola Kovachki, Chris Choy, Boyi Li, Jean Kossaifi, Shourya Otta, Mohammad Amin Nabian, Maximilian Stadler, Christian Hundt, Kamyar Azizzadenesheli, et al. Geometry-informed neural operator for large-scale 3d pdes. *Advances in Neural Information Processing Systems*, 36:35836–35854, 2023c.
- Marten Lienen, David Lüdke, Jan Hansen-Palmus, and Stephan Günnemann. From zero to turbulence: Generative modeling for 3d flow simulation. *arXiv preprint arXiv:2306.01776*, 2023.
- Julia Ling, Andrew Kurzawski, and Jeremy Templeton. Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics*, 807:155–166, 2016.
- Björn List, Li-Wei Chen, and Nils Thuerey. Learned turbulence modelling with differentiable fluid solvers: physics-based loss functions and optimisation horizons. *Journal of Fluid Mechanics*, 949:A25, 2022.
- Huakun Luo, Haixu Wu, Hang Zhou, Lanxiang Xing, Yichen Di, Jianmin Wang, and Mingsheng Long. Transolver++: An accurate neural solver for pdes on million-scale geometries. *arXiv preprint arXiv:2502.02414*, 2025.
- Michael McCabe, Bruno Régaldo-Saint Blanchard, Liam Holden Parker, Ruben Ohana, Miles Cranmer, Alberto Bietti, Michael Eickenberg, Siavash Golkar, Geraud Kräwezik, Francois Lanusse, et al. Multiple physics pretraining for physical surrogate models. *arXiv preprint arXiv:2310.02994*, 2023.
- Levi D McClenny and Ulisses M Braga-Neto. Self-adaptive physics-informed neural networks. *Journal of Computational Physics*, 474:111722, 2023.
- Nick McGreivy and Ammar Hakim. Weak baselines and reporting biases lead to overoptimism in machine learning for fluid-related partial differential equations. *Nature Machine Intelligence*, 6(10):1256–1269, 2024.
- Parviz Moin and Krishnan Mahesh. Direct numerical simulation: a tool in turbulence research. *Annual review of fluid mechanics*, 30(1):539–578, 1998.
- Geoffrey Négier, Michael W Mahoney, and Aditi S Krishnapriyan. Learning differentiable solvers for systems with hard constraints. *arXiv preprint arXiv:2207.08675*, 2022.
- Ruben Ohana, Michael McCabe, Lucas Meyer, Rudy Morel, Fruzsina Agocs, Miguel Beneitez, Marsha Berger, Blakesly Burkhardt, Stuart Dalziel, Drummond Fielding, et al. The well: a large-scale collection of diverse physics simulations for machine learning. *Advances in Neural Information Processing Systems*, 37:44989–45037, 2025.
- Pranshu Pant, Ruchit Doshi, Pranav Bahl, and Amir Barati Farimani. Deep learning for reduced order modelling and efficient temporal evolution of fluid simulations. *Physics of Fluids*, 33(10), 2021.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.
- Benjamin Peherstorfer and Karen Willcox. Data-driven operator inference for nonintrusive projection-based model reduction. *Computer Methods in Applied Mechanics and Engineering*, 306:196–215, 2016.
- Ugo Piomelli. Large-eddy simulation: achievements and challenges. *Progress in aerospace sciences*, 35(4):335–362, 1999.
- Stephen B Pope. Turbulent flows. *Measurement Science and Technology*, 12(11):2020–2021, 2001.
- Shaoxiang Qin, Fuyuan Lyu, Wenhui Peng, Dingyang Geng, Ju Wang, Naiping Gao, Xue Liu, and Liangzhu Leon Wang. Toward a better understanding of fourier neural operators: Analysis and improvement from a spectral perspective. *arXiv preprint arXiv:2404.07200*, 2024.
- Md Ashiqur Rahman, Robert Joseph George, Mogab Elleithy, Daniel Leibovici, Zongyi Li, Boris Bonev, Colin White, Julius Berner, Raymond A Yeh, Jean Kossaifi, et al. Pretraining codomain attention neural operators for solving multiphysics pdes. *Advances in Neural Information Processing Systems*, 37:104035–104064, 2024.

- Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.
- Varun Shankar, Dibyajyoti Chakraborty, Venkatasubramanian Viswanathan, and Romit Maulik. Differentiable turbulence: Closure as a partial differential equation constrained optimization. *Physical Review Fluids*, 10(2):024605, 2025.
- Joseph Smagorinsky. General circulation experiments with the primitive equations: I. the basic experiment. *Monthly weather review*, 91(3):99–164, 1963.
- Philippe Spalart. An old-fashioned framework for machine learning in turbulence modeling. *arXiv preprint arXiv:2308.00837*, 2023.
- Philippe Spalart and Steven Allmaras. A one-equation turbulence model for aerodynamic flows. In *30th aerospace sciences meeting and exhibit*, page 439, 1992.
- Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.
- Brendan D Tracey, Karthikeyan Duraisamy, and Juan J Alonso. A machine learning strategy to assist turbulence model development. In *53rd AIAA aerospace sciences meeting*, page 1287, 2015.
- Alasdair Tran, Alexander Mathews, Lexing Xie, and Cheng Soon Ong. Factorized fourier neural operators. *arXiv preprint arXiv:2111.13802*, 2021.
- Milton Van Dyke. *An album of fluid motion*, volume 176. Parabolic Press Stanford, 1982.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.
- Rui Wang, Karthik Kashinath, Mustafa Mustafa, Adrian Albert, and Rose Yu. Towards physics-informed deep learning for turbulent flow prediction. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1457–1466, 2020.
- Sifan Wang, Shyam Sankaran, and Paris Perdikaris. Respecting causality is all you need for training physics-informed neural networks. *arXiv preprint arXiv:2203.07404*, 2022.
- Sifan Wang, Shyam Sankaran, Panos Stinis, and Paris Perdikaris. Simulating three-dimensional turbulence with physics-informed neural networks. *arXiv preprint arXiv:2507.08972*, 2025.
- Gege Wen, Zongyi Li, Kamyar Azizzadenesheli, Anima Anandkumar, and Sally M Benson. U-fno—an enhanced fourier neural operator-based deep-learning model for multiphase flow. *Advances in Water Resources*, 163:104180, 2022.
- Matthew O Williams, Ioannis G Kevrekidis, and Clarence W Rowley. A data-driven approximation of the koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346, 2015.
- Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: A fast transformer solver for pdes on general geometries. *arXiv preprint arXiv:2402.02366*, 2024.
- Mingtao Xia, Lucas Böttcher, and Tom Chou. Spectrally adapted physics-informed neural networks for solving unbounded domain problems. *Machine Learning: Science and Technology*, 4(2):025024, 2023.
- Jin-Han Xie and Oliver Bühler. Exact third-order structure functions for two-dimensional turbulence. *Journal of Fluid Mechanics*, 851:672–686, 2018.
- PK Yeung, DA Donzis, and KR Sreenivasan. Dissipation, enstrophy and pressure statistics in turbulence simulations at high reynolds numbers. *Journal of Fluid Mechanics*, 700:5–15, 2012.
- PK Yeung, XM Zhai, and Katepalli R Sreenivasan. Extreme events in computational turbulence. *Proceedings of the National Academy of Sciences*, 112(41):12633–12638, 2015.

A Discussions on related works

We provide a broader overview of related work in ML for fluid dynamics, covering physics-informed learning, spectral methods, Transformer-based models, and reduced-order modeling.

ML for fluids. Except for the data-driven approaches discussed in §1, another line of work focuses on developing large pretrained models that can generalize across a wide range of flow conditions, geometries, and Reynolds numbers [McCabe et al., 2023, Herde et al., 2024, Rahman et al., 2024]. These models are trained on diverse simulation datasets and aim to serve as universal surrogates for fluid dynamics. In parallel, generative approaches have been proposed to capture the stochasticity and variability of turbulent flows. Lienen et al. [2023] apply diffusion models to synthesize physically plausible trajectories in a zero-shot setting, while Boral et al. [2023] introduce neural stochastic differential equations to model ensembles of turbulent trajectories.

Another popular direction involves physics-informed machine learning [Karniadakis et al., 2021], which integrates prior knowledge of problem during training. Physics-Informed Neural Networks (PINNs) [Raissi et al., 2019] incorporates the residuals of the governing equations into the loss function. This framework allows for training with sparse or no supervision. Works in this direction seek to address challenges such as training strategy [Krishnapriyan et al., 2021, Wang et al., 2022], residual estimation [McClenny and Braga-Neto, 2023, Du et al., 2024], and constraint enforcement [Négier et al., 2022, Chalapathi et al., 2024]. Recent advances also demonstrated that PINNs are able to simulate large-scale three-dimensional turbulence [Wang et al., 2025]. However, the training cost of PINNs, and solving for one problem at a time, makes their efficiency worse compared to state-of-the-art numerical simulations.

Spectral-based models. Spectral methods [Canuto, 2007] provide powerful inductive biases for neural fluid simulation. Li et al. [2020] introduced Fourier neural operator (FNO), which is based on spectral convolutions in the Fourier domain. Neural operators have demonstrated accuracy and efficiency advantages over traditional convolutional networks. Subsequent works extended neural operators by either improving the spectral kernel’s design [Tran et al., 2021, Fanaskov and Oseledets, 2023, Helwig et al., 2023], or by learning completely in the spectral domain [Du et al., 2024, Xia et al., 2023]. These models have shown promising results on low-dimensional problems; however, due to the inefficiency of parameterizing and learning large Fourier convolution kernels [Qin et al., 2024], their application to large-scale three-dimensional turbulence has been limited.

Moreover, they face challenges on irregular domain geometries since spectral methods are inherently homogeneous. To overcome this, several extensions have been proposed that incorporate geometry more explicitly [Li et al., 2023b,c]. However, these methods still encounter limitations in accurately capturing complex geometries, as well as in applying to problems with a wide range of length scales.

Transformers for spatiotemporal modeling. There has been an increased interest in exploring the Transformer architecture for fluid simulation, and more generally, for solving partial differential equations. One line of work [Dang et al., 2022, McCabe et al., 2023, Herde et al., 2024] applied well-established Vision Transformer (ViT) architectures to this task. However, due to the quadratic complexity of the self-attention mechanism, their scalability is limited by the spatial resolution of the solution field, especially for the resolution of three-dimensional turbulent flows.

Several approaches have been proposed to address this challenge: **(a) simplified attention:** Cao [2021] draw connections between the attention mechanism and kernel convolution, providing insights into reducing computational complexity. GNOT [Hao et al., 2023], FactFormer [Li et al., 2023a], OFormer [Li et al., 2022], and AViT [McCabe et al., 2023] applied different simplifications to the attention mechanism, enabling linear or almost-linear computational complexity; **(b) grouped tokens:** this approach reduces the number of tokens involved in the attention mechanism by grouping relevant mesh points together. UPT [Alkin et al., 2024] uses sub-sampling supernodes in combination with a perceiver-style architecture, while Transolver [Wu et al., 2024, Luo et al., 2025] operates on slice-based physical tokens. While these approaches have shown promise for moderate-scale problems, their effectiveness and accuracy on large-scale turbulence problems is unclear.

Reduced-order modeling. Reduced Order Modeling (ROM) provides a framework for approximating high-fidelity simulations with lower computational cost. Classical methods such as Proper Orthogonal Decomposition (POD) [Berkooz et al., 1993] have been extensively used as low-dimensional representations. Data-driven approaches, including Dynamic Mode Decomposition [Williams et al.,

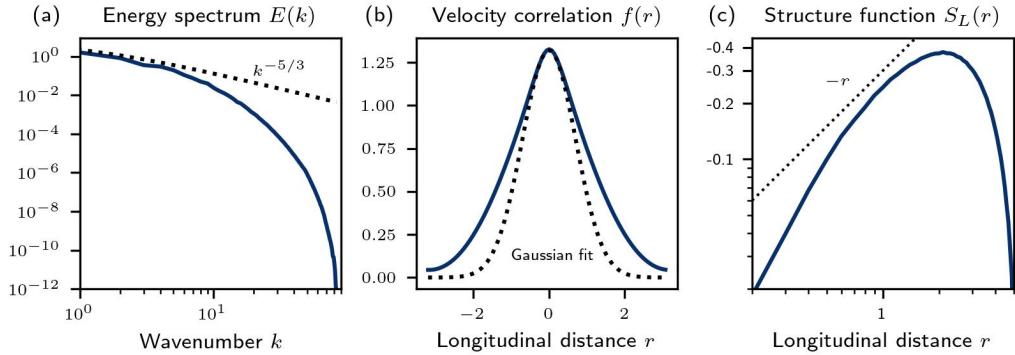


Figure 7: Energy spectrum, velocity correlation, and structure function [Pope, 2001] of Re94 . The energy spectrum follows the $-5/3$ scaling law, indicating inertial-range turbulence. The non-Gaussian velocity correlation shows intermittent behavior. Additionally, the structure function exhibits scaling consistent with Kolmogorov’s theory, further confirming the presence of turbulent dynamics.

2015, Kutz et al., 2016] and Operator Inference [Peherstorfer and Willcox, 2016], have expanded the scope of reduced-order modeling by constructing surrogate models directly from data. Recently, autoencoders has been employed to construct nonlinear reduced-order models [Lee and Carlberg, 2020, Pant et al., 2021, Kim et al., 2022]. These approaches aim to learn low-dimensional manifolds that capture the essential dynamics of complex systems beyond linear methods. However, these methods often require complete information of the solution regime; consequently, their accuracy and generalization usually remain problem-dependent.

B Dataset details

In this section, we provide details on the two and three-dimensional turbulence dataset used in our experiments. Tab. 4 and Tab. 5 contains simulation parameters and flow statistics in the dataset. Fig. 7 plots the energy spectrum of each flow, averaged across the whole test set.

Initial condition. The lowest Fourier modes $|\mathbf{k}| < 16$ of every initial velocity field are randomly initialized according to the $k^{-5/3}$ spectrum. The initial condition is then burned in for a fixed time t_0 , which is chosen empirically so that energy injection and dissipation reach equilibrium.

Solver details. We use pseudo-spectral method to generate the high-accuracy dataset as well as the low-resolution input for learned correction on top of it. Smooth dealiasing [Hou and Li, 2007] is used during advection in order to keep more modes than the common $2/3$ rule. We use fourth-order Runge-Kutta scheme for time-stepping, coupled with a second-order implicit diffusion step. The time step is determined dynamically by fixing Courant–Friedrichs–Lewy (CFL) number below $C_{\max} = 1$.

Dataset. The dataset is downsampled onto uniform grids using FFT at intervals of Δt , over a total duration t . The test set contains 16 longer trajectories: $t = 50s$ for KF4 and $t = 20s$ for Re94 .

Table 4: Simulation parameters of KF4 (§4.2) and Re94 (§4.1).

Name	KF4	Re94
Domain Ω	$[0, 2\pi]^2$	$[0, 2\pi]^3$
Burn-in time t_0	40s	20s
Record time Δt	0.125s	0.1s
Trajectory length t	10s	5.0s
Solver resolution	4096^3	384^3
Dataset resolution	256^2	96^3
Dataset size	89GB	1TB

Table 5: Turbulence statistics of KF4 (§4.2) and Re94 (§4.1).

Name	KF4	Re94
Viscosity ν	0.001	0.01
Kinetic energy E_k	2.0061	3.6023
Dissipation rate ϵ	0.0419	0.9749
Velocity r.m.s. $\langle v \rangle_2$	1.1539	1.5493
Taylor microscale λ	0.6939	0.6090
Kolmogorov scale η	0.0124	0.0318
Reynolds no. Re_λ	799.76	94.40

Evaluation metrics. We report standard error metrics, such as relative L_2 error, as well as other invariant flow statistics when evaluating all models and numerical solvers. Specifically, let \mathbf{u} be the reference solution and \mathbf{u}' be the prediction, the following metrics are reported:

1. Relative L2 error $\|\mathbf{u} - \mathbf{u}'\|_2 / \|\mathbf{u}\|_2$. Relative error is used when \mathbf{u} and \mathbf{u}' are highly correlated, i.e., during early steps in the rollout. It is also used as the training objective (see Eqn. (15)).
2. Point-wise correlation $\langle \mathbf{u}, \mathbf{u}' \rangle / \|\mathbf{u}\|_2^2$. Correlation is used in later solution rollout, when \mathbf{u} and \mathbf{u}' are statistically correlated, but relative error is too large to be a meaningful metric.
3. Energy spectrum $E(k)$,

$$E(k) = \int_{|\mathbf{k}|=k} \frac{1}{2} |\hat{\mathbf{u}}(\mathbf{k})|^2 d\mathbf{k}. \quad (17)$$

The energy spectrum describes how turbulent energy is distributed across different length scales. In the inertial range, $E(k)$ is expected to follow Kolmogorov's 5/3 law, i.e., $E(k) \propto k^{-5/3}$. Accurately reproducing the energy spectrum is necessary for a model to yield physically meaningful predictions; even after the model's prediction becomes decorrelated from the reference, $E(k)$ remains critical because it still reflects the turbulence cascade.

4. Two-point third-order longitudinal structure function S_L ,

$$S_L(r) = \langle ([\mathbf{u}(\mathbf{x} + \mathbf{r}) - \mathbf{u}(\mathbf{x})] \cdot \frac{\mathbf{r}}{r})^3 \rangle_{|\mathbf{r}|=r}. \quad (18)$$

The structure function $S_L(r)$ measures the statistical asymmetry of the velocity differences across a distance r , indicating the downscale energy transfer across turbulence scales. It is hypothesized to follow the Kolmogorov's 4/5 law, i.e., $S_L(r) = -\frac{4}{5}\epsilon r$, where ϵ is the dissipation rate. However, in two-dimensional turbulence where inverse energy transfer, i.e., “backscattering”, occurs, S_L exhibits more complex behavior [Kraichnan, 1967, Kraichnan and Montgomery, 1980, Xie and Bühler, 2018]. Evaluating the two-point structure function on a model's prediction tests whether it preserves the correct inter-scale statistics.

5. The Q-criterion q ,

$$q = \frac{1}{2} (\|\Omega\|^2 - \|S\|^2), \quad (19)$$

where $S = \frac{1}{2}(\nabla u + \nabla u^T)$ and $\Omega = \frac{1}{2}(\nabla u - \nabla u^T)$ are the strain rate and rotation tensor, respectively. The Q-criterion is a widely used diagnostic metric [Jeong and Hussain, 1995] for identifying vortical structures in turbulent flows. Regions where $q > 0$ indicate local dominance of rotational motion over strains, which typically corresponds to coherent vortex cores. It provides a frame-invariant and physically interpretable way to visualize and quantify vortical structures in complex turbulent flows. We use the Q-criterion to visually compare the vortex cores in the reference DNS solution and EddyFormer's prediction.

C Experiment details

Hyperparameters. For 2D and 3D problems, EddyFormer uses the hyperparameters described in Tab. 6 and Tab. 7, respectively. We consider both Chebyshev and Legendre basis as the orthogonal polynomials in SEM basis for most problems, which is indicated by EddyFormer (cheb) and EddyFormer (leg), respectively. We use the Adam optimizer with a learning rate of 10^{-3} . For homogeneous turbulence §4.1 and §4.2, we use a batch size of 16, while a batch size of 64 is used in §4.3.

Table 6: EddyFormer architecture (2D).

Stream	LES	SGS
Number of layers L	10	
Hidden dimension d	32	
Mesh size H	16^2	
Mode k_{\max} and M	4	24
Kernel mode m	4	24
Kernel size s	$\pi/4$	$\pi/4$
Number of heads n_{head}	8	—
Head dimension d_{head}	16	—

Table 7: EddyFormer architecture (3D).

Stream	LES	SGS
Number of layers L	4	
Hidden dimension d	32	
Mesh size H	8^3	
Mode k_{\max} and M	5	13
Kernel mode m	5	13
Kernel size s	$\pi/4$	$\pi/4$
Number of heads n_{head}	4	—
Head dimension d_{head}	32	—

Implementation details. EddyFormer is implemented using the JAX framework [Bradbury et al., 2018]. We implement our own three-dimensional pseudo-spectral solver for all data generations and learned correction used in our experiments. We acknowledge the JAX-CFD [Dresdner et al., 2022] and NSM [Du et al., 2024] codebases, which we referenced and built upon in our implementation.

Implementation of SEMConv. The SEM-based convolution, Eqn. (8), evaluates the convolution between a Fourier kernel and features on collocation points. For simplicity, here we only consider the implementation of 1D convolution. Its extension to multi-dimensional case is straightforward due to the use of factorized kernels. Specifically, the SEMConv performed along x axis is defined by,

$$\text{SEMConv}(\mathbf{u})(x') = \int_{|x| \leq s/2} \mathbf{k}(x) \mathbf{u}(x' - x)^T dx. \quad (20)$$

We use quadrature rules to directly compute the convolution on each collocation point. Specifically, Clenshaw–Curtis and Gauss-Lobatto-Legendre quadratures are used for Chebyshev and Legendre-based SEM bases, respectively. For each collocation point $x_h^{(n)}$ in the h 'th element, the integral is the sum of the quadratures in neighbor elements,

$$\text{SEMConv}(\mathbf{u})(x_h^{(n)}) \approx \sum_{|h'-h| \leq S} \sum_{m \leq M} w_{h'}^{(m)} \mathbf{k}(x_{h'}^{(m)} - x_h^{(n)}) \mathbf{u}(x_{h'}^{(m)})^T, \quad (21)$$

where $S = \lceil \frac{s}{2\Delta} \rceil$ is the half-width of the neighbors within the convolution window s , and $\{w_{h'}^{(m)}\}_m$ are the quadrature weights on element $\Omega_{h'}$, which can be pre-computed using routines from standard packages like QUADPACK [Virtanen et al., 2020]. Below is a pseudo-code of our implementation.

Algorithm 1: SEMConv Implementation (1D)

```

Input: Feature field  $\mathbf{u}$ ; Fourier convolution kernel  $\mathbf{k}(x)$ 
Output: Convolved feature field  $\text{SEMConv}(\mathbf{u})$ 
 $S \leftarrow \lceil s/2\Delta \rceil;$  // Number of neighbors
 $w \leftarrow \text{QuadratureWeights}();$  // Precomputed weights
foreach collocation point  $x_h^{(n)}$  do
     $y \leftarrow 0;$  // Output at  $x_h^{(n)}$ 
    for  $h' = h - S$  to  $h + S$  do
        for  $m = 0$  to  $M$  do
             $w \leftarrow w_{h'}^{(m)};$  // Quadrature weight
             $k \leftarrow \mathbf{k}(x_{h'}^{(m)} - x_h^{(n)})$ ; // Evaluate convolution kernel
             $u \leftarrow \mathbf{u}(x_{h'}^{(m)})$ ; // Evaluate input feature
             $y += w \cdot k \cdot u^T;$ 
     $\text{SEMConv}(\mathbf{u})(x_h^{(n)}) \leftarrow y$ 

```

Computation cost of SEMConv. The computational cost of Algorithm 1 consists of two main components: evaluating the kernel \mathbf{k} on collocation points and performing quadrature over the input field. Under the assumption of a uniform mesh, the kernel only needs to be evaluated at MS distinct offsets, which is equivalent to performing an $MS \times MS$ -size matrix multiplication. The quadrature step involves a weighted dot product across neighbors, which costs $\mathcal{O}(M^2S)$ per element. Therefore, the total cost scales with the number of resolved modes, with a sublinear factor of $\mathcal{O}(MS)$.

C.1 Two-dimensional turbulence KF4

Learned correction. All models are trained to correct a DNS at 256^2 resolution. Each model is first trained using the one-step loss, Eqn. (15), for 30k steps with a learning rate of 10^{-3} . In order to control the roll-out error over time, we also differentiate through the DNS steps and minimize accumulated error as Kochkov et al. [2021] did. Specifically, $\text{LC}_{\theta}^{(n)}$ represents the n -step prediction roll-out using learned corrections, and the N -step loss function is defined as follows:

$$\mathbb{E}_{t, \mathbf{u}_0} \left[\frac{1}{N} \sum_{n \leq N} \|\mathbf{u}_{t+n\Delta t} - \text{LC}_{\theta}^{(n)}(\mathbf{u}_t)\| / \|\mathbf{u}_{t+n\Delta t}\| \right], \quad (22)$$

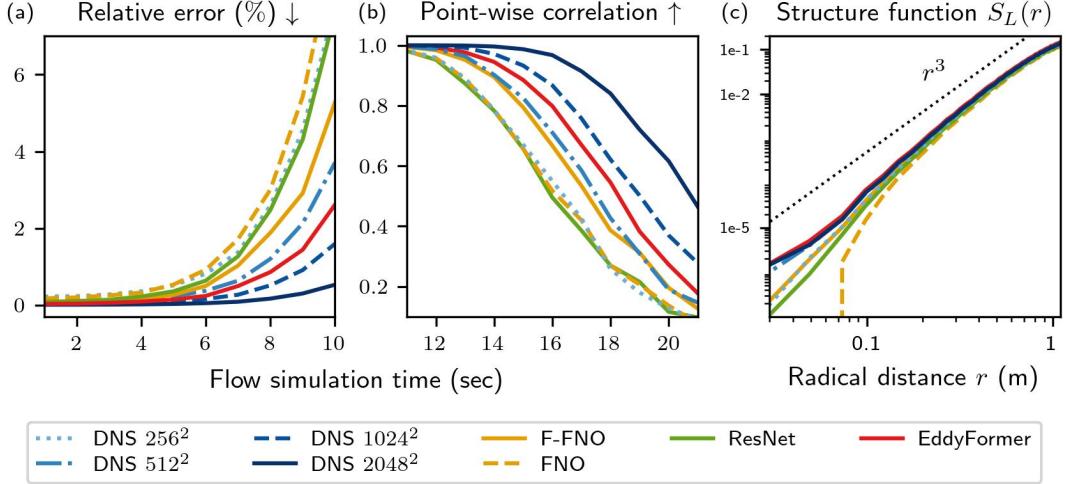


Figure 8: Performance of models and DNS at different resolutions on KF4 (§4.2), a two-dimensional fluid flow example (“Kolmogorov” flow). The models learn to correct DNS at 256^2 resolution, and the metrics here are relative to a reference DNS at 4096^2 resolution. EddyFormer achieves the accuracy of DNS between 512^2 and 1024^2 resolution in terms of (a) relative L2 error for $t \leq 10$ and (b) point-wise correlation for $10 \leq t \leq 20$; for (c) longer rollouts up to 50 seconds, EddyFormer also accurately captures the third-order structure function, Eqn. (18), which characterizes the forward and inverse energy transfer in fine length-scales.

However, as reported by Dresdner et al. [2022], directly minimizing multi-step loss is difficult and unstable. Therefore, we choose $N = 5$ and only fine-tuned using the 5-step loss for another 5k steps with a reduced learning rate of 10^{-5} , after the one-step training converged.

Baseline models. We consider ResNet [He et al., 2016], FNO [Li et al., 2020], and F-FNO [Tran et al., 2021] as baseline models for comparison. For ResNet, we adopt the standard ResNet-18 architecture. For both FNO and F-FNO, we use 64^2 modes, with 10 layers and 32 hidden dimensions, to ensure a fair comparison with EddyFormer. All models, except for FNO, have a model size of the order of 10MB.

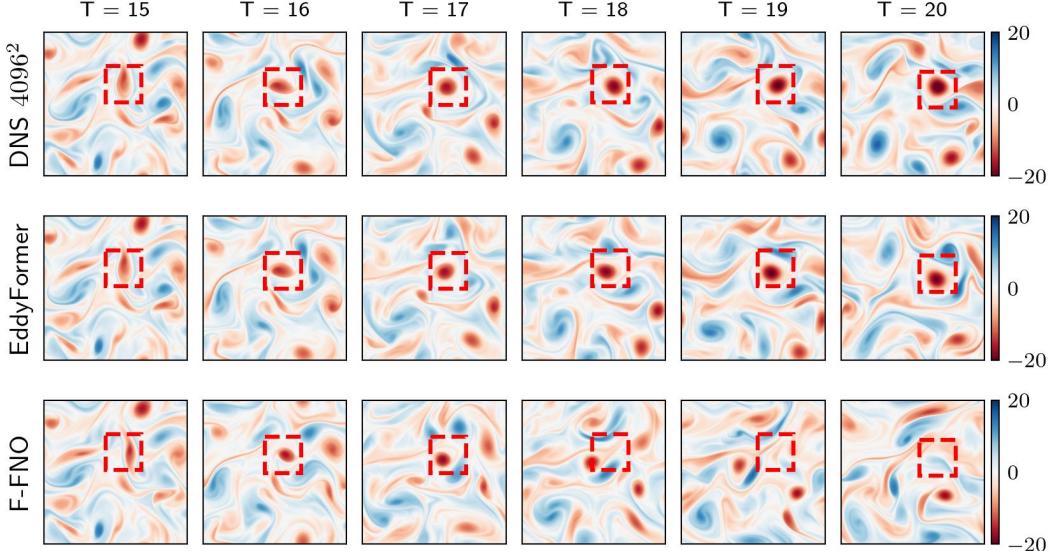


Figure 9: Vorticity visualization of a test sample on KF4 (§4.2). EddyFormer accurately tracks the dynamics of the vortex core (indicated by the red square) up to $t = 20$, while the baseline F-FNO fails to resolve it over the long rollout.

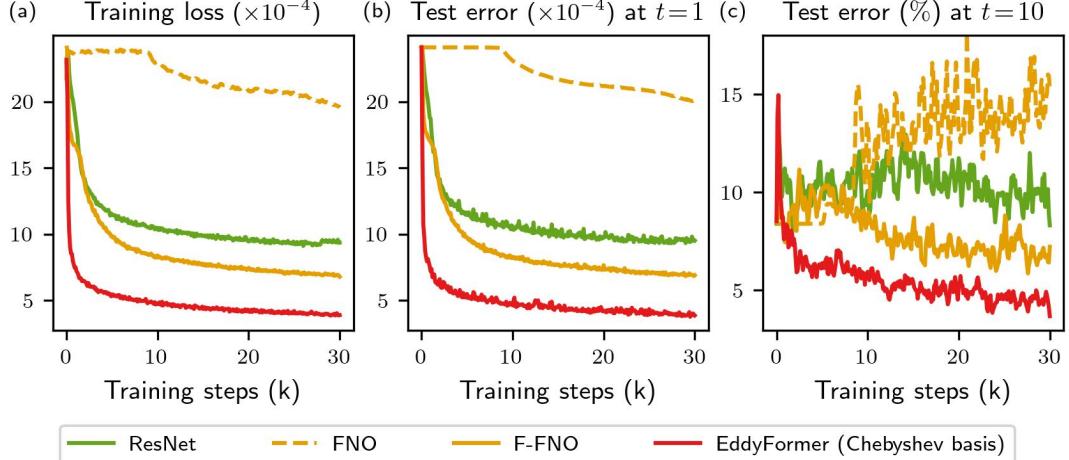


Figure 10: **(a)** Training loss, and **(b)** and **(c)** test error curves on KF4 (§4.2). EddyFormer converges significantly faster than all baselines, while achieving almost a 50% reduction in converged loss.

For reference, we also include results reported by previous work [Kochkov et al., 2021], which were obtained by correcting a lower-accuracy finite-volume DNS at 64^2 resolution. For a fair comparison, we reproduced their result by correcting our high-accuracy pseudo-spectral solver using a ResNet.

Results. Fig. 10 shows the training loss as well as test errors at different times. Tab. 8 and Tab. 9 report the relative L2 error and point-wise correlation metrics on test set. EddyFormer is able to converge faster than the baselines, consistently achieving lower test error. Fig. 9 visualizes the evolution of a vortex core, showing that EddyFormer captures its development up to $t = 20$.

Table 8: Relative L2 error (%) on KF4 test set with learned correction of DNS 256^2 .

	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$	$t=7$	$t=8$	$t=9$	$t=10$
DNS 256^2	0.23	0.24	0.28	0.36	0.52	0.82	1.38	2.59	4.55	7.79
DNS 512^2	0.09	0.10	0.12	0.16	0.23	0.38	0.64	1.21	2.13	3.71
DNS 1024^2	0.04	0.04	0.05	0.07	0.10	0.16	0.27	0.52	0.92	1.60
Kochkov et al.	0.63	1.38	2.53	4.54	8.09	14.7	24.3	36.5	56.4	74.5
FNO	0.17	0.20	0.28	0.40	0.71	1.19	2.06	3.58	6.59	11.4
ResNet	0.08	0.10	0.14	0.22	0.36	0.63	1.27	2.47	4.30	7.58
F-FNO	0.06	0.07	0.10	0.15	0.27	0.50	1.03	1.88	2.91	5.28
EddyFormer	0.03	0.04	0.06	0.09	0.14	0.24	0.49	0.86	1.44	2.61

Table 9: Point-wise correlation on KF4 test set with learned correction of DNS 256^2 .

	$t=11$	$t=12$	$t=13$	$t=14$	$t=15$	$t=16$	$t=17$	$t=18$	$t=19$	$t=20$
DNS 256^2	0.98	0.96	0.89	0.78	0.67	0.55	0.43	0.26	0.18	0.13
DNS 512^2	1.00	0.99	0.96	0.90	0.83	0.71	0.59	0.43	0.31	0.19
DNS 1024^2	1.00	1.00	0.99	0.97	0.93	0.87	0.76	0.62	0.50	0.37
Kochkov et al.	0.69	0.51	0.36	0.24	0.15	0.09	0.09	0.02	-0.01	0.02
FNO	0.98	0.96	0.89	0.79	0.66	0.52	0.42	0.27	0.21	0.14
ResNet	0.98	0.95	0.87	0.78	0.66	0.50	0.39	0.27	0.22	0.12
F-FNO	0.99	0.98	0.95	0.89	0.79	0.67	0.53	0.39	0.31	0.20
EddyFormer	1.00	0.99	0.98	0.94	0.88	0.80	0.67	0.55	0.38	0.27

Discussions on domain generalization. This expansion task (Fig. 5) is conceptually different from super-resolution [Li et al., 2020] aims to recover finer-scale details on a fixed physical domain, domain expansion keeps the grid spacing and increases only the physical extent of the flow, testing whether the model has overfitted to domain-specific flow dynamics. Neural operators have no corresponding expansion mechanism. We tried two ad-hoc work-arounds: rescaling the appended spatial coordinates and stretching the learned convolution kernels. Neither produced comparable results: Fig. 5 shows that F-FNO, by rescaling the appended coordinates, fails to match the energy spectra on the expanded domains.

Energy conservation. In addition to evaluating the statistical properties of model’s long-term prediction, we also assess the model’s ability to conserve total kinetic energy, E_{tot} , over time:

$$E_{\text{tot}}(T) = k|_{t=0} + \int_0^T I(t) - \epsilon(t) dt, \quad (23)$$

where the kinetic energy $k = \langle \frac{1}{2} \mathbf{u}^2 \rangle$, the dissipation rate $\epsilon = 2\nu \langle \|S\|_2^2 \rangle$ with $S = \frac{1}{2}(\nabla u + \nabla u^T)$ denoting the strain-rate tensor, and the injection rate $I = \langle \mathbf{f} \cdot \mathbf{u} \rangle$ given a forcing term \mathbf{f} . This cumulative energy accounts for the balance of injection and dissipation and should remain constant in time. Even though energy conservation is a necessary condition for a physically correct model and is implicitly present in the training data, the models are not explicitly informed of this constraint and must learn to conserve energy purely from data.

Tab. 10 reports the time-averaged total kinetic energy over a test trajectory. The results indicate that EddyFormer preserves the energy balance over long rollouts, whereas the baseline F-FNO exhibits significant and systematic energy dissipation over time.

Table 10: Time-averaged total kinetic energy $E_{\text{tot}}(t)$ of a trajectory in KF4 test set.

Time interval	[0, 5]	[5, 10]	[10, 15]	[15, 20]	[20, 25]	[25, 30]	[30, 35]	[35, 40]	[40, 45]	[45, 50]
EddyFormer	1.728	1.668	1.692	1.660	1.712	1.736	1.695	1.694	1.706	1.670
F-FNO	1.728	1.668	1.688	1.588	1.650	1.623	1.524	1.518	1.502	1.511

C.2 Three-dimensional turbulence Re94

The Re94 case is adopted from Bazilevs et al. [2007]. The forcing term $\mathbf{f}(\mathbf{x})$,

$$\mathbf{f}(\mathbf{x}) = \sum_{|\mathbf{k}| \leq 1, \mathbf{k} \neq 0} \frac{P_{\text{in}}}{E_1} \hat{\mathbf{u}}_{\mathbf{k}} e^{i\mathbf{k} \cdot \mathbf{x}}, \quad (24)$$

injects energy uniformly across all directions. Here, $\hat{\mathbf{u}}_{\mathbf{k}}$ is the \mathbf{k}' th Fourier coefficient of the velocity.

Direction prediction (“pure ML”) benchmark. For baseline comparisons, we adopt ResNet, FNO, and F-FNO in 3D without modifying their original architectures. For FNO, we reduce the number of modes to 16^3 in order to avoid exceeding the model size constraint. In addition, we also consider TF-Net [Qin et al., 2024], GNOT [Hao et al., 2023], and AViT [McCabe et al., 2023] as additional baselines. We use their default architectures and adapt them to handle three-dimensional input fields. The number of trainable parameters for each model is reported in Tab. 1.

All models are trained for 15k steps using one-step loss. Fig. 11 shows the training loss of different baseline models. EddyFormer converges significantly faster and consistently achieving lower test error than state-of-the-art neural operators and Transformers. In these baselines, F-FNO is the best model, yet EddyFormer achieves almost $3\times$ lower error.

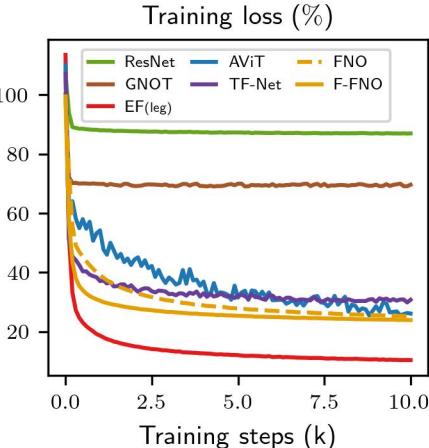


Figure 11: Direct prediction loss curves Re94 (§4.1). EF(leg) denotes EddyFormer with Legendre basis.

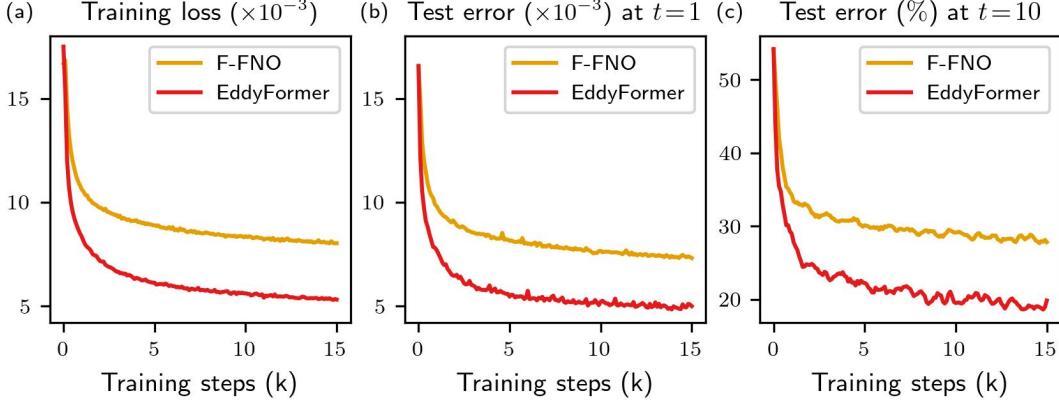


Figure 12: (a) Training loss, and (b) and (c) test error curves on Re94 (§4.1) with learned correction of DNS at 96^3 resolution. EddyFormer consistently achieves lower error on both training and test set.

Learned correction. Both F-FNO and EddyFormer are trained using one-step loss, Eqn. (15), for 15k steps with a learning rate of 10^{-3} . Fig. 12 shows the training loss as well as test errors at different times. Tab. 11 and Tab. 12 report the relative L2 error and point-wise correlation metrics on test set.

Table 11: Relative L2 error (%) on Re94 test set with learned correction of DNS 96^3 .

	$t=0.5$	$t=1.0$	$t=1.5$	$t=2.0$	$t=2.5$	$t=3.0$	$t=3.5$	$t=4.0$	$t=4.5$	$t=5.0$
DNS 96^3	1.53	3.34	5.64	8.74	13.0	18.7	26.0	34.8	44.8	55.9
DNS 128^3	1.04	2.28	3.88	6.08	9.1	13.3	19.0	26.2	35.0	45.5
DNS 192^3	0.53	1.17	2.00	3.17	4.84	7.16	10.4	14.9	20.9	28.8
DNS 256^3	0.27	0.60	1.02	1.62	2.48	3.70	5.44	7.92	11.3	16.3
F-FNO	0.68	1.18	1.87	2.93	4.44	6.60	9.71	14.0	19.8	27.4
EddyFormer	0.46	0.78	1.23	1.89	2.80	4.20	6.23	9.27	13.1	18.2

Table 12: Point-wise correlation on Re94 test set with learned correction of DNS 96^3 .

	$t=5.5$	$t=6.0$	$t=6.5$	$t=7.0$	$t=7.5$	$t=8.0$	$t=8.5$	$t=9.0$	$t=9.5$	$t=10$
DNS 96^3	0.78	0.71	0.63	0.55	0.47	0.39	0.30	0.23	0.16	0.09
DNS 128^3	0.84	0.78	0.70	0.62	0.54	0.45	0.36	0.28	0.20	0.13
DNS 192^3	0.93	0.88	0.82	0.75	0.67	0.60	0.51	0.43	0.34	0.25
DNS 256^3	0.97	0.95	0.91	0.86	0.80	0.74	0.67	0.59	0.51	0.43
F-FNO	0.93	0.89	0.83	0.76	0.69	0.62	0.54	0.46	0.38	0.30
EddyFormer	0.97	0.94	0.90	0.85	0.78	0.71	0.64	0.56	0.48	0.41

C.2.1 Ablation studies

In order to better understand EddyFormer and the right model design choices, we perform various ablation studies using the direct prediction (“pure ML”) benchmark on Re94 . First, we consider the following modifications to isolate the model components:

- **EF (LES).** EddyFormer using only the LES stream for large-scale dynamics.
- **EF (SGS).** EddyFormer using only the SGS stream for small-scale dynamics.
- **EF (FFT).** EddyFormer variant with SEMConv replaced by Fourier-based convolution. A factorized kernel with the same number of modes as F-FNO is used for a fair comparison.
- **EF (CNN).** EddyFormer variant with SEMConv replaced by standard convolution. A 3D convolution with the same kernel size is directly applied on the collocation points.

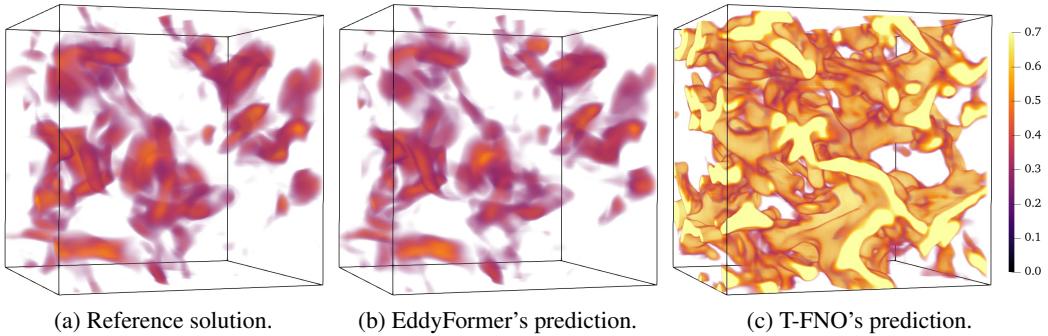


Figure 13: Visualization of the density field for a MHD_64 test sample at $t = 3$. EddyFormer is able to match the reference solution while the baseline T-FNO’s prediction diverges after three steps.

Table 13: Direct prediction benchmark on Re94 : one-step error of different ablation models.

	EF	EF (LES)	EF (SGS)	EF (FFT)	EF (CNN)	F-FNO	ResNet
Test error	8.61%	12.6%	23.2%	12.8%	25.3%	22.5%	87.2%

Results. Tab. 13 reports the one-step error of the ablation models. The results show that the LES stream alone performs worse without the support of the SGS stream, and the SGS stream alone is slightly less accurate than its F-FNO counterpart. This suggests that many global modes in FNOs may be redundant for turbulence modeling. EddyFormer with CNN shows a significant accuracy drop, consistent with the gap between ResNet and FNO. The FFT-based variant of EddyFormer also performs worse, highlighting that the SEM basis offers an advantage over FFT and serves as a core component of our architecture. Compared to the LES-only version, the full model demonstrates that **(a)** SEMAttn captures large-scale dynamics more effectively than F-FNO, and **(2)** SEMConv models small-scale dynamics that FFT-based spectral convolution fail to resolve.

Choice of the LES filter. In classical LES, the filter size defines the cutoff between resolved and subgrid-scale motions, and is typically set based on the grid resolution or an associated spectral cutoff based on the energy spectrum.

To assess the impact of the LES filter size, we vary k_{\max} while keeping all other hyperparameters fixed. The results in Tab. 14 show that increasing k_{\max} improves accuracy up to $k_{\max} = 5$, beyond which the performance saturates. This saturation reflects the limited resolution capacity of the SEMAttn module, which determines the range of large-scale structures the LES stream can effectively capture. When the spectral filter includes higher-frequency modes beyond this capacity, the LES stream is unable to utilize them effectively—especially under a fixed model size. In such cases, increasing the cutoff adds computational cost without improving performance. These results validate our choice of $k_{\max} = 5$ as a balance between expressivity and efficiency.

C.3 The Well dataset

We benchmark EddyFormer on the turbulence-related subsets in The Well. We follow all training protocols described by Ohana et al. [2025], with the exception of the training duration. EddyFormer is trained for 10k steps on each problem, rather than the one-day training budget used in the original benchmark, to ensure reproducibility. The batch size is set to 64. We report the variance-scaled mean square error (VRMSE) scores on test sets as Ohana et al. [2025] did, which is defined as:

$$\text{VRMSE}(u, u') = (\langle |u - u'|^2 \rangle / (\langle |u - \bar{u}|^2 \rangle + \epsilon))^{1/2}, \quad (25)$$

where u is the reference solution and u' is the prediction. The safety value $\epsilon = 10^{-7}$ prevents extreme values on near-constant fields. All reported VRMSE scores are averaged across all time steps in the test set. Baseline results in Tab. 3 are taken from the original benchmark by Ohana et al. [2025].

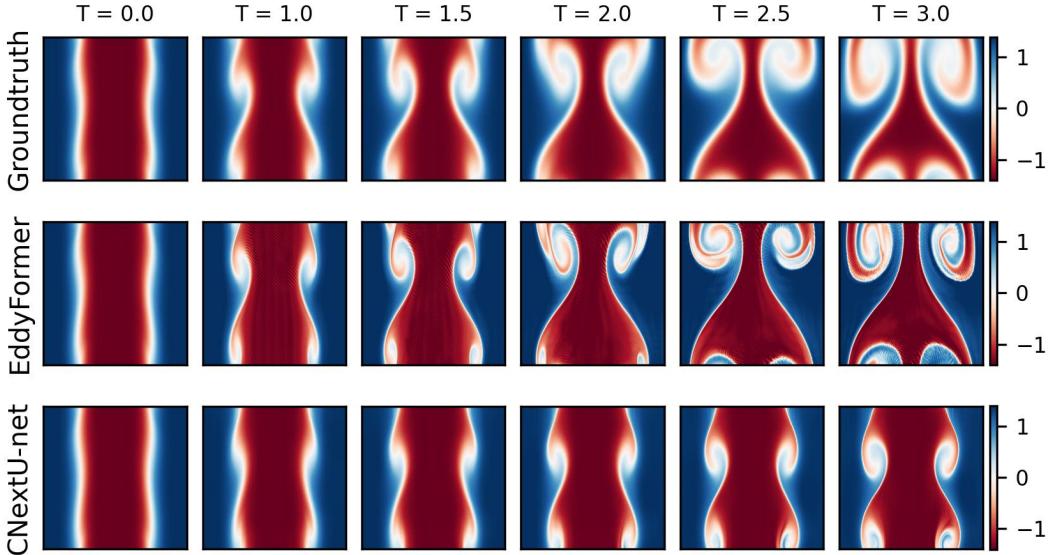


Figure 14: Visualization of the tracer field for a `shear_flow` test sample over time. EddyFormer predicts the development of two vortices in the shear layer, while the baseline CNextU-net converged to an over-dissipative solution.

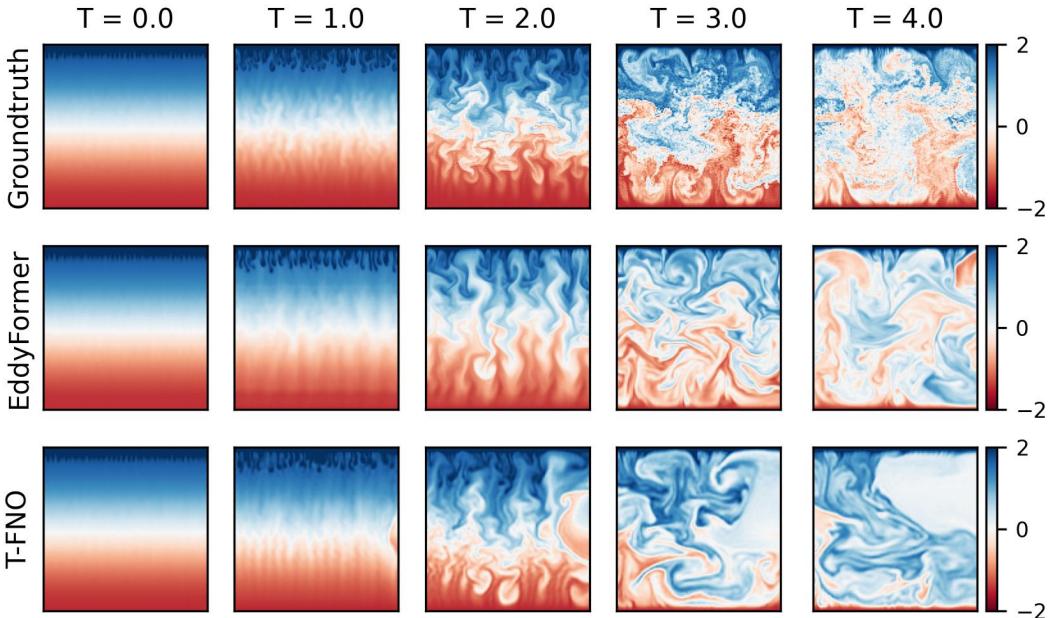


Figure 15: Visualization of the density field for a `rayleigh_benard` test sample over time. EddyFormer predicts the growth of the mixing layer as well as the plume structures, while the baseline T-FNO predicts a non-physical solution (the solution collapsed into a uniform region at $T = 4.0$).

Results. Tab. 15 and Tab. 16 shows the time-averaged per-field VRMSE scores, and Tab. 3 shows the field-average VRMSE scores. Fig. 13, Fig. 14, Fig. 15, and Fig. 16 show visualizations of the `MHD_64`, `shear_flow`, `rayleigh_benard`, and `rayleigh_taylor_instab`. cases, respectively.

Table 15: EddyFormer (Chebyshev basis) on The Well: per-field VRMSE \downarrow scores.

	Field	VRMSE
MHD_64	velocity	$[0.0944]$ $[0.1007]$ $[0.1076]$
	density	$[0.1647]$
	magnetic	$[0.1368]$ $[0.1052]$ $[0.1028]$
shear_flow	velocity	$[0.0071]$ $[0.3038]$
	pressure	$[0.1193]$
	tracer	$[0.0189]$
rayleigh_benard	velocity	$[0.6221]$ $[0.3112]$
	pressure	$[0.0220]$
	buoyancy	$[0.0555]$
rayleigh_taylor	velocity	$[0.1603]$ $[0.1280]$
	density	$[0.0960]$ $[0.0618]$

Table 16: EddyFormer (Legendre basis) on The Well: per-field VRMSE \downarrow scores.

	Field	VRMSE
MHD_64	velocity	$[0.1056]$ $[0.1142]$ $[0.1161]$
	density	$[0.1926]$
	magnetic	$[0.1611]$ $[0.1219]$ $[0.1161]$
shear_flow	velocity	$[0.0004]$ $[0.1908]$
	pressure	$[0.1295]$
	tracer	$[0.0106]$
rayleigh_benard	velocity	$[0.6289]$ $[0.3063]$
	pressure	$[0.0279]$
	buoyancy	$[0.0807]$
rayleigh_taylor	velocity	$[0.1446]$ $[0.1560]$
	density	$[0.1284]$ $[0.0679]$

D Limitations

While EddyFormer accelerates the simulation of three-dimensional turbulence, its application to real-world turbulent flows in near-wall regions with fine features is left unexplored. In addition, scaling EddyFormer to accelerate simulations of extreme-scale turbulence at high Reynolds numbers is particularly challenging due to the complex nature of these flows. Future works include extending EddyFormer to general domains and exploiting its scalability.

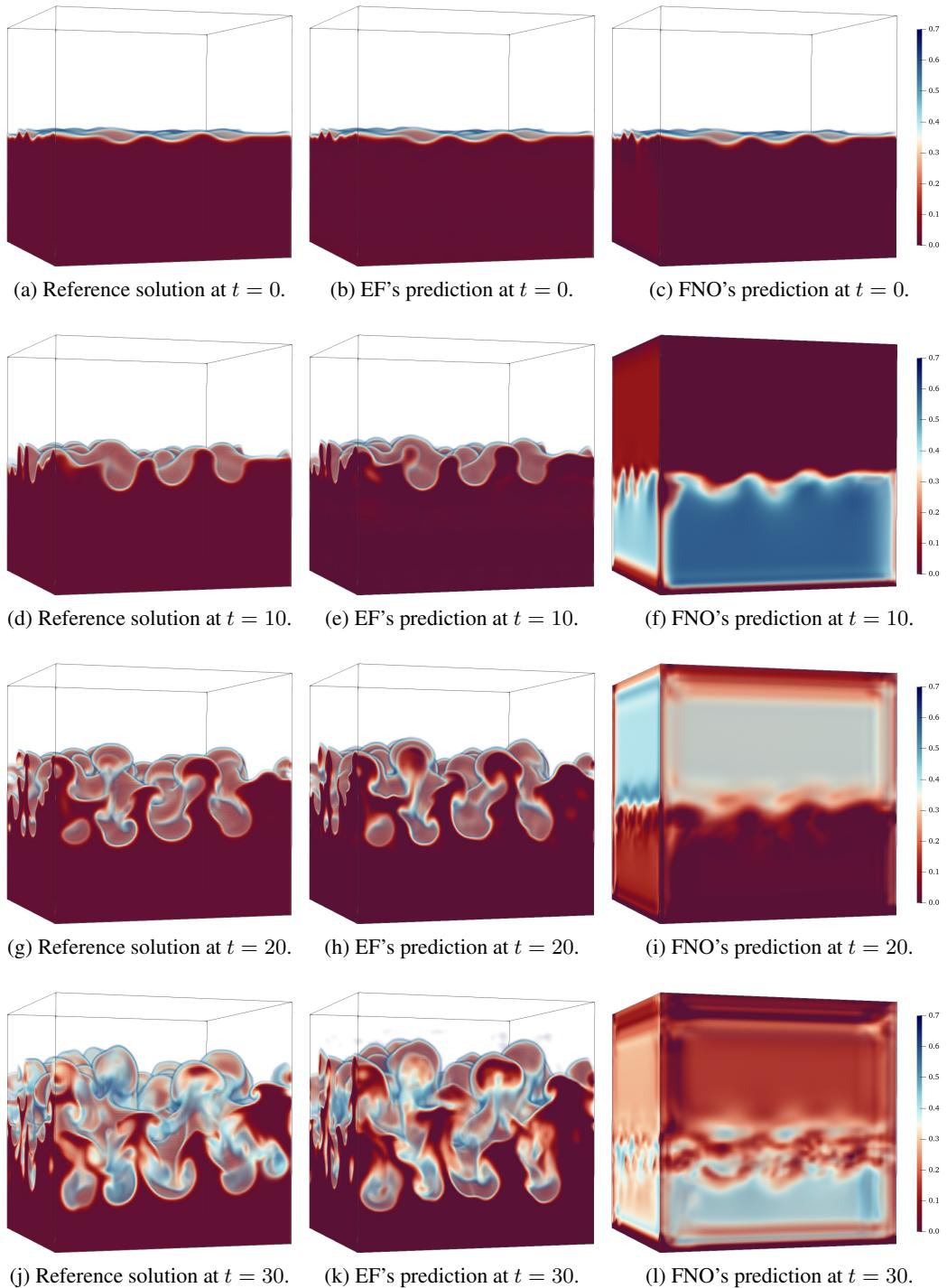


Figure 16: Visualization of the density evolution for a `rayleigh_taylor_instab.` test sample. EddyFormer successfully predicts the development of instability and the plume structures up to $t = 30$, while the baseline FNO fail to converge.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [Yes]

Justification: our experiments show that EddyFormer significantly accelerates the resolution of three-dimensional turbulence.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: we discussed that domain geometric effects are not explored in this work.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: this work doesn't include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: we include all architecture and experiment details in appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: all datasets and source code will be open sourced.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: all experiment details are in appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: error bars are not reported because it is expensive to repeat all experiments.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.).
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: we provide information on how long does each experiment take to reproduce on specific hardwares.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: we comply the NeurIPS code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: we discussed broader impacts of this work.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: this work poses no misuse risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: we cite all software assets used in this work.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. New assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: we provide detailed explanation for the three-dimensional turbulence dataset introduced in this work.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. Crowdsourcing and research with human subjects

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: this work does not involve crowdsourcing.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. Institutional review board (IRB) approvals or equivalent for research with human subjects

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: this work does not involve crowdsourcing.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. Declaration of LLM usage

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: this work does not involve LLMs.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.