

Conditional neural field for spatial dimension reduction of turbulence data: a comparison study

Junyi Guo^{a,1}, Pan Du^{b,1}, Xiantao Fan^{a,b}, Yahui Li^b, Jian-Xun Wang^{a,b,*}

^a*Sibley School of Mechanical and Aerospace Engineering, Cornell University, Ithaca, NY, USA*

^b*Department of Aerospace and Mechanical Engineering, University of Notre Dame, Notre Dame, IN*

Abstract

We investigate conditional neural fields (CNFs), mesh-agnostic, coordinate-based decoders conditioned on a low-dimensional latent, for spatial dimensionality reduction of turbulent flows. CNFs are benchmarked against Proper Orthogonal Decomposition and a convolutional autoencoder within a unified encoding–decoding framework and a common evaluation protocol that explicitly separates in-range (interpolative) from out-of-range (strict extrapolative) testing beyond the training horizon, with identical preprocessing, metrics, and fixed splits across all baselines. We examine three conditioning mechanisms: (i) activation-only modulation (often termed FiLM), (ii) low-rank weight + bias modulation (termed FP), and (iii) last-layer inner-product coupling, and introduce a novel domain-decomposed CNF that localizes complexities. Across representative turbulence datasets (WMLES channel inflow, DNS channel inflow, and wall pressure fluctuations over turbulent boundary layers), CNF–FP achieves the lowest training and in-range testing errors, while CNF–FiLM generalizes best for out-of-range scenarios once moderate latent capacity is available. Domain decomposition significantly improves out-of-range accuracy, especially for the more demanding datasets.. The study provides a rigorous, physics-aware basis for selecting conditioning, capacity, and domain decomposition when using CNFs for turbulence compression and reconstruction.

Keywords: Dimension Reduction, Turbulence, Domain-Decomposition, Conditional Neural

*Corresponding author. Tel: +1 540 3156512

¹Equal contribution

1. Introduction

Turbulent flows are characterized by high-dimensional, multi-scale spatiotemporal structures that pose significant challenges in computational fluid dynamics (CFD), both from computational and storage perspectives [1]. The detailed analysis, visualization, and interpretation of turbulence data obtained from high-fidelity numerical simulations, such as direct numerical simulation (DNS) or large-eddy simulation (LES), typically demand substantial computational resources. Consequently, there is a strong motivation to represent such high-dimensional data efficiently by encoding turbulent fields into compact, low-dimensional latent embeddings. These latent representations are valuable not only for flow modal analysis [2, 3], but also for downstream tasks such as reduced-order modeling [4–7], surrogate modeling [8–10], flow reconstruction [11–13], and optimization [14], all of which often require accurate inversion from latent space back to the physical domain.

Classical linear dimensionality reduction methods, notably Proper Orthogonal Decomposition (POD) [15, 16] and Dynamic Mode Decomposition (DMD) [17], provide well-established frameworks for projecting turbulent fields onto low-dimensional linear subspaces. POD identifies modes capturing maximal energy content, while DMD extracts coherent structures associated with characteristic frequencies and growth rates. Despite their utility and interpretability, these linear methods inherently struggle to represent strongly nonlinear features and multiscale structures of turbulence, often necessitating a prohibitively large number of modes to achieve acceptable accuracy [18, 19].

To better address nonlinear and complex turbulent structures, recent studies have increasingly explored nonlinear dimensionality reduction (NDR) techniques, offering a compelling alternative by discovering manifolds that better conform to the data’s intrinsic geometry. In fields like computer vision, techniques such as Kernel Principal Component Analysis (KPCA) [20], isometric mapping (Isomap) [21], Locally Linear Embedding (LLE) [22], and t-distributed stochastic neighbor embedding (t-SNE) [23] have shown success in unfolding

nonlinear manifolds [24]. In fluid dynamics, however, these manifold learning techniques have seen limited application until recently. One significant challenge is that many NDR methods lack a straightforward inverse mapping from latent space back to physical space - commonly known as the pre-image problem [25]. Without a reliable way to reconstruct the flow field from the reduced coordinates, these techniques are of limited use for compression or surrogate modeling. For example, KPCA can embed data nonlinearly, but computing an approximate inverse is non-trivial and often requires solving additional nonconvex optimization problems. Similarly, techniques like Isomap or t-SNE are primarily geared toward visualization or clustering, and do not provide an explicit decoder to generate flow fields from latents.

Recent advances in deep learning (DL) have introduced neural-network-based autoencoders, especially convolutional neural network autoencoders (CNN-AEs), which simultaneously learn nonlinear encoding and decoding mappings for effective compression and accurate reconstruction of turbulent fields [12, 26, 27]. For instance, Murata et al. [26] demonstrated that a CNN-AE significantly outperformed POD in reconstructing cylinder wake flows, achieving substantially lower errors for an equivalent latent dimensionality. CNN-AEs have further demonstrated efficacy in reconstructing high-fidelity fields from coarse simulations [11, 12] and in accelerating fluid dynamic simulations via latent-space dynamics forecasting [27, 28]. By leveraging convolutional layers, CNN-AEs efficiently capture spatial correlations inherent in data, thereby providing enhanced reconstruction fidelity compared to traditional linear methods. Nonetheless, CNN-AEs typically rely on structured grid data, which restricts their applicability to irregular, unstructured, or adaptively meshed flow domains frequently encountered in practical CFD applications.

A promising alternative is the emerging paradigm of neural field (NF) representations, coordinate-based neural networks that parameterize flow fields as continuous, implicit functions of spatial coordinates. Such representations are mesh-agnostic and naturally handle unstructured or adaptive grids – one can query field values at any coordinate, irrespective of how the training data were sampled [29]. They also provide implicit continuous resolution:

the network output can be evaluated on a finer grid than it was trained on, enabling super-resolution of the field without an explicit interpolation steps [30]. These features make NFs especially attractive for fluid dynamics, where geometric flexibility and multiscale resolution are often required. A conditional neural field (CNF) extends this concept to represent not just one fixed field but an entire family of fields conditioned on a latent code. In practice, a CNF is realized by augmenting the input of the NF network with a latent vector $\mathbf{z}$ that encodes the identity of a particular flow snapshot or flow condition. The latent $\mathbf{z}$ plays a role analogous to the code in an AE’s decoder – it is a compact description of the specific flow instance. The mapping from latent to physical space is deterministic and invertible in the sense that, given the trained CNFs, each $\mathbf{z}$ produces a unique field. Recovering $\mathbf{z}$ from a new field requires either an encoder network or optimization (i.e., auto-decoding) [31, 32]. This framework thus meets the criterion of invertible encoding by design.

Crucially, CNFs retain the advantages of implicit neural representations (INRs), as they do not require training data on a fixed mesh; one can train on a variety of discretizations or even point cloud samples of the field. Chen et al. [33, 34] highlight this in their continuous reduced-order modeling (CROM) approach: rather than building a basis for a fixed grid of PDE solutions, they construct a low-dimensional embedding of the continuous vector fields themselves using CNFs, enabling training on data from diverse grids and producing a single latent space characterizing the continuous solution manifold. Serrano et al. [35] further demonstrate the versatility of CNFs for operator learning in PDEs, showing their effectiveness in tackling PDEs defined on general geometries and highlighting their potential in broader scientific computing applications. Similarly, Yin et al. [36] leveraged these implicit neural representations to accurately forecast continuous PDE dynamics, effectively capturing both low-frequency and high-frequency content. Another strength of NFs, particularly relevant for turbulence, is the inherent capability to incorporate multi-scale details. Standard multilayer perceptrons with smooth activation functions exhibit a known spectral bias, prioritizing low-frequency (smooth) components and struggling with high-frequency content [37]. Turbulent fields, with their eddies and sharp gradients, inherently contain a

wide range of frequencies. To address this, recent advances like periodic activation functions (SIREN networks) [38] and Fourier feature embeddings [39] can be employed to better represent fine-scale structures. Pan et al. [40] demonstrated the efficacy of CNFs specifically for spatial dimension reduction and reconstruction of three-dimensional turbulent flows, highlighting their superior performance compared to classical linear methods. More recently, Du et al. [41] introduced CoNF FiLD , integrating latent diffusion models with CNFs to generate realistic spatiotemporal turbulence fields conditioned on partial or sparse observations, which has been successfully applied to inflow turbulence generation [42].

Despite these promising developments, CNFs remain relatively unexplored in turbulence modeling. They have yet to be systematically benchmarked against classical linear dimension reduction methods, such as POD, or widely-used nonlinear DL-based approaches, e.g., CNN-AEs. More importantly, generalizability, particularly extrapolation to flow conditions or time horizons beyond the training regime, has rarely been explicitly tested for most DL-based NDR models. Previous studies typically assess reconstruction accuracy primarily within training datasets or through interpolation over narrow parameter ranges [18, 40, 43]. However, real-world applications often demand robust performance under scenarios far outside the original training distribution. Another unresolved question pertains to how the conditioning mechanism (i.e., the way the latent code is incorporated into neural representations) impacts model performance for turbulent flows. While advanced conditioning strategies such as feature-wise linear modulation (FiLM) have demonstrated improved efficacy in certain contexts [41], their suitability and effectiveness specifically for turbulence representation remain underexplored. To address these gaps, we propose a unified framework for systematically comparing CNFs with classical linear and nonlinear ML-based dimension reduction approaches, enabling a consistent evaluation of reconstruction accuracy and generalization capabilities across diverse turbulence datasets. Furthermore, we introduce a novel domain-decomposition strategy within the CNF framework specifically tailored to handle large-scale, highly complex turbulent flow data, aiming to significantly enhance reconstruction accuracy and improve generalization. By explicitly testing interpolation and

extrapolation performance beyond the training horizons using both quantitative reconstruction metrics and qualitative assessments of physical fidelity, we investigate how structured latent-space architectures and domain decomposition can strengthen model robustness. Ultimately, this work seeks to establish CNFs, combined with our proposed improvements, as accurate, reliable, and practically viable NDR tools for turbulence data compression, reconstruction, and analysis.

The remainder of this paper is structured as follows: In section 2, we introduce CNFs for NDR with different conditioning mechanisms and formulate all baseline dimension reduction methods within a unified framework. In Section 3, we report the benchmarking results and explicitly compare the extrapolation and interpolation performance of each method. Finally, Section 4 discusses broader implications of our findings and outlines potential directions for future research.

2. Methodology

In this section, we present a unified framework for spatial dimension reduction and reconstruction of turbulent flow fields. We systematically introduce and compare classical POD, CNN-AEs, and CNF-based dimension reduction methods.

2.1. Unified framework for spatial dimension reduction

We introduce a generalized encoding-decoding framework for spatial dimension reduction of high-dimensional flow fields, structured to systematically represent a broad class of linear and nonlinear methods. This generic framework provides a clear mathematical foundation that covers the fundamental principles shared across diverse dimension reduction techniques, facilitating their comparative analysis and consistent evaluation.

In general, consider a spatiotemporal scalar field $q(\mathbf{x}, t)$, discretized in space and time forming the snapshot matrix $\Phi = [\phi^1, \phi^2, \dots, \phi^n] = [q(t_1), q(t_2), \dots, q(t_n)] \in \mathbb{R}^{m \times n}$, where m is the spatial dimension (e.g., the total number of grid points), and n denotes the number of temporal snapshots. The objective of spatial dimension reduction techniques is to identify a

compact, low-dimensional latent representation $\mathbf{Z} = [\mathbf{z}^1, \mathbf{z}^2, \dots, \mathbf{z}^n] \in \mathbb{R}^{r \times n}$ that effectively captures the primary features of the original high-dimensional fields, facilitating efficient storage, analysis, and reconstruction.

The dimension reduction process can be defined through two core mathematical operations: (1) an *encoding* operation, mapping the original field into a reduced latent space, and (2) a *decoding* operation, mapping the latent variables back into the original spatial representation.

Encoding operation. Let $\mathcal{E}$ denote a generic linear or nonlinear encoding operator parameterized by a set of parameters $\boldsymbol{\theta}^e$. This operator compresses each snapshot in the original spatiotemporal data Φ into its lower-dimensional latent representation $\mathbf{Z}$, defined as:

$$\mathbf{z}^i = \mathcal{E}(\boldsymbol{\phi}^i; \boldsymbol{\theta}^e), \quad \boldsymbol{\phi}^i \in \mathbb{R}^m, \mathbf{z}^i \in \mathbb{R}^r, i = 1, 2, \dots, n, r \ll m, \quad (1)$$

where the latent representation $\mathbf{z}^i$ encodes the essential spatial structures of the flow fields into a significantly reduced-dimensional form. We uniquely interpret the encoding process $\mathcal{E}$ as a two-step procedure. The first step, a transformation, maps the high-dimensional snapshot into a feature space that reorganizes the representation without loss of information, making it more amenable to compression. The second step, a reduction, projects this transformed representation onto a compact latent space, thereby discarding redundancy and retaining only the most essential flow structures.

- *Transformation step* ($\mathcal{T}^e$): This step maps the i^{th} data snapshot $\boldsymbol{\phi}^i$ into an intermediate feature representation $\boldsymbol{\gamma}^i$:

$$\boldsymbol{\gamma}^i = \mathcal{T}^e(\boldsymbol{\phi}^i; \boldsymbol{\theta}_T^e), \quad \boldsymbol{\gamma}^i \in \mathbb{R}^s, \quad (2)$$

where $\boldsymbol{\gamma}^i \in \mathbb{R}^s$ is an intermediate representation. Depending on the chosen method, the transformation may correspond, among others, to a linear projection (e.g., POD or Fourier bases), a nonlinear mapping (e.g., kernel embeddings), or a learned operator such as convolutional layers that capture localized flow features.

- *Reduction step* ($\mathcal{R}^e$): The subsequent reduction step explicitly projects the intermediate representation γ^i onto the lower-dimensional latent space:

$$\mathbf{z}^i = \mathcal{R}^e(\gamma^i; \theta_R^e), \quad \mathbf{z}^i \in \mathbb{R}^r. \quad (3)$$

Here, $\mathcal{R}^e$ may be realized through a variety of approaches, ranging from simple pooling or averaging operations, to rank-reducing linear transformations, to learned nonlinear projections implemented by neural networks.

The full encoding operation can be written as the composition of the transformation and reduction steps:

$$\mathbf{z}^i = \mathcal{R}_{\theta_R^e}^e \circ \mathcal{T}_{\theta_T^e}^e(\phi^i) = \mathcal{E}_{\theta^e}(\phi^i), \quad \theta^e = \{\theta_T^e, \theta_R^e\} \quad (4)$$

Decoding operation. The decoding operation defines an inverse mapping from the low-dimensional latent space $\mathbf{z}^i$ back to the original high-dimensional space. Formally, the decoding operator $\mathcal{D}$, parameterized by decoder parameters θ^d , is defined as:

$$\hat{\phi}^i = \mathcal{D}(\mathbf{z}^i; \theta^d), \quad \hat{\phi}^i \in \mathbb{R}^m, \quad i = 1, 2, \dots, n, \quad (5)$$

which produces reconstructed fields $\hat{\phi}^i$. Analogous to the encoder, the decoder $\mathcal{D}$ can also be decomposed into two counterpart sub-operators:

- *Reverse reduction step* ($\mathcal{R}^d$): This step maps the latent code $\mathbf{z}^i$ back into the intermediate representation space:

$$\hat{\gamma}^i = \mathcal{R}^d(\mathbf{z}^i; \theta_R^d), \quad \hat{\gamma}^i \in \mathbb{R}^s. \quad (6)$$

- *Reverse transformation step* ($\mathcal{T}^d$): Subsequently, this step maps the intermediate representation $\hat{\gamma}^i$ back to the original high-dimensional snapshot:

$$\hat{\phi}^i = \mathcal{T}^d(\hat{\gamma}^i; \theta_T^d), \quad \hat{\phi}^i \in \mathbb{R}^m. \quad (7)$$

Thus, the complete decoding process can be expressed explicitly as:

$$\hat{\phi}^i = \mathcal{T}_{\theta_T^d}^d \circ \mathcal{R}_{\theta_R^d}^d(\mathbf{z}^i) \quad (8)$$

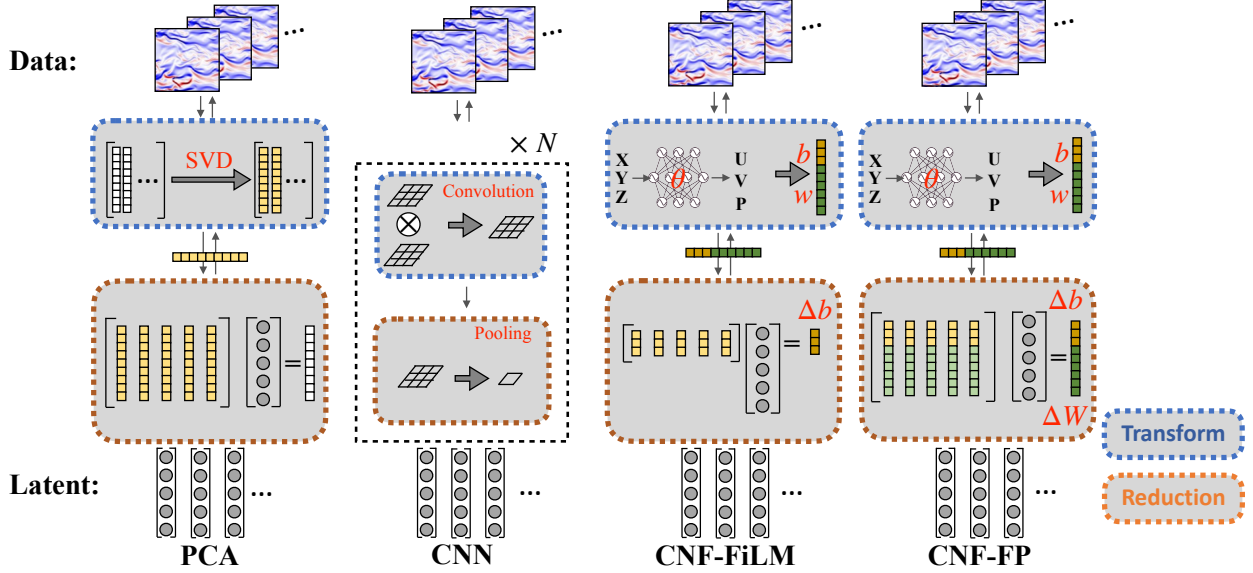


Figure 1: Schematic illustration of the unified encoding-decoding framework for spatial dimension reduction methods, demonstrating the common structure comprising transformation (blue dashed boxes) and reduction (orange dashed boxes) steps. Representative methods shown include linear (PCA/POD), convolutional neural network autoencoder (CNN-AE), and conditional neural fields (CNF-FiLM and CNF-FP)

Optimization. Given training data samples drawn from a distribution $\mathcal{G}$, the dimension reduction model parameters $\{\theta_T^e, \theta_R^e, \theta_T^d, \theta_R^d\}$ are identified through an optimization problem that seeks to minimize the reconstruction error between original and reconstructed fields:

$$\min_{\theta_T^e, \theta_R^e, \theta_T^d, \theta_R^d} \mathbb{E}_{\phi \sim \mathcal{G}} \left[\sum_{i=1}^n \mathcal{L}(\phi^i, \mathcal{D}_{\theta^d} \circ \mathcal{E}_{\theta^e}(\phi^i)) \right], \quad (9)$$

where $\mathcal{L}$ denotes a loss function that measures the discrepancy between the original snapshot and its reconstruction. Depending on the chosen method, $\mathcal{T}^e$, $\mathcal{R}^e$, and their decoder counterparts may represent linear transformations (e.g., singular value decomposition in POD), nonlinear mappings (e.g., kernel-based embeddings), or parameterized neural network layers (e.g., convolutional or fully connected layers in deep learning). This formulation thus provides a unifying mathematical framework that encompasses a wide range of dimensionality reduction methodologies, including classical POD, convolutional autoencoders (CNN-AEs), and variants of CNFs. As illustrated in Figure 1, these methods can all be represented in terms of a transformation step (blue dashed boxes) and a reduction step (orange dashed

boxes). This perspective highlights their shared conceptual foundations and enables direct comparison of their respective capabilities in dimensionality reduction and reconstruction fidelity. In the following subsections, each method will be formally introduced and analyzed within this unified encoding–decoding framework.

2.2. Baseline dimension reduction methods

2.2.1. Proper orthogonal decomposition (POD)

POD is classically defined via the singular value decomposition (SVD) of the snapshot matrix $\Phi \in \mathbb{R}^{m \times n}$ [44]:

$$\Phi = \mathbf{U} \Sigma \mathbf{V}^T, \quad (10)$$

where $\mathbf{U} \in \mathbb{R}^{m \times n}$ contains spatial orthonormal modes (columns), $\Sigma \in \mathbb{R}^{n \times n}$ holds singular values, and $\mathbf{V} \in \mathbb{R}^{n \times n}$ contains temporal right singular vectors. The POD coefficients for snapshot i are $\mathbf{a}^i = \Sigma \mathbf{V}^\top \mathbf{e}_i$ (equivalently $\mathbf{a}^i = \mathbf{U}^\top \phi^i$). Reinterpreting using our proposed framework, we can rewrite the encoding–decoding processes as follows:

Encoding operation. The encoding operation for POD is explicitly defined as,

- *Transformation step ($\mathcal{T}$):* The original snapshot ϕ^i is linearly projected onto the complete spatial POD basis $\mathbf{U}$, yielding an intermediate representation,

$$\gamma^i = \mathcal{T}^e(\phi^i; \theta_T^e) = \mathbf{U}^T \phi^i, \quad (11)$$

where the parameter θ_T^e is the POD spatial basis $\mathbf{U}$ itself.

- *Reduction step ($\mathcal{R}$):* The subsequent reduction step truncates the basis to the leading r dominant components of the intermediate representation γ^i , yielding the reduced-dimensional latent vector:

$$\mathbf{z}^i = \mathcal{R}^e(\gamma^i; \theta_R^e) = \gamma^i[:r], \quad \mathbf{z}^i \in \mathbb{R}^r, \quad (12)$$

where $\gamma^i[:r]$ denotes selecting the first r components of γ^i , corresponding to the dominant energetic features of the original flow data. In this classical linear setting, no parameters are involved in the reduction step (i.e., $\theta_R^e = \emptyset$).

Decoding operation. The decoding operation reconstructs the snapshot by linearly combining the retained orthonormal spatial modes with the latent coefficients $\mathbf{z}(t)$.

- *Reverse reduction step* ($\mathcal{R}^d$): Given the latent vector $\mathbf{z}^i$, the intermediate representation is reconstructed by padding zeros to restore the original dimension n :

$$\hat{\gamma}^i = \mathcal{R}^d(\mathbf{z}^i) = \begin{bmatrix} \mathbf{z}^i & \mathbf{0} \end{bmatrix}, \quad \hat{\gamma}^i \in \mathbb{R}^n, \quad (13)$$

where zeros are padded to the latent vector to restore its dimension to match the full set of spatial modes.

- *Reverse transformation step* ($\mathcal{T}^d$): The reconstructed high-dimensional field is obtained by linearly combining the spatial POD modes

$$\hat{\phi}^i = \mathcal{T}^d(\hat{\gamma}^i; \boldsymbol{\theta}_T^d) = \mathbf{U} \hat{\gamma}^i, \quad \hat{\phi}^i \in \mathbb{R}^m, \quad (14)$$

where $\boldsymbol{\theta}_T^d = \boldsymbol{\theta}_T^e$ is also the POD basis $\mathbf{U}$.

Optimization problem. In the context of POD, the optimization problem seeks an r -dimensional basis that minimizes the projection error, measured in the squared Frobenius norm of the snapshot matrix:

$$\min_{\mathbf{U}_r} \|\Phi - \mathbf{U}_r \mathbf{U}_r^T \Phi\|_F^2, \quad (15)$$

whose solution, by the Eckart–Young–Mirsky theorem [45], is given by the truncated SVD of Φ . This formulation allows POD to be reinterpreted within the unified encoding–decoding framework, facilitating direct comparison with modern deep learning–based dimensionality reduction approaches such as CNN-AEs and CNFs.

2.2.2. Convolutional autoencoder (CNN-AE)

The CNN-AE performs spatial dimensionality reduction and reconstruction progressively. At each stage, *transformation* (convolutional message passing) and *reduction* (stride/pooling) are interleaved; Likewise, the decoding process interleaves reverse reduction (upsampling) and reverse transformation (feature reconstruction) steps. we model this with *micro-blocks* that fit cleanly into the unified framework.

Encoding operation. Let $H^{(0)} = \text{reshape}(\phi^i)$ be the input feature map. For $\ell = 1, \dots, L$ we apply a micro-block

$$\tilde{H}^{(\ell)} = \underbrace{\mathcal{T}^{e,\ell}(H^{(\ell-1)}; \theta_T^{e,\ell})}_{\text{message passing: conv/BN/activation}}, \quad H^{(\ell)} = \underbrace{\mathcal{R}^{e,\ell}(\tilde{H}^{(\ell)})}_{\text{reduction: stride or pooling}}, \quad (16)$$

and denote $\mathcal{B}_\ell := \mathcal{R}^{e,\ell} \circ \mathcal{T}^{e,\ell}$. After L blocks we obtain an intermediate vector

$$\gamma^i = \text{vec}(\mathcal{B}_L \circ \dots \circ \mathcal{B}_1(\phi^i)) \in \mathbb{R}^s, \quad (17)$$

which is projected to the latent space by a bottleneck map:

$$\mathbf{z}^i = \mathcal{R}^{e,\text{bottleneck}}(\gamma^i; \theta_R^e) \in \mathbb{R}^r. \quad (18)$$

Remark. While downsampling reduces spatial resolution locally at each stage, we treat these as *micro-reductions* inside the hierarchical $\mathcal{T}^e$ pathway and reserve $\mathcal{R}^{e,\text{bottleneck}}$ for the final projection to dimension r .

Decoding operation. Starting from the latent, we first expand back to the encoder’s terminal feature size and reshape:

$$\hat{\gamma}^i = \mathcal{R}^{d,\text{expand}}(\mathbf{z}^i; \theta_R^d) \in \mathbb{R}^s \xrightarrow{\text{reshape}} \hat{H}^{(L)}. \quad (19)$$

Then for $\ell = L, \dots, 1$ we mirror the micro-blocks with upsampling (or transposed stride) followed by convolution:

$$\tilde{H}^{(\ell-1)} = \underbrace{\mathcal{R}^{d,\ell}(\hat{H}^{(\ell)})}_{\text{upsample / transposed stride}}, \quad \hat{H}^{(\ell-1)} = \underbrace{\mathcal{T}^{d,\ell}(\tilde{H}^{(\ell-1)}; \theta_T^{d,\ell})}_{\text{conv/BN/activation}}. \quad (20)$$

Finally, $\hat{\phi}^i = \text{reshape}^{-1}(\hat{H}^{(0)}) \in \mathbb{R}^m$. (Skip connections, if used, are part of $\mathcal{T}^{e,\ell}/\mathcal{T}^{d,\ell}$.) If using ConvTranspose layers, $\mathcal{R}^{d,\ell}$ and $\mathcal{T}^{d,\ell}$ can be implemented as a single deconvolutional operator.

Optimization Problem. The optimization problem for CNN-AE is consistent with the general framework presented in equation 9. The model parameters, which include all weights and biases within the encoder and decoder networks $\{\theta_T^e, \theta_R^e, \theta_T^d, \theta_R^d\}$, are learned by minimizing

the Mean-Squared Error (MSE) between the original and reconstructed fields. Unlike POD, which admits a closed-form solution via linear algebra factorization (i.e., SVD), CNN-AE training requires an iterative optimization procedure, typically carried out using stochastic gradient descent (SGD) or its variants.

2.3. Conditional neural fields (CNF)

2.3.1. Neural field representation

Neural fields (also called INRs) are continuous functions parameterized by neural networks that map spatial coordinates to physical quantities of interest. Given coordinates of n_v points in d -dimensional space, $\mathbf{X} \in \mathbb{R}^{n_v \times d}$ a single-snapshot field is represented as,

$$\phi \approx f(\mathbf{X}; \boldsymbol{\theta}), \quad (21)$$

where $f(\cdot; \boldsymbol{\theta})$ is a neural network with parameters $\boldsymbol{\theta}$. Once trained, the continuity of f enables evaluation at arbitrary $\mathbf{x}$, supporting interpolation and super-resolution.

2.3.2. CNF for dimension reduction

A naive extension to multiple snapshots would train one NF per snapshot, $f(\cdot; \boldsymbol{\theta}^i)$, which is computationally costly and ignores redundancy across time. In this work, we design a spatial dimension-reduction method that uses a single conditional neural field (CNF) as the decoder and auto-decoding [31, 32] to obtain latents. Specifically, a shared base network $f(\cdot; \boldsymbol{\theta})$ captures global structure, and a low-dimensional latent $\mathbf{z}^i \in \mathbb{R}^r$ modulates a subset of parameters via a linear/nonlinear projection:

$$\phi^i \approx f(\mathbf{X}^i; \boldsymbol{\theta} + \Delta\boldsymbol{\theta}(\mathbf{z}^i)), \quad (22)$$

with $\mathbf{X}^i \in \mathbb{R}^{n_v^i \times d}$. We denote by h the number of scalars actually modulated; typically $\Delta\boldsymbol{\theta}(\mathbf{z}^i) = \mathbf{M}\mathbf{z}^i$ with $\mathbf{M} \in \mathbb{R}^{h \times r}$, possibly organized per layer (block-diagonal, low-rank, or hypernetwork-generated). CNFs thus realize spatial dimension reduction by mapping each high-dimensional snapshot to a compact latent $\mathbf{z}^i$ while retaining an explicit, continuous decoder back to physical space.

Encoding operation (auto-decoder).

- *Transformation step* ($\mathcal{T}^e$): Conceptually, fitting an NF to $(\mathbf{X}^i, \phi^i)$ yields an implicit parameter vector γ^i in parameter space that best matches the snapshot:

$$\gamma^i = \mathcal{T}^e(\mathbf{X}^i, \phi^i; \theta_T^e = \emptyset) = \arg \min_{\tilde{\gamma}} \|\phi^i - f(\mathbf{X}^i; \tilde{\gamma})\|_2^2. \quad (23)$$

We do not solve (23) explicitly; it formalizes that the encoders intermediate representation γ^i lives in *parameter space*.

- *Reduction step* ($\mathcal{R}^e$): We relate this implicit parameter vector to the latent via the linear map $\mathbf{M}$:

$$\mathbf{z}^i = \mathcal{R}^e(\gamma^i; \theta_R^e) = \arg \min_{\mathbf{z} \in \mathbb{R}^r} \|\gamma^i - (\theta + \mathbf{M}\mathbf{z})\|_2^2 \approx \mathbf{M}^\dagger(\gamma^i - \theta), \quad (24)$$

where $\mathbf{M}^\dagger$ denotes a (regularized) pseudoinverse. *In practice we directly optimize $\mathbf{z}^i$ by minimizing reconstruction loss* (auto-decoding), which is numerically preferable to explicitly forming γ^i :

$$\mathbf{z}^i = \arg \min_{\mathbf{z} \in \mathbb{R}^r} \mathcal{L}(\phi^i, f(\mathbf{X}^i; \theta + \mathbf{M}\mathbf{z})). \quad (25)$$

This realizes the encoder $\mathcal{E}$ as an *optimization operator* $\mathcal{E} : (\mathbf{X}^i, \phi^i) \mapsto \mathbf{z}^i$. (An amortized encoder $\mathcal{E}_\psi$ can be used instead; we adopt auto-decoding here.)

Decoding operation.

- *Reverse reduction step* ($\mathcal{R}^d$): Given a low-dimensional latent vector $\mathbf{z}^i \in \mathbb{R}^r$, we expand it into higher-dimensional localized parameters corresponding to the snapshot-specific parameter shift $\theta + \Delta\theta^i$ via a linear transformation parameterized by $\mathbf{M} \in \mathbb{R}^{h \times r}$,

$$\hat{\gamma}^i = \mathcal{R}^d(\mathbf{z}^i; \theta_R^d) = \theta + \mathbf{M}\mathbf{z}^i \in \mathbb{R}^h, \quad (26)$$

where θ_R^d consists of the fixed base neural field parameters θ and the learned linear mapping matrix $\mathbf{M}$.

- *Reverse transformation step* ($\mathcal{T}^d$): The reverse transformation step for CNF is simply forward evaluation of the neural network, leading to the reconstructed field:

$$\hat{\phi}^i = \mathcal{T}^d(\hat{\gamma}^i; \theta_T^d) = f(\mathbf{X}^i; \hat{\gamma}^i), \quad (27)$$

where $\theta_T^d = \emptyset$ contains no parameters, but solely the predefined computational operations within the base neural network.

Optimization problem. The training objective jointly optimizes $(\theta, \mathbf{M})$ and the latents $\mathbf{Z} = \{\mathbf{z}^i\}_{i=1}^n$:

$$\min_{\mathbf{z}, \theta, \mathbf{M}} \frac{1}{n} \sum_{i=1}^n \mathcal{L}(\phi^i, f(\mathbf{X}^i; \theta + \mathbf{M}\mathbf{z}^i)) + \lambda_z \frac{1}{n} \sum_{i=1}^n \|\mathbf{z}^i\|_2^2, \quad (28)$$

where $\mathcal{L}$ is typically an L^2 or relative error on the field values, and $\lambda_z \geq 0$ regularizes latents for stability/identifiability. At inference, we freeze the decoder and solve for the test latent:

$$\mathbf{z}_{\text{test}}^* = \arg \min_{\mathbf{z}} \mathcal{L}(\phi_{\text{test}}, f(\mathbf{X}_{\text{test}}; \theta^* + \mathbf{M}^*\mathbf{z})), \quad (29)$$

and then reconstruct $\hat{\phi}_{\text{test}} = f(\mathbf{X}_{\text{test}}; \theta^* + \mathbf{M}^*\mathbf{z}_{\text{test}}^*)$, where $(*)$ denotes the optimal values after training.

Remark. Unlike a standard encoder–decoder, there is no explicit encoder $\mathcal{E}$ that maps high-dimensional fields Φ to latents using *auto-decoder* formulation. Instead, the latents for the training snapshots, $\mathbf{Z} = [\mathbf{z}^1, \dots, \mathbf{z}^n]$, are introduced as free learnable variables and are optimized jointly with the shared base-network parameters θ and the conditioning-module parameters. Thus, “encoding” is realized implicitly by optimization rather than by a separate operator. At inference, given an unseen snapshot, its latent $\mathbf{z}_{\text{test}}$ is obtained by solving a small optimization problem with base network held fixed.

2.3.3. Conditioning mechanisms

Conditioning specifies how auxiliary context modulates the decoder so that the predicted field depends on both spatial coordinates and context. We write the generic conditioned mapping as

$$\hat{\phi} = f(\mathbf{X}, \mathbf{C}; \theta) \quad (30)$$

where $\mathbf{X}$ denotes spatial coordinates and $\mathbf{C}$ carries conditioning information. In our CNF, the conditioning variable is the latent $\mathbf{z}$; conditioning is realized either by *parameter modulation* $\Delta\theta(\mathbf{z})$ of the base network or via a *last-layer coupling* (DeepONet-style inner product) without modifying internal weights.

A simple historical baseline of conditioning is *concatenation*, which appends $\mathbf{C}$ to the input or intermediate features. While easy to implement and inexpensive, concatenation typically induces only weak interactions between $\mathbf{X}$ and $\mathbf{C}$ and often underperforms when strong, structured coupling is required.

To obtain stronger inductive bias and controllable capacity, we adopt the three mechanisms illustrated in Figure 2. Below we give their layer-wise forms and brief context. Let layer ℓ have pre-activation $u^{(\ell)} = W^{(\ell)}h^{(\ell)} + b^{(\ell)}$, activation $h^{(\ell+1)} = \rho(u^{(\ell)})$, widths $d_\ell, d_{\ell+1}$, and denote $M_B^{(\ell)} \in \mathbb{R}^{d_{\ell+1} \times r}$ and $M_W^{(\ell)} \in \mathbb{R}^{(d_{\ell+1} \times d_\ell) \times r}$ as the latent-to-bias projection and the latent-to-weights projection respectively.

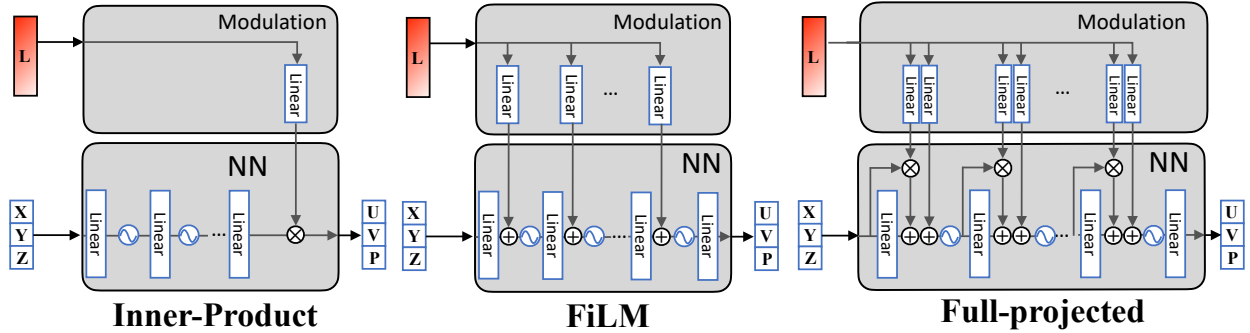


Figure 2: Diagram of different conditioning mechanism

- CNF-FiLM (feature-wise linear modulation). FiLM-style activation modulation is widely used in conditional representation learning because it injects sample-dependent information with minimal memory/latency overhead and good training stability. In NFs, it preserves the base operator while allowing snapshot-specific shifts at each layer. Specifically, we keep weights fixed and modulate activations using latent-dependent shifts (and

optionally gains). Our default bias-only variant reads

$$h^{(\ell+1)} = \rho\left(W^{(\ell)}h^{(\ell)} + b^{(\ell)} + \underbrace{M_B^{(\ell)}\mathbf{z}}_{\Delta b^{(\ell)}(\mathbf{z})}\right). \quad (31)$$

Complexity is $O(d_{\ell+1}r)$ parameters per layer and negligible runtime overhead. We zero-initialize $M_B^{(\ell)}$ so that $\mathbf{z} = \mathbf{0}$ recovers the base network.

- CNF-FP (full-projected weight+bias modulation). We generalize FiLM by permitting latent-driven weight updates. It is analogous to hypernetwork-style adaptation used to increase expressivity while controlling parameter growth. The added flexibility often improves fit but can increase overfitting and computation overhead. We allow latent-driven updates of both weights and biases:

$$\begin{aligned} u^{(\ell)}(\mathbf{z}) &= (W^{(\ell)} + \Delta W^{(\ell)}(\mathbf{z}))h^{(\ell)} + (b^{(\ell)} + \Delta b^{(\ell)}(\mathbf{z})), \\ \Delta b^{(\ell)}(\mathbf{z}) &= M_B^{(\ell)}\mathbf{z}, \quad \Delta W^{(\ell)}(\mathbf{z}) = M_W^{(\ell)}\mathbf{z}. \end{aligned} \quad (32)$$

We zero-initialize $M_B^{(\ell)}$ and $M_W^{(\ell)}$ so that $\mathbf{z} = \mathbf{0}$ recovers the base network.

- CNF-Inner (DeepONet-style last-layer coupling). Inner-product coupling is standard in branch-trunk operator learning: a trunk encodes coordinates, a branch encodes context, and a final inner product yields the output. It is memory/latency efficient and naturally rank-controlled by r , but lacks internal parameter adaptation. Internal weights are not modulated. A trunk produces $\psi(\mathbf{x}) \in \mathbb{R}^r$ and a branch maps the latent $b(\mathbf{z}) \in \mathbb{R}^r$ (often $b(\mathbf{z}) = \mathbf{z}$ or $b(\mathbf{z}) = \mathbf{B}\mathbf{z}$); the output is

$$f(\mathbf{x}; \mathbf{z}) = \psi(\mathbf{x})^\top b(\mathbf{z}) \quad (\text{or } f(\mathbf{x}; \mathbf{z}) = \Psi(\mathbf{x})b(\mathbf{z}) \text{ for multi-channel outputs}). \quad (33)$$

CNF-FiLM and CNF-FP implement the decoder’s reverse-reduction $\mathcal{R}^d : \mathbf{z} \mapsto \boldsymbol{\theta} + \Delta\boldsymbol{\theta}(\mathbf{z})$ in parameter space, followed by forward evaluation $\mathcal{T}^d$, whereas CNF-Inner realizes conditioning entirely inside $\mathcal{T}^d$ with a minimal $\mathcal{R}^d$. These mechanisms span a practical capacity–cost trade space, and we benchmark all three in Section 3.

2.3.4. Domain-decomposition for CNFs

High-fidelity turbulent snapshots routinely contain 10^6 – 10^8 spatial degrees of freedom. With $\mathcal{O}(10^2$ – $10^3)$ snapshots available for training, a single global CNF must explain all multi-scale variability with one latent per snapshot and therefore tends to overfit and generalize poorly. To improve the bias-variance tradeoff, we introduce a *domain-decomposed CNF* that conditions locally while sharing a global base network.

Let the domain be partitioned into P subdomains (patches) $\{\Omega_p\}_{p=1}^P$ with $\bigcup_{p=1}^P \Omega_p = \Omega$ (non-overlapping in our default, though the formulation also supports overlaps). For snapshot i we associate a *patch latent* $\mathbf{z}_p^i \in \mathbb{R}^r$ with each Ω_p . All patches share the same decoder parameters $\boldsymbol{\theta}$ and conditioning projection $\mathbf{M} \in \mathbb{R}^{h \times r}$ learned from data. We also define a local coordinate normalization $\tilde{\mathbf{x}} = \mathcal{N}_p(\mathbf{x}) \in [-1, 1]^d$ for $\mathbf{x} \in \Omega_p$ to improve conditioning of the SIREN/MLP. Within patch Ω_p the decoder evaluates

$$\hat{\phi}_p^i(\mathbf{x}) = f(\tilde{\mathbf{x}}; \boldsymbol{\theta} + \Delta\boldsymbol{\theta}(\mathbf{z}_p^i)), \quad \Delta\boldsymbol{\theta}(\mathbf{z}_p^i) = \mathbf{M}\mathbf{z}_p^i, \quad \mathbf{x} \in \Omega_p, \quad (34)$$

i.e., the localized reverse-reduction $\mathcal{R}^d : \mathbf{z}_p^i \rightarrow \boldsymbol{\theta} + \mathbf{M}\mathbf{z}_p^i$ followed by the forward evaluation $\mathcal{T}^d$. For a non-overlapping tiling we assemble the global prediction by restriction, $\hat{\phi}^i(\mathbf{x}) = \hat{\phi}_p^i(\mathbf{x})$ if $\mathbf{x} \in \Omega_p$. For overlapping tiles, weighted summation is needed (not pursued in this study).

Let $\mathcal{S}_p^i \subset \Omega_p$ be the training samples for patch p in snapshot i . We minimize the sum of patchwise reconstruction errors, with optional interface and smoothness regularization:

$$\begin{aligned} \min_{\boldsymbol{\theta}, \mathbf{M}, \{\mathbf{z}_p^i\}} & \sum_i \sum_p \mathbb{E}_{\mathbf{x} \in \mathcal{S}_p^i} \left\| \phi^I(\mathbf{x}) - \hat{\phi}_p^I(\mathbf{x}) \right\|_2^2 + \lambda_{\text{int}} \sum_i \sum_{(p,q) \in \mathcal{E}} \mathbb{E}_{\mathbf{x} \in \mathcal{S}_{pq}^I} \left\| \hat{\phi}_p^I(\mathbf{x}) - \hat{\phi}_q^I(\mathbf{x}) \right\|_2^2 \\ & + \lambda_{\text{spa}} \sum_i \sum_{(p,q) \in \mathcal{E}} \left\| \mathbf{z}_p^I - \mathbf{z}_q^I \right\|_2^2 + \lambda_{\text{tem}} \sum_{p,i} \left\| \mathbf{z}_p^{i+1} - \mathbf{z}_p^I \right\|_2^2, \end{aligned} \quad (35)$$

where $\mathcal{E}$ indexes neighboring patches, $\mathcal{S}_{pq}^I \subset \Omega_p \cap \Omega_q$ samples the interface (used only if seam suppression is desired), and $\lambda_{\text{int}}, \lambda_{\text{spa}}, \lambda_{\text{tem}} \geq 0$ control the strength of the regularizers.

The parameter count of the shared decoder $(\boldsymbol{\theta}, \mathbf{M})$ is independent of P ; only the per-snapshot latents scale as $O(Pr)$. Smaller patches reduce local complexity and improve extrapolation (especially near walls) at the cost of more latents and patch evaluations; In practice, we

choose uniform tiling of equal-sized patches and train with balanced mini-batches across patches. While anisotropic tiling is compatible with our formulation and may further improve accuracy for inhomogeneous flows, but we leave such adaptivity to future work.

3. Numerical Results

3.1. Experimental setup

We evaluate spatial dimension reduction and reconstruction across time by training on a subset of snapshots and assessing accuracy on *unseen* snapshots drawn from two disjoint regimes: *interpolation* (in-range) and *extrapolation* (out-of-range). Let $\mathbf{q}(\mathbf{x}, t)$ denote the spatiotemporal field with spatial coordinate $\mathbf{x} \in \Omega$ and time $t \in [0, T+T']$.

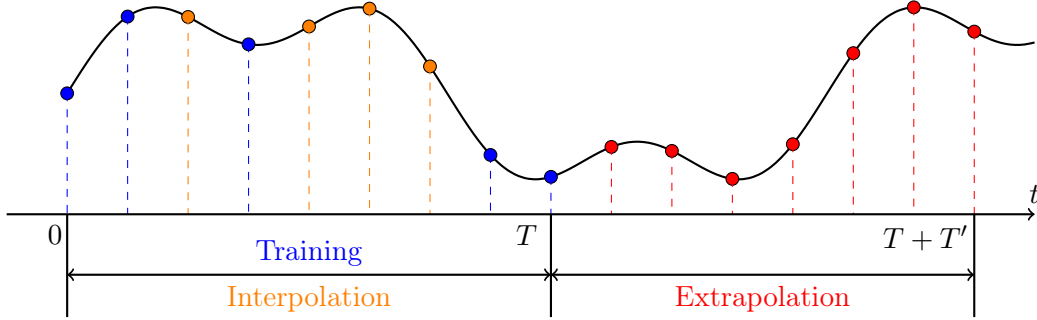


Figure 3: Demo diagram of dataset splitting strategy

The training dataset, $\mathcal{D}_{\text{train}}$, is defined as:

$$\mathcal{D}_{\text{train}} = \{\mathbf{q}(\mathbf{x}, t) \mid t \in \mathcal{T}_{\text{train}}, \mathbf{x} \in \Omega\}, \quad (36)$$

where $\mathcal{T}_{\text{train}} \subset [0, T)$ is a randomly sampled subset of the time domain $[0, T)$. This dataset is used to construct the dimension reduction models.

To evaluate in-range generalization (interpolation within the training horizon), we define the interpolative test set as:

$$\mathcal{D}_{\text{interp}} = \{\mathbf{q}(\mathbf{x}, t) \mid t \in \mathcal{T}_{\text{interp}}, \mathbf{x} \in \Omega\}, \quad (37)$$

where $\mathcal{T}_{\text{interp}}$ is a set of time indices drawn from $[0, T)$ with $\mathcal{T}_{\text{train}} \cap \mathcal{T}_{\text{interp}} = \emptyset$. Thus, every interpolation snapshot is *unseen* yet lies within the training time span $[0, T)$ (in-distribution), i.e., at disjoint indices but not beyond the training horizon.

To assess out-of-range generalization beyond the training horizon, we define the extrapolative dataset as:

$$\mathcal{D}_{\text{extrap}} = \{\mathbf{q}(\mathbf{x}, t) \mid t \in [T, T + T'), \mathbf{x} \in \Omega\}. \quad (38)$$

Unless otherwise noted, all quantitative claims of “extrapolation” refer strictly to $\mathcal{D}_{\text{extrap}}$.

Most ML-based dimensionality-reduction studies for turbulence evaluate on *interpolative* test sets, held-out samples whose indices remain within the training range, rather than on truly out-of-range data. In such settings, test snapshots are *in-distribution*, which partially explains why many methods report strong performance. To make this distinction explicit and fair, we evaluate and report both protocols side by side, using identical preprocessing, metrics, and evaluation grids across splits; the same $(\mathcal{T}_{\text{train}}, \mathcal{T}_{\text{interp}}, \mathcal{T}_{\text{extrap}})$ indices are fixed and shared across all baselines for comparability.

We consider three datasets: (i) velocity fields on planes perpendicular to the streamwise direction from turbulent channel flows (DNS), (ii) wall pressure fluctuations over a turbulent flat boundary layer, and (iii) inlet streamwise velocity for turbulent channel flows from wall modeled LES (WMLES). Each dataset includes uniformly sampled time snapshots over $[0, T + T')$ and fields stored on their native evaluation grids. Details of data generation (governing equations, Reynolds numbers, numerical schemes, discretizations, sampling cadence, and boundary conditions) are provided in [Appendix A](#).

3.2. Benchmark study against linear and DL baselines

We first compare CNFs with different conditioning mechanisms (FiLM, FP, and inner product) to widely used dimensionality-reduction baselines under identical splits and metrics on the WMLES–Inlet dataset. Table 1 reports relative L^2 errors across latent sizes $r \in \{8, 16, 32, 64, 128, 256\}$ and splits (training, interpolation, strict extrapolation).

Latent size	Split	POD	ConvAE	CNF-FP	CNF-FiLM	CNF-Inner
8	Training	4.78%	3.82%	1.17%	2.48%	5.66%
	Interpolation	4.83%	4.04%	2.27%	3.14%	5.58%
	Extrapolation	5.74%	6.14%	6.60%	6.22%	6.00%
16	Training	4.21%	3.28%	0.81%	2.26%	5.65%
	Interpolation	4.32%	3.50%	1.06%	2.60%	5.66%
	Extrapolation	5.49%	5.87%	6.01%	5.80%	6.02%
32	Training	3.46%	2.86%	0.46%	2.10%	5.65%
	Interpolation	3.64%	3.10%	0.73%	2.37%	5.66%
	Extrapolation	5.01%	5.43%	5.21%	5.05%	6.02%
64	Training	2.54%	2.19%	0.39%	1.73%	5.65%
	Interpolation	2.81%	2.45%	0.54%	2.02%	5.66%
	Extrapolation	4.48%	4.98%	4.37%	4.12%	6.02%
128	Training	1.52%	1.76%	0.21%	1.37%	5.65%
	Interpolation	1.85%	1.99%	0.36%	1.64%	5.66%
	Extrapolation	3.80%	4.80%	3.61%	3.10%	6.02%
256	Training	0.55%	1.39%	0.10%	1.21%	5.65%
	Interpolation	0.84%	1.65%	0.28%	1.36%	5.66%
	Extrapolation	3.09%	4.82%	2.82%	2.04%	6.01%

Table 1: Relative L^2 errors (%) on the WMLES-Inlet dataset across latent sizes and evaluation splits.

Focusing on *fitting capability* (training), errors decrease with latent size for all methods, and CNF-FP is the most accurate at every r (e.g., 0.46% at $r=32$, 0.21% at $r=128$, 0.10% at $r=256$), followed by CNF-FiLM (2.10%, 1.37%, 1.21%) and ConvAE (2.86%, 1.76%, 1.39%); POD improves to 0.55% at $r=256$ but remains above CNF-FP, and CNF-Inner stays near 5.65%, indicating persistent underfitting. On *in-distribution testing* (interpolation), the ranking persists: CNF-FP attains the lowest errors throughout, CNF-FiLM is next, then ConvAE; POD improves with capacity, while CNF-Inner remains $\approx 5.66\%$. For *out-of-range testing*, the picture changes: once $r \geq 64$, CNF-FiLM provides the best accuracy among learnable decoders, with errors 4.12% ($r=64$), 3.10% ($r=128$), and 2.04% ($r=256$), outperforming CNF-FP (4.37%, 3.61%, 2.82%), ConvAE (4.98%, 4.80%, 4.82%, which plateaus), and POD at the same r ; at very small latent sizes ($r \leq 32$) the linear POD baseline is competitive in L^2 , consistent with its bias toward energetic low-wavenumber

content. Generalization gaps at $r=128$ further quantify the trade-off: extrapolation minus training increases by +1.73 for CNF-FiLM ($1.37 \rightarrow 3.10\%$) versus +3.40 for CNF-FP ($0.21 \rightarrow 3.61\%$), with POD and ConvAE in between (+2.28 and +3.04); thus, while CNF-FP offers the strongest data fitting and in-range reconstruction, activation-only modulation (CNF-FiLM) yields more stable out-of-range generalization once moderate capacity is available. Figure D.12 in Appendix complements these findings with side-by-side reconstructions and absolute-error maps at $r=128$. Consistent with the quantitative results, CNF-FiLM presents the best out-of-range performance with reduced small-scale errors, CNF-FP is visually sharp in-range but degrades more beyond the horizon, POD appears smoother, and ConvAE sits between POD and CNFs.

While the relative L^2 error comparisons quantify pointwise reconstruction accuracy, they are agnostic to flow physics and can obscure scale-dependent errors. We therefore investigate the turbulence statistics of the reconstructed fields. Figure 4 presents wall-normal profile of root mean square (rms) of streamwise-velocity fluctuations $C_{u,\text{rms}}(y^+)$ for training, interpolation, and extrapolation at representative latent sizes, providing a physics-grounded view of how each model recovers the near-wall peak and outer-layer decay. Increasing the latent dimension improves agreement with the ground truth in all regimes: the near-wall peak and the outer-layer decay are progressively recovered as r grows from 8 to 256. In the *out-of-range testing* panel (Fig. 4c), POD exhibits the largest deficit across y^+ , particularly around the peak region, despite its relatively low L^2 error at small r (Table 1); this reflects POD’s bias toward energetic low-wavenumber content that suppresses small-scale intensity. By contrast, CNF decoders move closer to the ground truth with capacity: at $r = 64$ and 256, CNF-FiLM tracks both the magnitude and the position of the $C_{u,\text{rms}}$ peak most closely under extrapolation, while CNF-FP, which is the best fitter on the training and in-distribution testing, retains a small but visible underprediction in the outer region. ConvAE improves with r yet consistently underperforms than CNFs (FiLM and FP), and CNF-Inner changes little with capacity, consistent with its flat L^2 performance. Taken together, these diagnostics confirm that the apparent L^2 advantage of linear POD at small r does not translate to

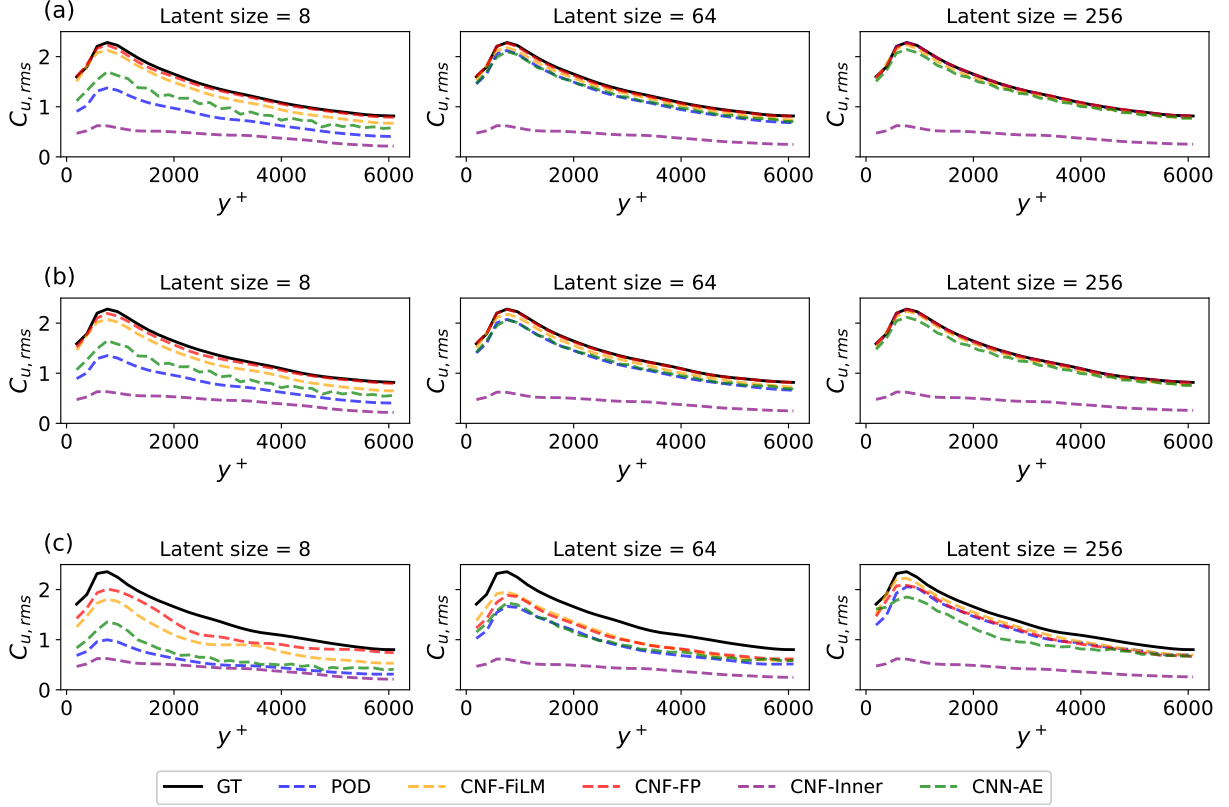


Figure 4: Wall-normal profiles of the normalized streamwise-velocity fluctuation RMS, $C_{u,rms}(y^+)$, on the WMLES-Inlet dataset. Columns correspond to latent sizes $r \in \{8, 64, 256\}$; rows show evaluation splits: (a) training, (b) in-range testing, (c) out-of-range testing.

physically faithful fluctuation levels, whereas CNF-FiLM offers the most robust recovery of wall-normal turbulence intensity beyond the training horizon.

3.3. CNF with domain decomposition

Building on the WMLES-Inlet benchmark where CNFs outperformed or matched POD and ConvAE, we now assess whether a *single global latent* remains sufficient for more demanding data. Two cases increase difficulty substantially relative to WMLES: (i) DNS-resolution inlet slices of channel flow, which contain richer small-scale content, and (ii) instantaneous wall-pressure fluctuations over a flat boundary layer, whose signal is intermittent and broadband. In both settings a global CNF underresolves fine structures and degrades out of range, whereas introducing *domain decomposition*, one latent per spatial patch with a shared de-

coder, recovers sharpness and improves generalization. The effectiveness of the proposed domain-decomposed CNFs is systematically assessed through visual comparisons of snapshot reconstructions, quantitative error analysis, and evaluations of turbulence statistics.

3.3.1. Inflow turbulence of DNS channel flows

When the grid is refined to DNS resolution, the instantaneous velocity fields carry a much broader spectral bandwidth and sharper gradients, and near-wall viscous streaks coexist with outer-layer energy-containing motions, so a single global latent struggles to represent all scales. The limitation is most evident under out-of-range testing: Fig. 5 presents out-of-range reconstructions at $r=128$ (global CNFs vs. domain-decomposed CNFs with FiLM/FP).

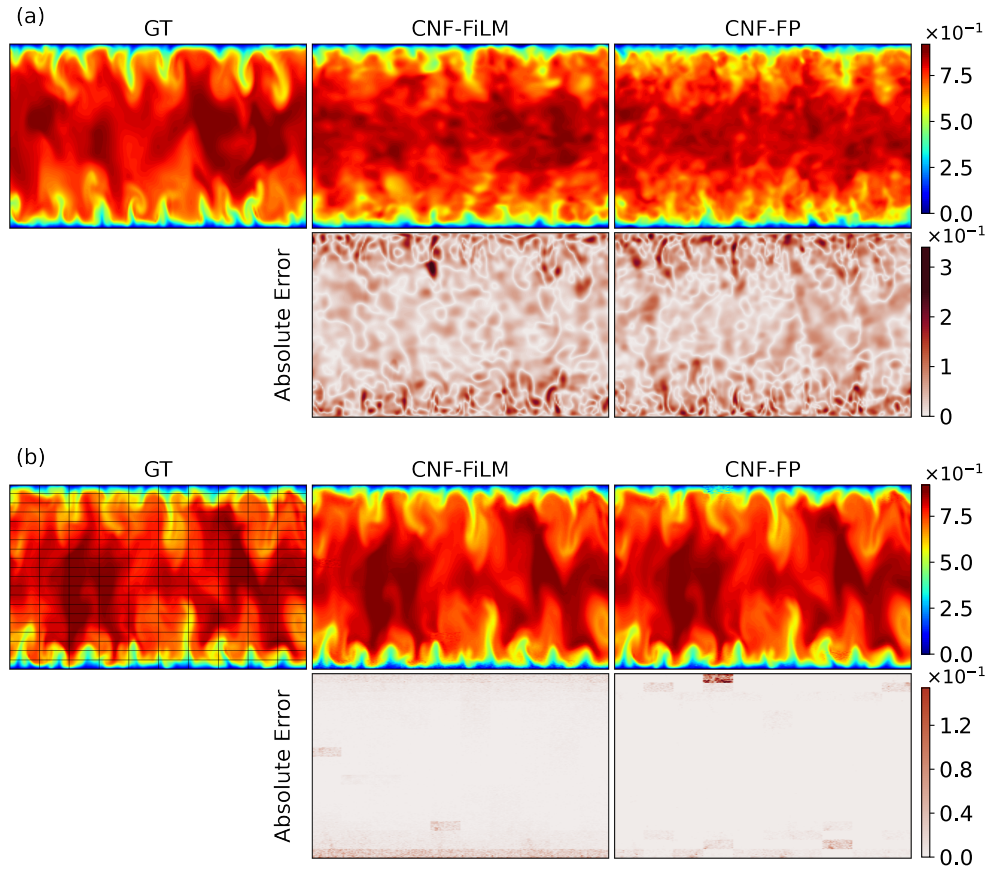


Figure 5: DNS-inlet, out-of-range testing at $r = 128$. Top: reconstructions; bottom: absolute error. (a) Global CNFs (no decomposition): blurred streaks, spurious high-wavenumber textures, larger structured errors. (b) Domain-decomposed CNFs: streak spacing and amplitude recovered; artifacts suppressed.

Without decomposition, both CNF-FiLM and CNF-FP blur core-region streaks, attenuate small-scale contrast, underresolve near-wall modulation, and introduce speckle-like high-wavenumber artifacts absent in the DNS. With decomposition, streak spacing and amplitude are largely restored and spurious fine-scale textures are suppressed.

Table 2 quantifies the reconstruction errors for both training and out-of-range testing scenarios. Across latent sizes $r \in \{32, 64, 128\}$, domain decomposition significantly reduces strict-extrapolation error for both conditioning mechanisms. Notably, the reconstruction error of decomposed CNF-FiLM model on out-of-range testing samples drops from 6.64% to 1.11% at $r=128$, with similarly large reductions at $r=32$ and 64, indicating that local conditioning primarily improves out-of-range robustness.

Latent size	Dataset	With Decomposition		Without Decomposition	
		CNF-FP	CNF-FiLM	CNF-FP	CNF-FiLM
32	Training	0.35%	0.73%	1.25%	4.03%
	Testing extrap	0.79%	0.96%	9.66%	9.33%
64	Training	0.13%	0.55%	0.88%	3.59%
	Testing extrap	3.03%	0.76%	8.67%	8.28%
128	Training	0.06%	0.48%	0.27%	2.93%
	Testing extrap	1.37%	1.11%	7.28%	6.64%

Table 2: Relative L^2 errors (%) on the DNS-Inlet dataset, comparing CNFs *with* and *without* domain decompositions.

Further statistical comparisons are provided in Figure 6. Panel (a) shows the wall-normal distribution of streamwise-velocity fluctuations $C_{u,\text{rms}}(y^+)$, where domain-decomposed CNFs (w/ DD) reproduce both the near-wall peak and the outer-layer decay with high fidelity, while global CNF models (w/o DD) systematically underpredict fluctuation intensity, consistent with the instantaneous snapshot visualizations. Figures 6b and 6c present the spanwise energy spectra $E(k_z)$ at $y^+ = 5$ (near-wall) and $y^+ = 50$ (outer layer), respectively. At both locations, models with domain decomposition align markedly better with the DNS spectrum, sustaining the inertial-range slope and delaying spectral roll-off to higher k_z . By contrast,

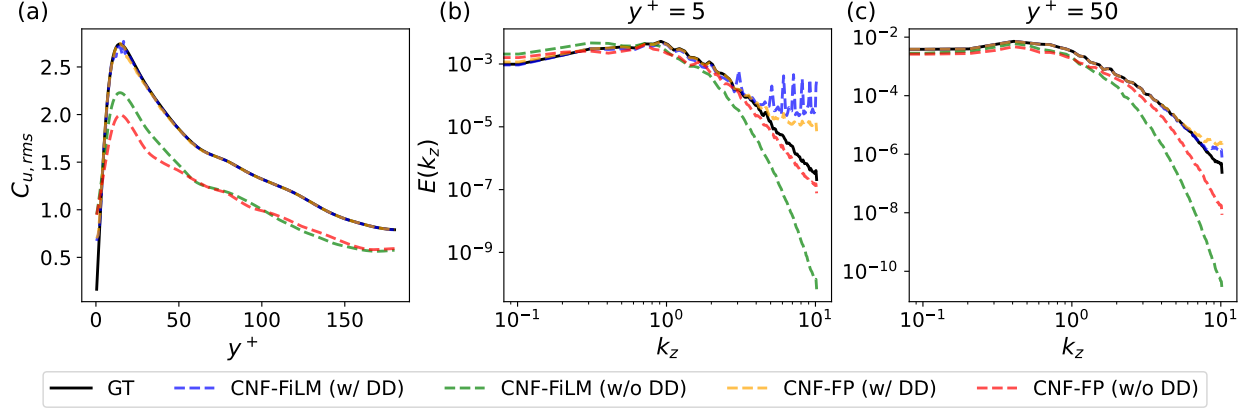


Figure 6: Turbulence statistics of reconstructed out-of-range testing samples at $r = 128$. (a) Wall-normal profile of streamwise-velocity fluctuations $C_{u,rms}(y^+)$. (b,c) Spanwise energy spectra $E(k_z)$ at near-wall ($y^+ = 5$) and outer-layer ($y^+ = 50$) locations.

global CNFs exhibit a clear high- k_z energy deficit, with CNF-FiLM (w/o DD, green) worst and CNF-FP (w/o DD, red) somewhat better yet still biased low. However, at the very near-wall region ($y^+ = 5$), the domain decomposition introduces a mild overshoot/oscillation at high k_z (most visible for CNF-FiLM w/ DD), indicative of slight over-amplification of small-scale energy at patch interfaces. Overall, domain-decomposed CNF substantially corrects the spectral bias of global CNFs while introducing only small, localized ripples that are negligible at $y^+ = 50$ and can be mitigated with overlap-and-blend or interface regularization.

3.3.2. Wall pressure fluctuation of turbulent boundary layers

Instantaneous wall-pressure fluctuations $p'(x, z, t)$ in zero-pressure-gradient turbulent boundary layers are dominated by fine, intermittent structures generated by near-wall vortical events and their footprints; the field is broadband in both space and time and more challenging than the inlet-velocity slices considered earlier. This makes the problem a stringent test of whether the models can capture locally varying dynamics.

Figure 7 compares out-of-range testing results at $r=128$ with and without domain decomposition (DD). Global CNFs (w/o DD) reproduce the broad streamwise pattern but smear and misplace intermittent high-amplitude regions; the accompanying absolute-error maps

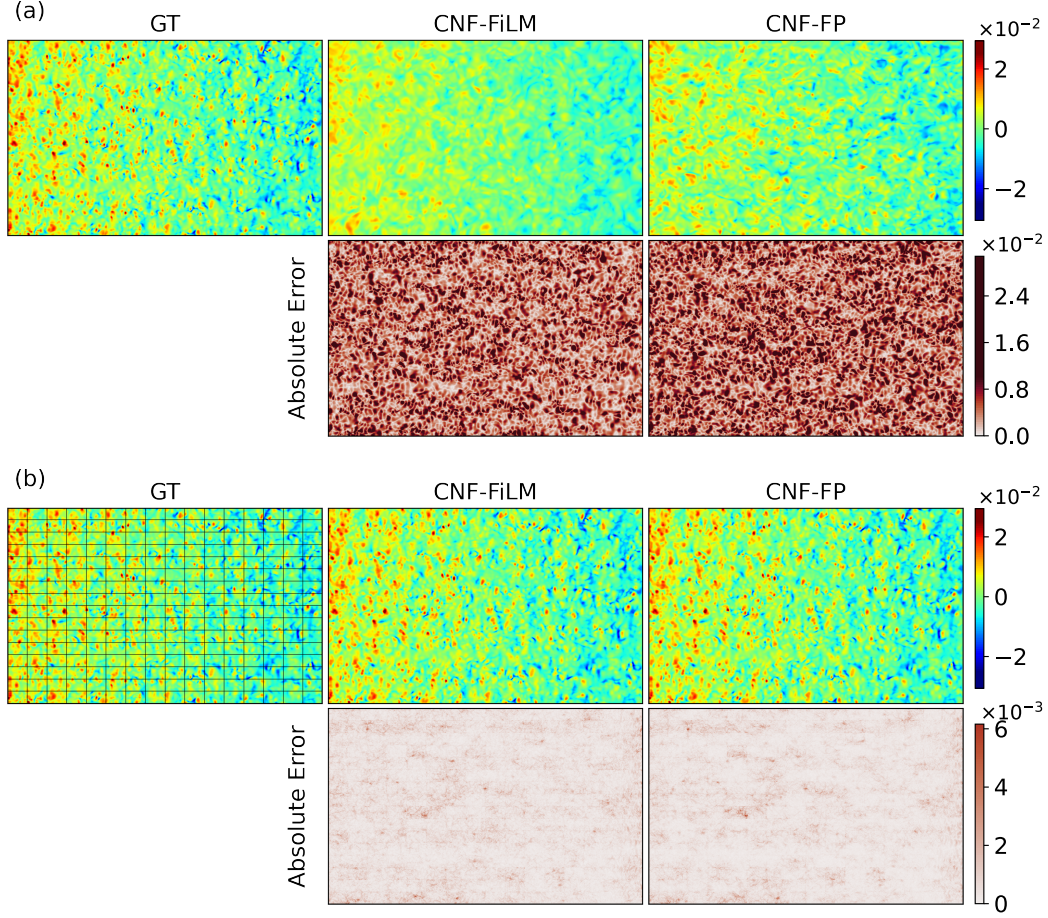


Figure 7: Wall-pressure fluctuations for *out-of-range testing* at $r = 128$. Top: reconstructions; bottom: absolute error. (a) Global CNFs (w/o DD) blur intermittent high-amplitude regions; errors are large and structured (scale $\times 10^{-2}$). (b) Domain-decomposed CNFs recover the spatial distribution and peaks; errors are diffuse and about an order of magnitude smaller (scale $\times 10^{-3}$).

exhibit large, structured residuals that persist across the field, indicating both amplitude and phase errors. With DD, both CNF-FiLM and CNF-FP recover the spatial distribution and amplitudes of p' much more faithfully, and the error maps become diffuse and an order of magnitude smaller, indicating localization suppresses the over-smoothing and misalignment inherent to a single global latent on this intermittent signal.

Table 3 summarizes relative L^2 errors across latent sizes. DD yields large improvements at every r : at $r=128$, CNF-FP improves from 24.45% $\rightarrow$ 8.92% (training) and from 109.44% $\rightarrow$

Latent size	Dataset	With Domain Decomposition		Without Domain Decomposition	
		CNF-FP	CNF-FiLM	CNF-FP	CNF-FiLM
32	Training	32.08%	40.94%	46.43%	85.53%
	Testing extrap	41.53%	39.61%	128.43%	104.01%
64	Training	14.67%	23.50%	34.03%	82.70%
	Testing extrap	17.83%	22.82%	122.78%	101.71%
128	Training	8.92%	10.19%	24.45%	78.89%
	Testing extrap	8.72%	9.49%	109.44%	99.81%

Table 3: Relative L^2 errors (%) on the wall pressure fluctuation dataset, comparing CNFs *with* and *without* domain decompositions.

8.72% (out-of-range testing); CNF-FiLM improves from 78.89% $\rightarrow$ 10.19% (training) and from 99.81% $\rightarrow$ 9.49% (out-of-range testing). Similar trends hold at $r=64$ and $r=32$. Without DD, both conditionings fail dramatically out of range (errors $> 100\%$), confirming that a single global latent is inadequate for these complex pressure fluctuation fields. With DD, CNF-FP typically fits training best, whereas FiLM and FP are comparable on extrapolation at large r .

Figure 8 (out-of-range testing, $r=128$) evaluates spatial and temporal statistics of the reconstructed wall-pressure field. Panel (a) shows the spanwise spectrum $E(k_z)$: models with DD track the DNS closely across the inertial and dissipative ranges, delaying spectral roll-off to higher k_z . In contrast, global CNFs (w/o DD) exhibit a broadband energy deficit, most severe for CNF-FiLM (green), with premature decay at moderate k_z ; CNF-FP (red) is less extreme but still biased low. Panel (b) reports the RMS of p' versus the normalized streamwise coordinate x/Θ : DD preserves both the magnitude and the weak streamwise variation of the intensity, whereas global CNFs remain uniformly underpowered along x . Panel (c) presents the frequency spectrum $E(\omega)$ at a centerline probe: DD reproduces the broadband shape and cutoff frequency, while global CNFs show an elevated low-frequency plateau and insufficient decay at high ω , which is completely off from the reference. Taken together, these diagnostics show that domain decomposition is necessary to recover the spatial and temporal

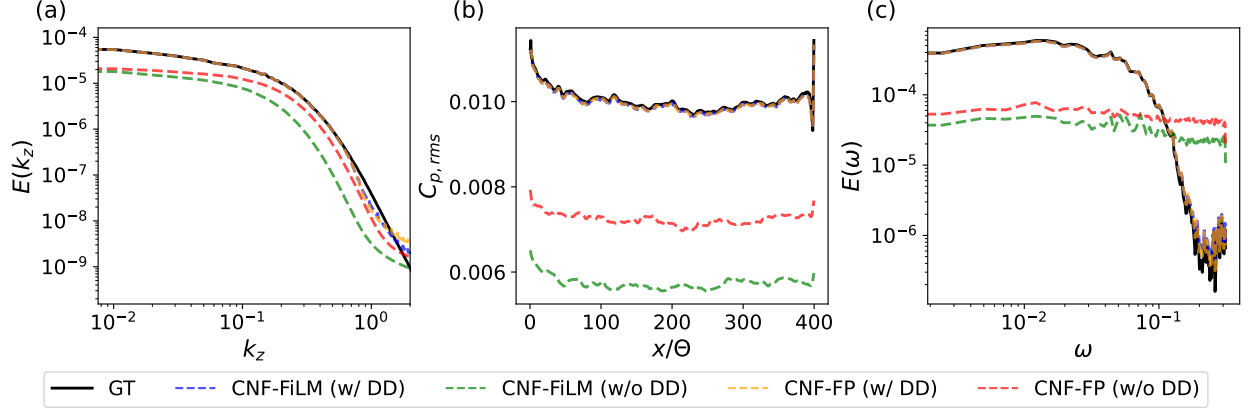


Figure 8: Statistics of wall-bounded pressure fluctuations, out-of-range testing at latent size $r = 128$. (a) Spanwise spectrum $E(k_z)$; (b) RMS of p' versus normalized streamwise location x/Θ ; (c) Frequency spectrum $E(\omega)$ at a centerline probe.

spectral content of intermittent wall-pressure fluctuations under out-of-range testing.

4. Discussion

4.1. Interpreting learned CNF modes and latent geometry

A CNF decodes a low-dimensional latent $\mathbf{z} \in \mathbb{R}^r$ into a continuous field by modulating a shared base decoder. Intuitively, a CNF mode is the spatial pattern that appears in the output field when one moves in a particular direction of the latent space. From this idea, we define a finite-amplitude mode field, which shows the nonlinear change in the reconstructed field produced by exciting one latent direction to a standardized amplitude. Let the trained decoder be

$$\phi(\mathbf{x}; \mathbf{z}) = f(\mathbf{x}; \boldsymbol{\theta}^* + \mathbf{M}^* \mathbf{z}), \quad \mathbf{z} \in \mathbb{R}^r, \quad (39)$$

where $\boldsymbol{\theta}^*$ are trained base parameters and $\mathbf{M}^* \in \mathbb{R}^{h \times r}$ maps the latent to a parameter update. The zero CNF mode $\phi^{(0)}$ is defined as the unconditioned output, which is obtained by setting the latent vector to zero:

$$\phi^{(0)} := f(\mathbf{x}; \boldsymbol{\theta}^*) \quad (40)$$

For the i^{th} latent axis, we define the mode-excited field at standardized amplitude $\alpha > 0$ as

$$\phi^{(i)}(\mathbf{x}; \alpha) := f(\mathbf{x}; \boldsymbol{\theta}^* + \mathbf{M}^* \alpha \mathbf{e}_i), \quad i = 1, \dots, r, \quad (41)$$

where $\mathbf{e}_i$ is the unit basis vector and the corresponding mode increment as

$$\Delta\phi^{(i)}(\mathbf{x}; \alpha) := \phi^{(i)}(\mathbf{x}; \alpha) - \phi^{(0)}(\mathbf{x}). \quad (42)$$

In practice we choose α to make different directions comparable, e.g. $\alpha = 1$ or $\alpha = \sigma_i$ (the empirical standard deviation of the i^{th} latent over the training set).

Figure 9 compares the empirical mean field with the CNF base field (“Mode 0”) for FiLM and FP conditioning across $r \in \{8, 64, 256\}$ on the WMLES-Inlet dataset. In all cases, Mode 0 reproduces the large-scale organization of the mean (centerline high, near-wall low) for all settings. For FiLM, Mode 0 is already close to the mean at $r = 8$ and changes mildly as r

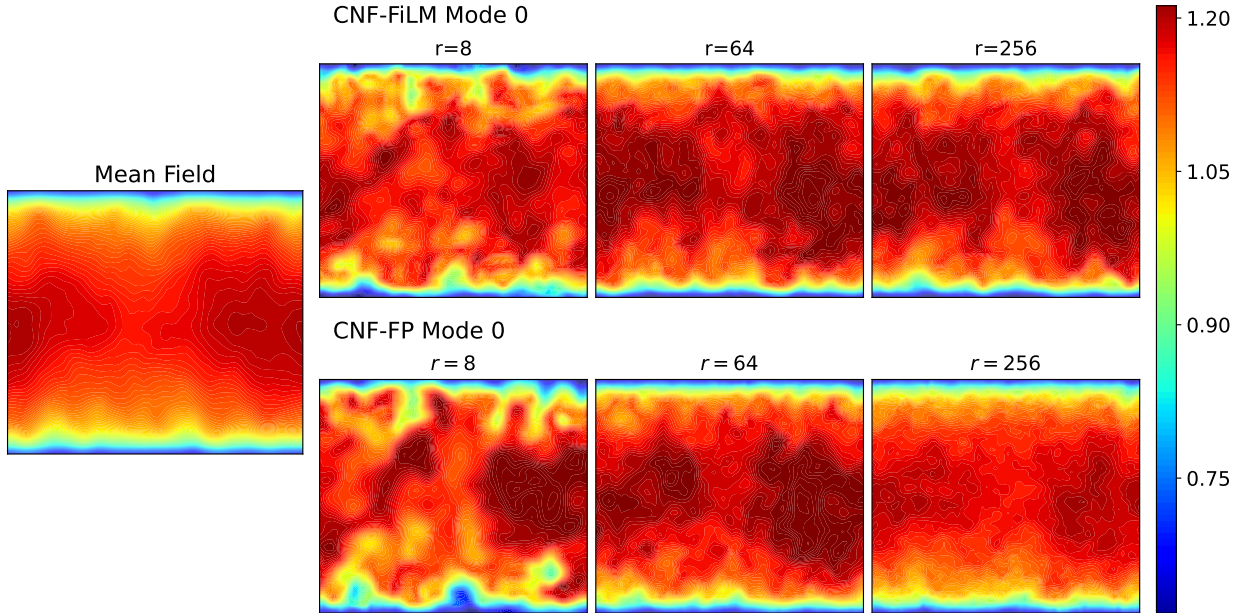


Figure 9: Mean field and CNF mode visualization at different latent size configuration (8, 64 256) for WMLES-Inlet dataset

increases; differences are localized near the walls and along gentle large-scale undulations, consistent with a base decoder that carries the low-frequency “average” structure while the latent biases modulate departures. For FP, discrepancies are more pronounced at low latent size ($r = 8$); as r grows to 64 and 256, Mode 0 progressively approaches the mean and the large-scale bias diminishes. These trends hold true regardless of hyperparameters and training recipes.

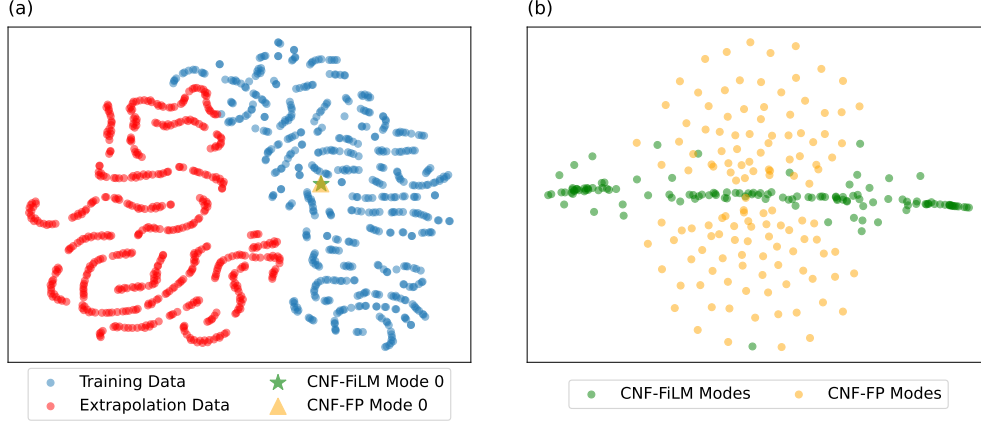


Figure 10: (a) T-SNE analysis on training WMLES-Inlet data. (b) T-SNE analysis of CNF modes

Figure 10 provides a qualitative view of the learned latent geometry using t-SNE (a non-linear embedding that preserves local neighborhoods but distorts global distances; we therefore use it only for visualization). In panel (a), the training snapshots (blue) form a coherent manifold that is clearly separated from the out-of-range testing snapshots (red), consistent with our split design in Sec. 3.1. The Mode 0 markers for FiLM (green star) and FP (orange triangle) lie near the centroid of the training cloud, in line with Fig. 9 and the interpretation of the unconditioned decoder as a mean-like representative of the training distribution. Panel (b) embeds *finite-amplitude mode* fields generated by unit excitations along single latent axes. The FiLM modes occupy a narrow, nearly one-dimensional band, whereas the FP modes spread over a substantially larger area of the embedding. This broader dispersion indicates that individual FP coordinates induce a wider variety of field perturbations than FiLM coordinates, which is consistent with FP’s stronger fitting ability and larger out-of-range generalization gap reported in Sec. 3. We emphasize that t-SNE does not support metric claims; a principled, basis-aware quantification can be obtained by analyzing the energy and rank of the standardized mode increments $\{\Delta\phi^{(i)}(\cdot; \alpha)\}$, which we view as complementary future diagnostics.

4.2. Conditioning strategy, capacity, and generalization

We probed whether the generalization gap stems from how expressive power is placed in the architecture rather than from raw trainable parameter count. First, we fixed an identical base decoder $f(\cdot; \theta)$ (depth, widths, activations) for both mechanisms and trained across latent sizes r . Under this control, FP (which applies latent-driven weight and bias updates $\Delta W^{(\ell)}(\mathbf{z}), \Delta b^{(\ell)}(\mathbf{z})$) consistently achieved lower training and in-range errors but exhibited a larger extrapolation error $\Delta_{\text{gen}} := \mathcal{E}_{\text{extrap}} - \mathcal{E}_{\text{train}}$ than FiLM (which modulates biases only). Second, to rule out parameter-count effects, we matched the *total* number of trainable parameters by increasing FiLM’s base widths. The pattern persisted across datasets and r : FP remained the best fitter yet generalized worse under strict out-of-range testing, whereas FiLM maintained smaller Δ_{gen} . The details of the result are presented in [Appendix D.2](#).

A mechanistic explanation follows from the latent-to-output sensitivity. Let $J_z(\mathbf{x}; \mathbf{z}) = \partial \phi(\mathbf{x}; \mathbf{z}) / \partial \mathbf{z}$ denote the Jacobian of the decoder with respect to the latent, its size $\|J_z\|$ measures the gain from latent perturbations to field changes. Under matched parameter budgets, FP’s multiplicative weight modulation $\Delta W^{(\ell)}(\mathbf{z})$ yields a larger gain than FiLM’s additive bias modulation $\Delta b^{(\ell)}(\mathbf{z})$, enabling sharper snapshot-specific adaptations (lower training/in-range error) but amplifying errors in out-of-range tests. Practically, capacity should therefore be allocated rather than merely increased: prefer FiLM when extrapolation is critical or combine either mechanism with domain decomposition to localize complexity; use FP when the priority is best possible in-distribution accuracy.

5. Conclusion

We presented a unified encoding–decoding framework to benchmark spatial dimensionality reduction methods for turbulent flows, placing CNFs alongside POD and CNN-AEs under identical preprocessing, metrics, and fixed train/interpolation/extrapolation splits. In contrast to most prior studies that evaluate only interpolative testing accuracy, our protocol explicitly separates *in-range* from *strict out-of-range* testing and augments pointwise errors with physics-grounded diagnostics (turbulence statistics measures).

First, among learnable decoders, CNF with full projected weight and bias modulation (CNF-FP) delivers the strongest data fitting and in-range reconstruction across latent sizes, whereas activation-only modulation (CNF-FiLM) generalizes more reliably under strict extrapolation once moderate capacity is available; linear POD is competitive in L^2 only at very small latent dimension and underrecovers fluctuation statistics. Second, when flows become more demanding, a single global latent is insufficient; a domain-decomposed CNF that localizes the mapping markedly improves extrapolation accuracy and better preserves near-wall peaks and high-wavenumber content. Third, analysis of CNF “modes” and latent-to-output sensitivity provides a mechanism for these trends: weight modulation increases latent gain, aiding fit but amplifying errors under distribution shift, whereas bias-only modulation yields a lower, more uniform gain and thus smaller extrapolation gaps. These results lead to practical guidance. For applications that prioritize robustness beyond the training horizon, prefer CNF-FiLM and allocate capacity spatially via domain decomposition; for best in-range accuracy, CNF-FP is effective, provided latent sensitivity is controlled.

Limitations and opportunities remain. Our study focuses on spatial reduction with auto-decoding; future work should assess amortized encoders for fast inference under partial observations, extend domain decomposition with overlap and adaptive tiling, and couple spatial CNFs with temporal models for fully spatiotemporal reduction. Incorporating uncertainty quantification for latents and sensitivity-aware training objectives may further stabilize extrapolation. We expect the evaluation protocol and analyses here to serve as a physics-aware basis for choosing conditioning, capacity, and localization when deploying CNFs for turbulence compression, reconstruction, and as building blocks for operator learning and generative flow models.

Acknowledgements

The authors would like to acknowledge the funds from Office of Naval Research under award numbers N00014-23-1-2071 and National Science Foundation under award numbers OAC-2047127.

Appendix A. Dataset generation

We construct training and testing datasets from two benchmark flow configurations: a 3D turbulent channel flow and a 3D turbulent flat boundary layer. Both cases follow the unsteady incompressible Navier–Stokes equations:

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

where $\mathbf{u}(\mathbf{x}, t)$ denotes the velocity vector, $p(\mathbf{x}, t)$ the pressure, ν the viscosity, and $\mathbf{f}(\mathbf{x}, t)$ the forcing term. We employ a wall-modeled large-eddy simulation (WMLES) framework and direct numerical simulation (DNS) within our in-house Navier–Stokes solver to generate datasets [46].

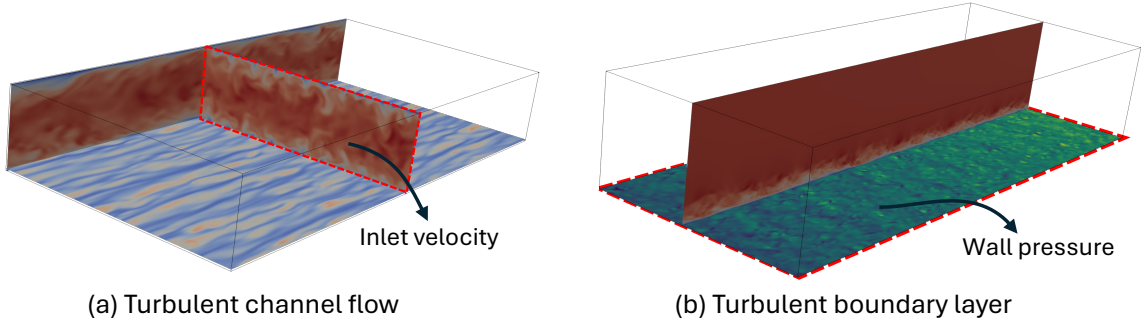


Figure A.11: The schematic of dataset extraction from 3D simulations. We crop the data from the 2D plane marked in red dashed line.

Appendix A.1. Turbulent channel flow (WMLES)

The friction Reynolds number is $Re_\tau = 6000$. The computational domain measures $2\pi \times 2 \times \pi$ and is discretized using a structured grid of $64 \times 64 \times 64$. We advance the solution with a uniform time step $\Delta t = 0.05$ s. For training and evaluation, we consider only the velocity field in the streamwise direction (x) and extract two-dimensional inlet plane snapshots with a resolution of 64×64 . We collect 3000 total snapshots from $t = 0$ s to $t = 150$ s. From these, 1000 randomly chosen snapshots in $t \in [0, 100)$ serve as training data, with the remaining

snapshots in this interval used for interpolation testing. All snapshots in $t \in [100, 150)$ are reserved for extrapolation testing.

Appendix A.2. Turbulent channel flow (DNS)

The simulations are performed at a friction Reynolds number of $Re_\tau = 180$. The computational domain has dimensions $4\pi \times 2 \times 2\pi$ and is discretized on a structured grid of $320 \times 400 \times 200$ points. Time integration is carried out with a uniform step size of $\Delta t = 5 \times 10^{-4}$ s. For training and evaluation, we focus on the streamwise velocity field and extract two-dimensional inlet-plane snapshots with a resolution of 400×200 . A total of 21,900 snapshots are collected over the interval $t \in [0, 1095]$, sampled every 100 numerical steps. From these, 1,000 randomly selected snapshots in $t \in [0, 100)$ are used for training, while the remaining snapshots in this interval are employed for interpolation testing. All snapshots in $[100, 150)$ are reserved for extrapolation testing.

Appendix A.3. Turbulent flat boundary layer (DNS)

The Reynolds number based on the free-stream velocity and inlet momentum thickness is $Re_\theta = 300$. The computational domain, normalized by the inlet momentum thickness, is $L_x \times L_y \times L_z = 400 \times 80 \times 160$. The domain is uniformly discretized in the streamwise (x) and spanwise (z) directions, while a stretched grid is employed in the wall-normal (y) direction, with a total resolution of $N_x \times N_y \times N_z = 512 \times 320 \times 512$. Wall pressure fluctuation data are sampled at intervals of $\Delta T = 1.6$ from $x - z$ plane, which corresponds to ten times the numerical time step. The total duration of the collected wall pressure data spans 12 flow-through times, yielding 3000 snapshots. Of these, the first 2000 snapshots are used as the training and interpolation dataset, while the last 1000 snapshots are reserved for extrapolation testing.

Appendix B. Model implementation details

Appendix B.1. CNF & decomposed CNF architecture and hyper-parameters

As outlined in the methodology, our CNF model’s base network maps input coordinates to field values. We implement this network using a Multilayer Perceptron (MLP) based on the

SIREN architecture [38], which employs ω_0 -scaled sine activation functions, $\sigma(x) = \sin(\omega_0 x)$, for all hidden layers. This choice is motivated by SIREN’s effectiveness in representing continuous signals and their derivatives, crucial for accurately capturing field details and mitigating the spectral bias towards low frequencies often seen in standard MLPs [37]. Our specific architecture uses 8 hidden layers, each with 64 neurons ($n_h = 64$) and frequency parameter $\omega_0 = 30$ for all testing cases. Furthermore, SIREN requires a specific weight initialization scheme. For the first layer, weights $W^{(0)}$ are initialized element-wise from a uniform distribution $\mathcal{U}(-1/n_i, 1/n_i)$. For all subsequent hidden layers, weights $\{W^{(l)}\}_{l=1}^8$ are initialized element-wise from $\mathcal{U}\left(-\frac{\sqrt{6/n_h}}{\omega_0}, \frac{\sqrt{6/n_h}}{\omega_0}\right)$.

For the decomposed framework, it was observed that optimal performance necessitates the use of distinct patch sizes for different datasets. Specifically, the 512×512 full domain of the Wall-Pressure fluctuation dataset is decomposed into 32×32 non-overlapping patches, while the 400×200 full domain of the Inlet-DNS dataset is decomposed into 20×20 non-overlapping patches.

Appendix B.2. Baseline CNN-AE architecture and hyper-parameter selection

For baseline comparison, we implemented a standard convolutional autoencoder (CNN-AE) closely following Pan et al. [40]. Its encoder consists of three convolutional stages—each a 3×3 kernel, stride 2, padding 1 convolution followed by batch normalization and ReLU—and a fully connected layer projecting to the latent space. The decoder mirrors this design with three 3×3 transposed convolutions (stride 2, padding 1, padding 1), each again paired with batch normalization and ReLU, to recover the original spatial resolution. To ensure fair comparison in representational capacity, we tuned the number of channels at every stage so that, for each latent size configuration, the total parameter count of the CNN-AE matches that of our CNF model. Detailed per-layer output shapes and trainable parameter counts for all latent-size configurations appear in Table B.4.

Layer Type	Latent size					
	8	16	32	64	128	256
Conv2D	(B, 31, 32, 32)	(B, 37, 32, 32)	(B, 42, 32, 32)	(B, 52, 32, 32)	(B, 54, 32, 32)	(B, 60, 32, 32)
BatchNorm2d	(B, 31, 32, 32)	(B, 37, 32, 32)	(B, 42, 32, 32)	(B, 52, 32, 32)	(B, 54, 32, 32)	(B, 60, 32, 32)
ReLU	(B, 31, 32, 32)	(B, 37, 32, 32)	(B, 42, 32, 32)	(B, 52, 32, 32)	(B, 54, 32, 32)	(B, 60, 32, 32)
Conv2D	(B, 62, 16, 16)	(B, 74, 16, 16)	(B, 84, 16, 16)	(B,104, 16, 16)	(B,108, 16, 16)	(B,120, 16, 16)
BatchNorm2d	(B, 62, 16, 16)	(B, 74, 16, 16)	(B, 84, 16, 16)	(B,104, 16, 16)	(B,108, 16, 16)	(B,120, 16, 16)
ReLU	(B, 62, 16, 16)	(B, 74, 16, 16)	(B, 84, 16, 16)	(B,104, 16, 16)	(B,108, 16, 16)	(B,120, 16, 16)
Conv2D	(B,124, 8, 8)	(B,148, 8, 8)	(B,168, 8, 8)	(B,208, 8, 8)	(B,216, 8, 8)	(B,240, 8, 8)
BatchNorm2d	(B,124, 8, 8)	(B,148, 8, 8)	(B,168, 8, 8)	(B,208, 8, 8)	(B,216, 8, 8)	(B,240, 8, 8)
ReLU	(B,124, 8, 8)	(B,148, 8, 8)	(B,168, 8, 8)	(B,208, 8, 8)	(B,216, 8, 8)	(B,240, 8, 8)
Flatten	(B, 7936)	(B, 9472)	(B,10752)	(B,13312)	(B,13824)	(B,15360)
Linear	(B, 8)	(B, 16)	(B, 32)	(B, 64)	(B, 128)	(B, 256)
Linear	(B, 7936)	(B, 9472)	(B,10752)	(B,13312)	(B,13824)	(B,15360)
Unflatten	(B,124, 8, 8)	(B,148, 8, 8)	(B,168, 8, 8)	(B,208, 8, 8)	(B,216, 8, 8)	(B,240, 8, 8)
ConvTranspose2d	(B, 62, 16, 16)	(B, 74, 16, 16)	(B, 84, 16, 16)	(B,104, 16, 16)	(B,108, 16, 16)	(B,120, 16, 16)
BatchNorm2d	(B, 62, 16, 16)	(B, 74, 16, 16)	(B, 84, 16, 16)	(B,104, 16, 16)	(B,108, 16, 16)	(B,120, 16, 16)
ReLU	(B, 62, 16, 16)	(B, 74, 16, 16)	(B, 84, 16, 16)	(B,104, 16, 16)	(B,108, 16, 16)	(B,120, 16, 16)
ConvTranspose2d	(B, 31, 32, 32)	(B, 37, 32, 32)	(B, 42, 32, 32)	(B, 52, 32, 32)	(B, 54, 32, 32)	(B, 60, 32, 32)
BatchNorm2d	(B, 31, 32, 32)	(B, 37, 32, 32)	(B, 42, 32, 32)	(B, 52, 32, 32)	(B, 54, 32, 32)	(B, 60, 32, 32)
ReLU	(B, 31, 32, 32)	(B, 37, 32, 32)	(B, 42, 32, 32)	(B, 52, 32, 32)	(B, 54, 32, 32)	(B, 60, 32, 32)
ConvTranspose2d	(B,1,64,64)	(B,1,64,64)	(B,1,64,64)	(B,1,64,64)	(B,1,64,64)	(B,1,64,64)
total params	309389	560789	1018449	2206529	4080369	8530817

Table B.4: CNN-AE output shapes by layer for different latent size configuration

Appendix C. Evaluation Metrics

Relative L^2 error. To quantify the model's performance, we adopt the mean relative L^2 error. Let $\{\phi^i\}_{i=1}^N$ denote the ensemble of the ground truth turbulence data used in this work, and $\{\hat{\phi}^i\}_{i=1}^N$ denote the corresponding model predictions. Relative L^2 error ϵ is defined as:

$$\epsilon = \frac{1}{N} \sum_{i=1}^N \frac{\|\phi^i - \hat{\phi}^i\|_2}{\|\phi^i\|_2}, \quad (\text{C.1})$$

where $\|\cdot\|_2$ denotes the standard Euclidean norm.

Turbulence statistics. To compute the turbulence statistics, we first decompose flow quantities $q(x, y, z, t)$ into a mean $\langle q \rangle(y)$ and fluctuating q' components, i.e. $q' = q - \langle q \rangle$. Here, $\langle \cdot \rangle$ denotes the operator that averages over time and the homogeneous directions. All the turbulence statistics are normalized by the inner scale, taking the friction velocity u_τ as the reference. Spatial dimensions are normalized by viscous length, $\delta_\nu = \nu/u_\tau$. For example, the dimensionless wall-normal coordinate is defined as $y^+ = y/\delta_\nu$.

The normalized root-mean-square of velocity fluctuation $C_{u,rms}$ is computed as:

$$C_{u,rms} = \frac{\sqrt{\langle u'^2 \rangle}}{u_\tau}. \quad (\text{C.2})$$

For wall pressure measured at $y = 0$, with fluctuation p' , the normalized root-mean-square is:

$$C_{p,rms} = \frac{\sqrt{\langle p'^2 \rangle}}{\rho u_\tau^2}. \quad (\text{C.3})$$

We report 1D spectrum consistent with an energy-conserving FFT implementation. Define two-sided power spectral density (PSD) $\Psi(\omega, k_x, k_z)$ of q' . The 1D spanwise wavenumber spectrum is obtained by integrating out frequency and streamwise wavenumber and then adopting a one-sided convention in $k_z > 0$:

$$E(k_z) = \left\langle \int_0^\infty \int_0^\infty \Psi(\omega, k_x, k_z) dk_x d\omega \right\rangle. \quad (\text{C.4})$$

The 1D temporal spectrum at a fixed streamwise location x_{loc} is the temporal PSD of $q'(t, x_{loc}, z)$ averaged over z , denoted $S(\omega; x_{loc})$, with

$$E(\omega) = \left\langle \int_0^\infty S(\omega; x_{loc}) d\omega \right\rangle. \quad (\text{C.5})$$

Appendix D. Additional results

This section presents supplementary results supporting the main paper's findings. We provide a visualization of the flow field reconstructions referenced in Sec. 3.2 and quantitative results of the study discussed in Sec. 4.2.

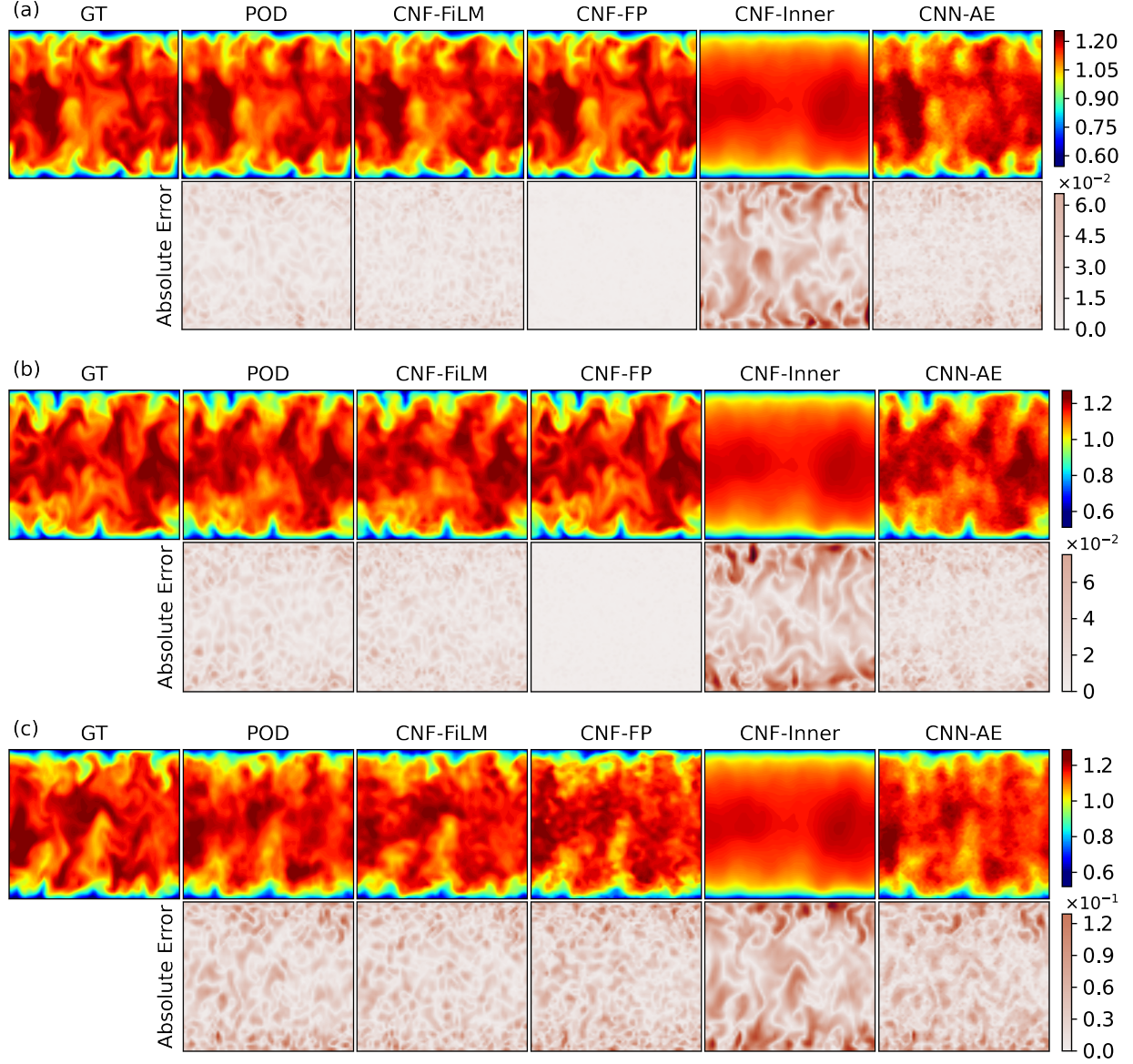


Figure D.12: WMLES-Inlet snapshots at latent size $r = 128$. Each panel shows the reconstructed stream-wise-velocity field (top) and the corresponding absolute error (bottom). Rows follow our evaluation splits: (a) training, (b) in-range testing, (c) out-of-range testing.

Appendix D.1. Supplementary visualization for benchmark study

Figure D.12 offers a qualitative assessment of the model's reconstruction fidelity for a latent size of $r = 128$. The reconstructions for CNF-FP for the training and in-range test cases are visually indistinguishable from the ground truth, with absolute errors that are small in

magnitude. This indicates the CNF-FP’s strong capacity for high-fidelity representation of data within the training distribution. For the out-of-range test case, while the primary flow structures are accurately captured, a noticeable increase in reconstruction error is observed. These errors appear to be concentrated in regions characterized by high spatial frequency content and strong velocity gradients, which are expected challenges when generalizing to unseen flow conditions.

Appendix D.2. Supplementary result for generalization discussion

We construct a new baseline model, denoted CNF-FiLM*, specifically designed to have the same total number of trainable parameters as the FP model. To achieve this parameter equivalence, we increase its hidden layer width while keeping its depth fixed. Table D.5 lists the architectural choices and resulting parameter counts; reconstruction errors for training, interpolation, and extrapolation splits are given in Table D.6.

The CNF-FP model demonstrates superior performance on both training and interpolation splits, achieving the lowest error. This suggests its architectural design is more efficient for fitting in-distribution data. For out-of-distribution data, the most compact model CNF-FiLM achieves the lowest error. The proposed CNF-FP is the second-best performer, whereas the high-capacity CNF-FiLM* exhibits the worst generalization. This outcome indicates that over-parameterizing the FiLM architecture is detrimental to its extrapolation capabilities.

Latent size	Configuration	CNF-FiLM*	CNF-FP	CNF-FiLM
32	Network width	350	64	64
	Network depth	8	8	8
	Total trainable parameters	1.09×10^6	1.10×10^6	5.20×10^4
64	Network width	480	64	64
	Network depth	8	8	8
	Total trainable parameters	2.13×10^6	2.18×10^6	7.04×10^4
128	Network width	650	64	64
	Network depth	8	8	8
	Total trainable parameters	4.14×10^6	4.32×10^6	1.07×10^5

Table D.5: Architectural hyperparameters and total trainable parameters for three Conditional Neural Field variants at latent sizes. “Width” is the number of hidden units per layer; “depth” is the number of hidden layers. CNF-FiLM* denotes the FiLM variant widened to match CNF-FP’s parameter count.

Latent size	Split	CNF-FiLM*	CNF-FP	CNF-FiLM
32	Training	0.84%	0.46%	2.10%
	Interpolation	1.67%	0.73%	2.37%
	Extrapolation	6.57%	5.21%	5.05%
64	Training	0.55%	0.39%	1.73%
	Interpolation	1.59%	0.54%	2.02%
	Extrapolation	5.74%	4.37%	4.12%
128	Training	0.29%	0.21%	5.05%
	Interpolation	1.06%	0.36%	1.64%
	Extrapolation	4.79%	3.61%	3.10%

Table D.6: Reconstruction error (%) by latent size and data split. CNF-FiLM* is the widened FiLM model with parameter count comparable to CNF-FP at each latent size.

References

- [1] S. B. Pope, Turbulent flows, *Measurement Science and Technology* 12 (11) (2001) 2020–2021.
- [2] K. Taira, S. L. Brunton, S. T. Dawson, C. W. Rowley, T. Colonius, B. J. McKeon, O. T. Schmidt, S. Gordeyev, V. Theofilis, L. S. Ukeiley, Modal analysis of fluid flows: An overview, *Aiaa Journal* 55 (12) (2017) 4013–4041.
- [3] K. Taira, M. S. Hemati, S. L. Brunton, Y. Sun, K. Duraisamy, S. Bagheri, S. T. Dawson, C.-A. Yeh, Modal analysis of fluid flows: Applications and outlook, *AIAA journal* 58 (3) (2020) 998–1022.
- [4] Y. Kim, Y. Choi, D. Widemann, T. Zohdi, A fast and accurate physics-informed neural network reduced order model with shallow masked autoencoder, *Journal of Computational Physics* 451 (2022) 110841.
- [5] Z. C. Khoo, C. H. Chan, Y. Hwang, A sparse optimal closure for a reduced-order model of wall-bounded turbulence, *Journal of Fluid Mechanics* 939 (2022) A11.
- [6] A. Solera-Rico, C. Sanmiguel Vila, M. Gómez-López, Y. Wang, A. Almashjary, S. T. Dawson, R. Vinuesa, β -variational autoencoders and transformers for reduced-order modelling of fluid flows, *Nature Communications* 15 (1) (2024) 1361.
- [7] S. Fresca, A. Manzoni, Pod-dl-rom: Enhancing deep learning-based reduced order models for nonlinear parametrized pdes by proper orthogonal decomposition, *Computer Methods in Applied Mechanics and Engineering* 388 (2022) 114181.
- [8] P. Ren, C. Rao, Y. Liu, J.-X. Wang, H. Sun, Phycrnet: Physics-informed convolutional-recurrent network for solving spatiotemporal pdes, *Computer Methods in Applied Mechanics and Engineering* 389 (2022) 114399.

- [9] L. Sun, H. Gao, S. Pan, J.-X. Wang, Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data, *Computer Methods in Applied Mechanics and Engineering* 361 (2020) 112732.
- [10] L. Sun, X. Han, H. Gao, J.-X. Wang, L. Liu, Unifying predictions of deterministic and stochastic physics in mesh-reduced space with sequential flow generative model, *Advances in Neural Information Processing Systems* 36 (2023) 60636–60660.
- [11] P. Ren, C. Rao, Y. Liu, Z. Ma, Q. Wang, J.-X. Wang, H. Sun, Physr: Physics-informed deep super-resolution for spatiotemporal data, *Journal of Computational Physics* 492 (2023) 112438.
- [12] N. B. Erichson, L. Mathelin, Z. Yao, S. L. Brunton, M. W. Mahoney, J. N. Kutz, [Shallow neural networks for fluid flow reconstruction with limited sensors](#), *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 476 (2238) (2020) 20200097. doi:[10.1098/rspa.2020.0097](https://doi.org/10.1098/rspa.2020.0097).
URL <http://dx.doi.org/10.1098/rspa.2020.0097>
- [13] K. Fukami, K. Fukagata, K. Taira, Machine-learning-based spatio-temporal super resolution reconstruction of turbulent flows, *Journal of Fluid Mechanics* 909 (2021) A9.
- [14] Z. Zhang, X. Gao, Q. Chen, Y. Yuan, A novel thermal turbulence reconstruction method using proper orthogonal decomposition and compressed sensing coupled based on improved particle swarm optimization for sensor arrangement, *Physics of Fluids* 36 (5) (2024).
- [15] J. L. Lumley, The structure of inhomogeneous turbulent flows, *Atmospheric turbulence and radio wave propagation* (1967) 166–178.
- [16] C. Picard, J. Delville, Pressure velocity coupling in a subsonic round jet, *International Journal of Heat and Fluid Flow* 21 (3) (2000) 359–364.
- [17] P. J. Schmid, Dynamic mode decomposition of numerical and experimental data, *Journal of fluid mechanics* 656 (2010) 5–28.

- [18] H. Csala, S. Dawson, A. Arzani, Comparing different nonlinear dimensionality reduction techniques for data-driven unsteady fluid flow modeling, *Physics of Fluids* 34 (11) (2022).
- [19] M. A. Mendez, Linear and nonlinear dimensionality reduction from fluid mechanics to machine learning, *Measurement Science and Technology* 34 (4) (2023) 042001.
- [20] B. Schölkopf, A. Smola, K.-R. Müller, Nonlinear component analysis as a kernel eigenvalue problem, *Neural computation* 10 (5) (1998) 1299–1319.
- [21] M. Balasubramanian, E. L. Schwartz, The isomap algorithm and topological stability, *Science* 295 (5552) (2002) 7–7.
- [22] S. T. Roweis, L. K. Saul, Nonlinear dimensionality reduction by locally linear embedding, *science* 290 (5500) (2000) 2323–2326.
- [23] L. Van der Maaten, G. Hinton, Visualizing data using t-sne., *Journal of machine learning research* 9 (11) (2008).
- [24] J. A. Lee, M. Verleysen, et al., *Nonlinear dimensionality reduction*, Vol. 1, Springer, 2007.
- [25] J.-Y. Kwok, I.-H. Tsang, The pre-image problem in kernel methods, *IEEE transactions on neural networks* 15 (6) (2004) 1517–1525.
- [26] T. Murata, K. Fukami, K. Fukagata, Nonlinear mode decomposition with convolutional neural networks for fluid dynamics, *Journal of Fluid Mechanics* 882 (2020) A13.
- [27] K. Fukami, T. Murata, K. Zhang, K. Fukagata, [Sparse identification of nonlinear dynamics with low-dimensionalized flow representations](https://doi.org/10.1017/jfm.2021.697), *Journal of Fluid Mechanics* 926 (Sep. 2021). [doi:10.1017/jfm.2021.697](https://doi.org/10.1017/jfm.2021.697).
URL <http://dx.doi.org/10.1017/jfm.2021.697>
- [28] A. Racca, N. A. K. Doan, L. Magri, Predicting turbulent dynamics with the convolutional autoencoder echo state network, *Journal of Fluid Mechanics* 975 (2023) A2. [doi:10.1017/jfm.2023.716](https://doi.org/10.1017/jfm.2023.716).

- [29] Y. Xie, T. Takikawa, S. Saito, O. Litany, S. Yan, N. Khan, F. Tombari, J. Tompkin, V. Sitzmann, S. Sridhar, Neural fields in visual computing and beyond, in: Computer Graphics Forum, Vol. 41, Wiley Online Library, 2022, pp. 641–676.
- [30] K. Gao, Y. Gao, H. He, D. Lu, L. Xu, J. Li, Nerf: Neural radiance field in 3d vision, a comprehensive review, arXiv preprint arXiv:2210.00379 (2022).
- [31] P. Bojanowski, A. Joulin, D. Lopez-Pas, A. Szlam, Optimizing the latent space of generative networks, in: International Conference on Machine Learning, PMLR, 2018, pp. 600–609.
- [32] J. J. Park, P. Florence, J. Straub, R. Newcombe, S. Lovegrove, Deepsdf: Learning continuous signed distance functions for shape representation, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 165–174.
- [33] P. Y. Chen, J. Xiang, D. H. Cho, Y. Chang, G. A. Pershing, H. T. Maia, M. M. Chiaramonte, K. T. Carlberg, E. Grinspun, [CROM: Continuous reduced-order modeling of PDEs using implicit neural representations](#), in: The Eleventh International Conference on Learning Representations, 2023.
URL <https://openreview.net/forum?id=FUORz1tG80g>
- [34] H. Chen, R. Wu, E. Grinspun, C. Zheng, P. Y. Chen, [Implicit neural spatial representations for time-dependent pdes](#) (2023). [arXiv:2210.00124](#).
URL <https://arxiv.org/abs/2210.00124>
- [35] L. Serrano, L. Le Boudec, A. Kassai Koupai, T. X. Wang, Y. Yin, J.-N. Vittaut, P. Gallinari, Operator learning with neural fields: Tackling pdes on general geometries, Advances in Neural Information Processing Systems 36 (2023) 70581–70611.
- [36] Y. Yin, M. Kirchmeyer, J.-Y. Franceschi, A. Rakotomamonjy, P. Gallinari, Continuous pde dynamics forecasting with implicit neural representations, arXiv preprint arXiv:2209.14855 (2022).

- [37] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, A. Courville, On the spectral bias of neural networks, in: International conference on machine learning, PMLR, 2019, pp. 5301–5310.
- [38] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, G. Wetzstein, Implicit neural representations with periodic activation functions, *Advances in neural information processing systems* 33 (2020) 7462–7473.
- [39] M. Tancik, P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. Barron, R. Ng, Fourier features let networks learn high frequency functions in low dimensional domains, *Advances in neural information processing systems* 33 (2020) 7537–7547.
- [40] S. Pan, S. L. Brunton, J. N. Kutz, Neural implicit flow: a mesh-agnostic dimensionality reduction paradigm of spatio-temporal data, *Journal of Machine Learning Research* 24 (41) (2023) 1–60.
- [41] P. Du, M. H. Parikh, X. Fan, X.-Y. Liu, J.-X. Wang, Conditional neural field latent diffusion model for generating spatiotemporal turbulence, *Nature Communications* (2024).
- [42] X.-Y. Liu, M. H. Parikh, X. Fan, P. Du, Q. Wang, Y.-F. Chen, J.-X. Wang, Confield-inlet: Synthetic turbulence inflow using generative latent diffusion models with neural fields, *Physical Review Fluids* 10 (5) (2025) 054901.
- [43] G. Zhang, Z. Wang, H. Huang, H. Li, T. Sun, Comparison and evaluation of dimensionality reduction techniques for the numerical simulations of unsteady cavitation, *Physics of Fluids* 35 (7) (2023).
- [44] C. Eckart, G. Young, The approximation of one matrix by another of lower rank, *Psychometrika* 1 (3) (1936) 211–218.
- [45] M. T. Chu, R. E. Funderlic, R. J. Plemmons, Structured low rank approximation, *Linear algebra and its applications* 366 (2003) 157–172.

- [46] X. Fan, X. Liu, M. Wang, J.-X. Wang, Diff-flowfsi: A gpu-optimized differentiable cfd platform for high-fidelity turbulence and fsi simulations, *Computer Methods in Applied Mechanics and Engineering* (2026).