Journal of Machine Learning Research 24 (2023) 1-60 Submitted 4/22; Revised 1/23; Published 1/23

Neural Implicit Flow: a mesh-agnostic dimensionality reduction paradigm of spatio-temporal data

Shaowu Pan SHAWNPAN@UW.EDU
Department of Applied Mathematics
University of Washington
Seattle, WA 98195-4322, USA

Steven L. Brunton SBRUNTON@UW.EDU
Department of Mechanical Engineering
University of Washington
Seattle, WA 98195-4322, USA

J. Nathan Kutz KUTZ@UW.EDU
Department of Applied Mathematics
University of Washington
Seattle, WA 98195-4322, USA

Editor: Animashree Anandkumar

Abstract

High-dimensional spatio-temporal dynamics can often be encoded in a low-dimensional subspace. Engineering applications for modeling, characterization, design, and control of such large-scale systems often rely on dimensionality reduction to make solutions computationally tractable in real time. Common existing paradigms for dimensionality reduction include linear methods, such as the *singular value decomposition* (SVD), and nonlinear methods, such as variants of *convolutional autoencoders* (CAE). However, these encoding techniques lack the ability to efficiently represent the complexity associated with spatio-temporal data, which often requires variable geometry, non-uniform grid resolution, adaptive meshing, and/or parametric dependencies. To resolve these practical engineering challenges, we propose a general framework called *Neural Implicit Flow* (NIF) that enables a mesh-agnostic, low-rank representation of large-scale, parametric, spatial-temporal data. NIF consists of two modified *multilayer perceptrons* (MLPs): (i) *ShapeNet*, which isolates and represents the spatial complexity, and (ii) *ParameterNet*, which accounts for any other input complexity, including parametric dependencies, time, and sensor measurements. We demonstrate the utility of NIF for parametric surrogate modeling, enabling the interpretable representation and compression of complex spatio-temporal dynamics, efficient many-spatial-query tasks, and improved generalization performance for sparse reconstruction.

Keywords: Deep learning, dimensionality reduction, partial differential equations

1. Introduction

Machine learning and artificial intelligence algorithms have broadly transformed science and engineering, including the application areas of computer vision (Krizhevsky et al., 2012), natural language processing (Sutskever et al., 2014), molecular dynamics (Zhang et al., 2018; Mardt et al., 2018), and dynamical systems (Brunton and Kutz, 2019). The subfield

of *scientific machine learning*, which often focuses on modeling, characterization, design, and control of large-scale, physics-based models, has also experienced significant growth. Despite the achievements in scientific machine learning (Duraisamy et al., 2019; Karniadakis et al., 2021; Kutz, 2017; Brunton et al., 2020), there remain significant challenges in the representation of high-dimensional spatio-temporal dynamics, which are often modeled by nonlinear partial differential equations (PDEs). Data-driven modeling of PDE systems often relies on a more advantageous representation of the physics. In general, *manifold-based* methods are a dominant paradigm (Hesthaven et al., 2016; Carlberg et al., 2011; Peherstorfer and Willcox, 2016; Zahr and Farhat, 2015; Benner et al., 2015). However, there are recent innovations in developing *mesh-based* methods (Long et al., 2018; Zhu and Zabaras, 2018; Geneva and Zabaras, 2020; Bar-Sinai et al., 2019; Li et al., 2020a; Pfaff et al., 2020) and *mesh-agnostic* methods (Raissi et al., 2020; Lu et al., 2021a,b; Sun et al., 2020). Despite the diversity of algorithmic innovations, each has various challenges in efficiently or accurately representing complex spatio-temporal dynamics. Indeed, practical engineering applications require handling variable geometry, non-uniform grid resolutions, adaptive meshes, and/or parametric dependencies. We advocate a general mathematical framework called *Neural Implicit Flow* (NIF) that enables a mesh-agnostic, low-rank representation of large-scale, parametric, spatial-temporal data. NIF leverages a hypernetwork structure that allows one to isolate the spatial complexity, thus accounting for all other complexity in a second network where parametric dependencies, time, and sensor measurements are encoded and modulating the spatial layer. We show that NIF is highly advantageous for representing spatio-temporal dynamics in comparison with current methods.

Spatio-temporal data is ubiquitous. As such, a diversity of methods have been developed to characterize the underlying physics. In manifold-based modeling, which is the most dominant paradigm for reduced-order modeling, one first extracts a low-dimensional manifold from the solution of a PDE, typically using the singular value decomposition (Benner et al., 2015; Noack et al., 2003; Rowley et al., 2004) or a convolutional autoencoder (CAE) (Brunton and Kutz, 2019; Holmes et al., 2012; Mohan et al., 2019; Murata et al., 2020; Xu and Duraisamy, 2020; Lee and You, 2019; Ahmed et al., 2021b). Then one either directly solves the projected governing equation on the manifold (Carlberg et al., 2011, 2013) or learns the projected dynamics from the data on the manifold with either Long Short-Term Memory (LSTM) (Mohan and Gaitonde, 2018), Artificial Neural Network (Pan and Duraisamy, 2018b; San et al., 2019; Lui and Wolf, 2019), polynomials (Qian et al., 2020; Brunton et al., 2016; Peherstorfer and Willcox, 2016). Other variants include jointly learning dynamics together with the manifold (Champion et al., 2019; Kalia et al., 2021; Lusch et al., 2018; Takeishi et al., 2017; Yeung et al., 2019; Otto and Rowley, 2019; Pan and Duraisamy, 2020; Lange et al., 2021; Mardt et al., 2020), and closure modeling to account for non-Markovian effects (Pan and Duraisamy, 2018a; Wang et al., 2020; Maulik et al., 2020; Ahmed et al., 2021a). Manifold-based approaches first reduce the prohibitively large spatial degrees of freedom (e.g., $O(10^4) - O(10^{11})$ in fluid dynamics) into a moderate number (e.g., $O(10) - O(10^2)$) by learning a low-dimensional representation on which we project and solve the PDE, thus inheriting the physics (Carlberg et al., 2011), or simply performing time-series modeling on the reduced manifold (Xu and Duraisamy, 2020) or modal expansion (Taira et al., 2017). Equivalently, this preprocessing step can be viewed as learning a time-dependent vector-valued low dimensional representation of a spatio-temporal field.

More broadly, learning an effective low-dimensional representation is often domain specific, for example, using a real-valued matrix with RGB channels for representing images in computer vision (Szeliski, 2010), Word2Vec for representing words in natural language processing (NLP) (Mikolov et al., 2013), spectrograms for representing audio signal in speech recognition (Flanagan, 2013), among others.

Manifold-based methods have several drawbacks when applied to realistic spatio-temporal data. Specifically, practical engineering applications require handling variable geometry, non-uniform grid resolutions, adaptive meshes, and/or parametric dependencies. This includes data from incompressible flow in the unbounded domain (Yu et al., 2022), combustion (Bell and Day, 2011), astrophysics (Almgren et al., 2010), multiphase flows (Sussman, 2005), and fluid-structure interactions (Bhalla et al., 2013)) which are generated with advanced meshing techniques such as *adaptive mesh refinement* (AMR) (Berger and Oliger, 1984), and/or overset grids (Chan, 2009). Such meshes typically change with time or parameters (e.g., Mach number dependency on shock location) in order to efficiently capture the multi-scale phenomena[1], which violates the requirement of common SVD approaches. While CNNs require preprocessing the flowfield as an image, i.e., voxel representation with a uniform Cartesian grid, in order to perform discrete convolution, which is affordable in 2D but becomes increasingly expensive in 3D (Park et al., 2019) due to cubic memory requirements. The memory footprint limits the resolution to 64^3 typically (Mescheder et al., 2019) unless one resorts to an optimized parallel implementation of 3D CNN on clusters (Mathuriya et al., 2018). Additionally, such *uniform* processing is inherently inconsistent with the *nonuniform* multi-scale nature of PDEs and can decrease the prediction accuracy of downstream tasks, e.g., failing to accurately recover total lift/drag from flowfield predictions (Bhatnagar et al., 2019) or efficiently capture small-scale geometry variations that can trigger critical physics phenomena. This leads us to pursue a mesh-agnostic and expressive paradigm beyond SVD and CAE for dimensionality reduction of spatio-temporal data.

Recent advances in mesh-based methods[2] have shown promising results either with discretization-invariant operator learning (Li et al., 2020a,c,b) or meshes based on the numerical solver (Pfaff et al., 2020; Sanchez-Gonzalez et al., 2020; Xu et al., 2021). On the other hand, a mesh-agnostic framework (e.g., physics-informed neural network (PINN) (Raissi et al., 2020)) has been successfully applied to solving canonical PDEs on problems where mesh-based algorithms can be cumbersome due to, for instance, arbitrary geometry (Berg and Nyström, 2018; Sun et al., 2020) and/or high-dimensionality (Sirignano and Spiliopoulos, 2018). In addition to leveraging the PDE governing equation, it successfully leverages the structure of multilayer perceptrons (MLPs) with the coordinate as input and the solution field as output. It can be viewed as using an adaptive global basis in space-time to approximate the solution with the known PDE, instead of the traditional local polynomial basis inside a cell, e.g., finite element analysis (Hughes, 2012). Concurrently, such MLP structure has been also employed in computer graphics community for learning 3D shape representations (Park et al., 2019; Mescheder et al., 2019), scenes (Mildenhall et al., 2020; Sitzmann et al., 2020; Tancik et al., 2020). A closely related work called Mesh-

1. for some systems (Bryan et al., 2014), AMR is the only way to make such simulation possible.
2. In this paper, "mesh-based" in our paper is the same as "graph-based". A key characteristics of graph-based method is that the online computational cost scales with the size of the graph (i.e., how much grid points) (Chen et al., 2021). While manifold-based methods only access mesh during postprocessing.

freeFlowNet (MFN) (Esmaeilzadeh et al., 2020) uses CNN and coordinate-based MLP to perform super-resolution for fluid problems. MFN first uses a 2D CNN to extract features from coarse-scale spatial-temporal field. Then, the extracted features are concatenated with t, x, y as an augmented input to a MLP, which outputs the high-resolution measurement at t, x, y.

Motivated by the above seminal works, we introduce a mesh-agnostic representation learning paradigm called *neural implicit flow* (NIF) that exploits the expressiveness and flexibility of the multilayer perceptrons (MLPs) for dimensionality reduction of parametric spatio-temporal fields. In section 2, we present the NIF paradigm and several derived problem-specific frameworks. In section 3, we demonstrate the following capabilities of NIF:

1. NIF enables an efficient scalable 3D nonlinear dimensionality reduction on spatial-temporal datasets from arbitrary different meshes. Example includes a three-dimensional fully turbulent flows with over 2 million cells. (see section 3.3)

2. NIF also enables modal analysis of spatial-temporal dynamics on adaptive meshes. As an example, we explore dynamic mode decomposition on the fluid flow past a cylinder with adaptive mesh refinement (see section 3.5).

NIF also provides a performance improvement in the following applications on several canonical spatio-temporal dynamics:

1. NIF generalizes 40% better in terms of root-mean-square error (RMSE) than a generic mesh-agnostic MLP in terms of mesh-agnostic surrogate modeling for the parametric Kuramoto–Sivashinsky PDE under the same size of training data or trainable parameters (see section 3.1)

2. NIF outperforms both (linear) SVD and (nonlinear) CAE in terms of nonlinear dimensionality reduction, as demonstrated on the Rayleigh-Taylor instability on adaptive mesh with a factor from 10 times to 50% error reduction. (see section 3.2)

3. Compared with the original implicit neural representation which takes *all* information (including time t) into a single feedforward network (Sitzmann et al., 2020; Lu et al., 2021c), NIF enables efficient spatial sampling with 30% less CPU time and around 26% less memory consumption under the same level of accuracy for learning the aforementioned turbulence dataset. (see section 3.4)

4. NIF outperforms the state-of-the-art method (POD-QDEIM (Drmac and Gugercin, 2016)) in the task of data-driven sparse sensing with 34% smaller testing error on the sea surface temperature dataset. (see section 3.6)

Finally, conclusions and future perspectives of NIF are presented in section 4.

2. Neural Implicit Flow

We begin by considering 3D spatial-temporal data with varying time/parameters, namely $\mathbf{u}(\mathbf{x}; t, \boldsymbol{\mu}) \in \mathbb{R}^n$, with spatial coordinate $\mathbf{x} \in \mathbb{R}^3$, time $t \in \mathbb{R}^+$, and parameters $\boldsymbol{\mu} \in \mathbb{R}^d$ (e.g.,

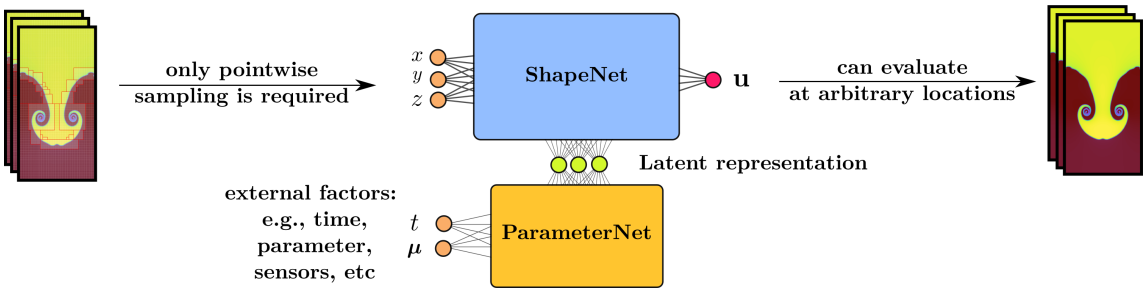

Figure 1: Neural implicit flow framework for dimensionality reduction of spatio-temporal field from PDE. NIF uses the bottleneck layer, which is linearly mapped to weights $\mathscr{W}$ (and biases $\mathscr{B}$) of ShapeNet, as the latent representation. ParameterNet correlates external factors with such a latent representation.

Reynolds number). Without loss of generality, consider a supervised learning problem: using an L-layer MLP with spatial coordinate $\mathbf{x}$ as input to fit a *single* spatial realization at an arbitrary time t_0 and parameter $\boldsymbol{\mu}_0$, i.e., $\mathbf{u}(\mathbf{x}; t_0, \boldsymbol{\mu}_0)$. An L-layer MLP with $\mathbf{x}$ as input is a vector-valued function defined as $\mathbf{u}_{\text{MLP}}(\mathbf{x}; \mathscr{W}, \mathscr{B}) = \mathbf{W}_L \eta_L + \mathbf{b}_L$, $\forall l \in \{1, \ldots, L-1\}$ with $\eta_l = \sigma(\mathbf{W}_l \eta_{l-1} + \mathbf{b}_l)$. The first layer weight $\mathbf{W}_1$ has size $n_h \times 3$, and the remaining hidden layer weight $\mathbf{W}_l$ has size $n_h \times n_h$. The last layer weight $\mathbf{W}_L$ has size $n \times n_h$. Biases are denoted as b_l where subscript l denotes the index of layer. $\eta_0 = \mathbf{x}$. The set of weights and biases are defined as $\mathscr{W} = \{\mathbf{W}_l\}_{l=1}^L$ and $\mathscr{B} = \{\mathbf{b}_l\}_{l=1}^L$ correspondingly. The activation $\sigma : \mathbb{R} \mapsto \mathbb{R}$ is a non-linear continuous function. One can use gradient-based optimization to find the weights $\mathscr{W}$ and biases $\mathscr{B}$ that minimize the discrepancy between $\mathbf{u}_{\text{MLP}}$ and the single snapshot data $\mathbf{u}$ at t and $\boldsymbol{\mu}$:

$$\min_{\mathscr{W}, \mathscr{B}} \int \mathcal{L}(\mathbf{u}_{\text{MLP}}(\mathbf{x}; \mathscr{W}, \mathscr{B}), \mathbf{u}(\mathbf{x}; t_0, \boldsymbol{\mu}_0)) d\nu(\mathbf{x}; t_0, \boldsymbol{\mu}_0), \tag{1}$$

where $\mathcal{L}$ is a loss function (e.g., mean squared error, mean absolute error) and ν is some spatial measure which can depend on $t_0, \boldsymbol{\mu}_0$. This minimization leads to a key observation: a well-trained set of weights $\mathscr{W}$ and biases $\mathscr{B}$ of the MLP fully determines a spatial field that approximates the target field, in a mesh-agnostic sense. Such a trained MLP (what we call *ShapeNet*) is closely related to the so-called *neural implicit* representation (Sitzmann et al., 2020), and works in computer graphics (Park et al., 2019; Mescheder et al., 2019) used such an MLP to fit signed distance functions (SDFs) of a desired surface, which is *implicitly* defined by the zero iso-surface of the SDF.

To explicitly correlate the corresponding spatial fields with external factors of interests, such as time, parameters, and sparse sensor measurements, we use a second MLP (denoted as *ParameterNet*) to learn mappings from these parameters to weights $\mathscr{W}$ and biases $\mathscr{B}$. Note that the typical output dimension of ParameterNet, e.g.,, the total number of scalars in $\mathscr{W}$ and $\mathscr{B}$, is above thousands to tens of thousands. Dimensionality reduction assumes the existence of a rank-r subspace that can approximate the spatio-temporal data. The reduced coordinates are denoted as $\zeta_1, \ldots, \zeta_r$. One can simply consider a *linear* output

layer in ParameterNet after a bottleneck layer of width r. Once $\zeta_1, \ldots, \zeta_r$ are determined, $\mathscr{W}$ and $\mathscr{B}$ in the ShapeNet are completely determined, which in turn determines the spatial *"flow"* field conditioned on the time t or more generally, any other external parameters. As a result, the bottleneck layer in fig. 1 can be viewed as an r-dimensional latent representation, similar to the bottleneck layer in generic autoencoders (Goodfellow et al., 2016)[3]. In summary, we can find a mesh-agnostic representation of parametric spatio-temporal fields as in the aforementioned manifold-based methods, where spatial complexity, e.g., coherent structures, is explicitly decoupled from temporal and parametric complexity, e.g., chaos and bifurcations.

Note that any neural network that generates the weights and biases of another neural network is generally called a *hypernetwork* (Ha et al., 2016). This concept has been applied in image (Sitzmann et al., 2020) and language modeling (Ha et al., 2016) and its inspiration can be dated back to control fast-weight memory (Schmidhuber, 1992) in the early 90's. Therefore, NIF can be viewed as a special family of hypernetworks for MLP with only spatial input $\mathbf{x}$ while any other factors, e.g., time t, parameter $\boldsymbol{\mu}$, are fed into the hypernetwork. This implies the dimensionality reduction is only performed towards spatial complexity, which is the biggest cause of computational and storage challenges for non-linear PDEs. Thus, spatial complexity is decoupled from other factors.

A comparison of the NIF architecture with other recent frameworks of machine learning for PDEs is shown in fig. 2. DeepONet (Lu et al., 2021a; Wang et al., 2021b) pioneered the learning of general operators associated with PDEs from data. It has been a building block of a DeepM&MNet (Cai et al., 2021), which has been applied to data assimilation in electroconvection (Cai et al., 2021) and hypersonic flows (Mao et al., 2021). Interestingly, the structure of DeepONet can be viewed as that of NIF with only the last linear layer of the ShapeNet determined by the so-called trunk net. To highlight the expressiveness of NIF, we compare a tiny NIF with only 51 parameters and a moderate DeepONet with 3003 parameters on a 1D modulated traveling sine wave. Figure 3 shows no visual difference between the ground truth and NIF.

Alternatively, neural operators (Li et al., 2020b,c,a; Kovachki et al., 2021) are discretization-independent mesh-based frameworks for learning solution operators of PDEs beyond the straightforward CNN approach on a fixed grid (Zhu and Zabaras, 2018; Long et al., 2018). Neural operators have been successful in recovering Green's function, learning chaotic PDE systems (Li et al., 2021a), and leveraging known PDEs such as Navier-Stokes equations (Li et al., 2021b). It inherits translational and rotational invariance using graph neural networks, as opposed to NIF and DeepONet, which might be advantageous in the small data regime.

Distinct from the above pioneering works aimed at learning solution operators of PDEs, NIF emphasizes a scalable nonlinear dimensionality reduction paradigm that outperforms SVD and CAE in dealing with complex parametric spatial-temporal data (e.g., turbulence) on *arbitrary mesh structure*. In the following subsections, we will develop problem-specific, NIF-based frameworks on a variety of learning tasks.

3. However, note that we haven't introduced "encoder" in fig. 1. We will introduce an "encoder" based on sparse sensors in section 3.2.

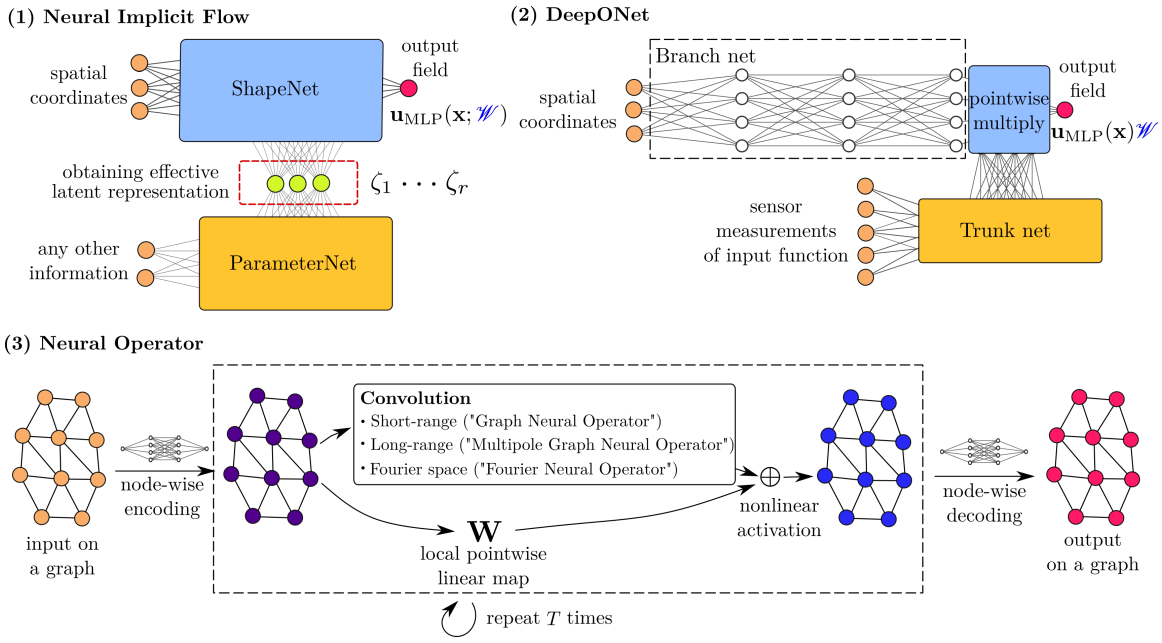

Figure 2: Comparion of NIF with DeepONet (Lu et al., 2021a) and Neural Operator (Li et al., 2020b,c,a). Notice that the skeleton of DeepONet can be viewed as that of a *last layer parameterized* NIF. Information from sensor measurements is parameterized as a matrix multiplication while NIF is parameterized everywhere inside the ShapeNet. Neural Operator learns continuous nonlinear convolution in order to be mesh-invariant and spatial invariant. Neural Operator and DeepONet focus on learning solution operator while NIF focus on learning a reduced latent representation of spatial-temporal dynamics on arbitrary mesh.

2.1 Data-fit parametric surrogate modeling for PDEs

Consider a class of non-linear parametric PDEs, $\partial \mathbf{u}/\partial t = \mathcal{G}(\boldsymbol{\mu}, \mathbf{u}, \nabla \mathbf{u}, \nabla^2 \mathbf{u}, \ldots)$, $(\mathbf{x}, t, \boldsymbol{\mu}) \in \Omega = \mathcal{X} \times \mathcal{T} \times \mathcal{D}$. Here $\mathcal{X} \subset \mathbb{R}^3, \mathcal{T} \subset \mathbb{R}^+, \mathcal{D} \subset \mathbb{R}^d$ and $\mathcal{G}$ is a non-linear function or operator in general. An example of data-fit parametric surrogate modeling (Benner et al., 2015; Sobieszczanski-Sobieski and Haftka, 1997; Frangos et al., 2010; Amsallem et al., 2013; Bhatnagar et al., 2019) is to find an approximated relation between the parameter $\boldsymbol{\mu}$ and the corresponding PDE solution $\mathbf{u}(\mathbf{x}; t, \boldsymbol{\mu})$ under fixed initial and boundary conditions. After the data-fit surrogate model is trained, it is expected to make predictions for an unseen input possibly in *real-time*[4]. An illustration of the method is shown in fig. 4. This is attractive for many engineering tasks that require *many-query* analyses such as optimization, uncertainty

4. Note that our major goal in the following PDE examples is to demonstrate NIF for dimensionality reduction, not for learning solution operator of PDEs as is advocated in other neural network architectures. However, we present the following example just to show the capability of efficient surrogate modeling where computational cost is independent of mesh size. Note that existing graph-based frameworks still suffer from poor scaling of computational cost with mesh size..

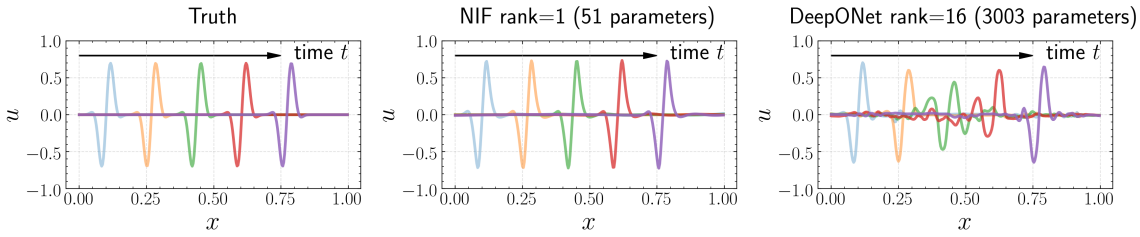

Figure 3: Comparison between DeepONet and NIF on a 1D modulated traveling sine wave traveling from left to right. The wave is described as $u(x,t) = e^{-c_0(x-x_0-ct)^2}\sin(\omega(x-x_0-ct))$ with $c_0 = -1000, c = 0.012, \omega = 70$. Data is generated by uniformly sampling x in $[0,1]$ with 300 samples and t in $[0,70]$ with 20 samples. Data is then standard normalized for training. Configuration of NIF: ShapeNet with input x: 1-2-2-2-1, ParameterNet with input t: 2-2-1-19. Configuration of DeepONet: branch net with input t: 1-30-30-17. trunk net with input x: 1-30-30-16-1.

quantification, and control. In contrast to more physics-based models (Benner et al., 2015), a data-fit model simplifies the surrogate modeling for PDEs as a high-dimensional regression without access to prior knowledge. Despite the many disadvantages, including large sample complexity, lack of interpretability etc., it is the simplest and most widely used type of surrogate model (Qian et al., 2020; Loiseau et al., 2021).

As illustrated in fig. 4, we apply NIF to the above parametric surrogate modeling by simply allowing the weights $\mathscr{W}$ and biases $\mathscr{B}$ to depend on time and parameters through the ParameterNet f_{MLP},

$$\left[\text{vec}^\top(\mathbf{W}_1) \quad \dots \quad \text{vec}^\top(\mathbf{W}_L) \quad \mathbf{b}_1^\top \quad \dots \quad \mathbf{b}_L^\top\right] = f_{\text{MLP}}(t, \boldsymbol{\mu}; \boldsymbol{\Theta}), \tag{2}$$

where $f_{\text{MLP}} : \mathbb{R}^+ \times \mathbb{R}^d \mapsto \mathbb{R}^m$ is an MLP with its own weights and biases denoted as $\boldsymbol{\Theta}$, vec is the matrix vectorization, and m is the total number of unknown parameters in $\mathscr{W}$ and $\mathscr{B}$. Again, it is important to note that the width of the layer *before* the last linear layer of f_{MLP} approximates the total number of parameters we need to optimize. We set the bottleneck width $r = d + 1$ which equals the input dimension of ParameterNet. Hence, the rank of $\boldsymbol{\Theta}$ is $d + 1$. We denote the above dependency of weights and biases simply as $\mathscr{W}(t, \boldsymbol{\mu}; \boldsymbol{\Theta}), \mathscr{B}(t, \boldsymbol{\mu}; \boldsymbol{\Theta})$. By considering eq. (2), and extending eq. (1) from $\mathcal{X}$ to Ω, we arrive at the following minimization formulation,

$$\min_{\boldsymbol{\Theta}} \int \mathcal{L}(\mathbf{u}_{\text{MLP}}(\mathbf{x}; \mathscr{W}(t, \boldsymbol{\mu}; \boldsymbol{\Theta}), \mathscr{B}(t, \boldsymbol{\mu}; \boldsymbol{\Theta})), \mathbf{u}(\mathbf{x}, t, \boldsymbol{\mu}))d\nu(\mathbf{x}, t, \boldsymbol{\mu}), \tag{3}$$

where ν now becomes a measure in Ω. A natural choice of ν to cover the domain of interest is the empirical distribution based on the numerical discretization of Ω where the data comes from. For example a tensor product of discretizations independently in $\mathcal{X}, \mathcal{T}$ and $\mathcal{D}$ leads to $\nu(\mathbf{x}, t, \boldsymbol{\mu}) = \sum_{i=1}^{M_{\mathbf{x}}} \sum_{j=1}^{M_t} \sum_{k=1}^{M_{\boldsymbol{\mu}}} \delta(\mathbf{x}_i, t_j, \boldsymbol{\mu}_k)/(M_{\mathbf{x}} M_t M_{\boldsymbol{\mu}})$ where δ denotes Dirac measure. Here total number of the spatial mesh points is $M_{\mathbf{x}}$, the number of snapshots in time is M_t,

and the number of parameters selected is $M_{\boldsymbol{\mu}}$. Once a proper ν is chosen, eq. (3) can be solved via gradient-based method, e.g., Adam optimizer (Kingma and Ba, 2014).

It should be noted that eq. (3) permits using any kind of proper empirical measure to cover Ω. As shown in fig. 1, this can be especially advantage for problems where an efficient adaptive mesh (e.g., AMR, Overset), moving mesh (e.g., fluid-structure interaction) or simply parameter-dependent mesh (e.g., varying discontinuity with parameters) is adopted. It is a distinct feature that makes NIF different from previous pipelines of dimensionality reduction (Xu and Duraisamy, 2020; Mohan et al., 2019) with CAE (i.e., a homogeneous isotropic spatial measure) and SVD (i.e., a tensor product measure structure between $\boldsymbol{\mu}$ and $\mathbf{x}$). In this paper, we take $\mathcal{L}$ as the mean square error and rewrite $\nu(\mathbf{x}, t, \boldsymbol{\nu}) = \sum_{i=1}^{M} \delta(\mathbf{x}_i, t_i, \boldsymbol{\nu}_i)/M$ as the most general form[5]. So eq. (3) becomes the standard least-squares regression,

$$\min_{\boldsymbol{\Theta}} \frac{1}{M} \sum_{i=1}^{M} \left(\mathbf{u}_{\text{MLP}}(\mathbf{x}_i; \mathscr{W}(t_i, \boldsymbol{\mu}_i; \boldsymbol{\Theta}), \mathscr{B}(t_i, \boldsymbol{\mu}_i; \boldsymbol{\Theta})) - \mathbf{u}(\mathbf{x}_i, t_i, \boldsymbol{\mu}_i)\right)^2 . \tag{4}$$

2.2 Learning representations for multi-scale spatial-temporal data

As described in section 2, ShapeNet is a general MLP with activation function σ still to be determined. Given the universal approximator theorem (Hornik et al., 1989), the choice seems to be initially arbitrary. However, there is a significant and often overlooked difference between "what can MLP approximate" and "what can MLP efficiently learn with gradient-based algorithms". Standard MLPs with common activation functions, such as ReLU, tanh, swish, sigmoid, have been recently observed and proven (Tancik et al., 2020) to suffer from extreme inefficiency on learning high-frequency functions even with increased network complexity. Indeed, the failure of standard MLPs on high frequency datasets is a well-known phenomenon called *spectral bias* (Rahaman et al., 2019) or *F-principle* (Xu et al., 2019). Recall that in fig. 1, ShapeNet critically relies on an MLP with input $\mathbf{x}$ approximating the "shape" of spatial data, which can be problematic in the case of fluid flows with high wavenumber content, e.g., eddies, hairpin vorticies. Note that this issue is also relevant to PINNs (Wang et al., 2021a), which might explain the challenges of using data-free PINNs to solve Naiver-Stokes in fully turbulent regimes.

Recent advances in computer graphics propose remedies that uses Fourier features (Tancik et al., 2020) or ω_0-scaled sine activation functions[6] in MLP, called SIREN (Sitzmann et al., 2020), to learn high frequency content. The former Fourier feature approach has been recently introduced in the PINN community by Wang et al. (2021a) with the need to tune the length scale of the Fourier features for each dataset. Here we take the latter approach since we empirically found SIREN is much less sensitive to hyper-parameters of the network compared to the original Fourier feature approach (Tancik et al., 2020). Thus, as shown in fig. 5, we design ShapeNet using SIREN with a ResNet-like structure. However, implementing such an ω_0-scaled sine activation function requires a special type of initialization (Sitzmann et al., 2020) of both the MLP parameters and the uniform normalization

5. Taken the example of tensor product measure, the number of total training data points M is the product of resolution on each dimensions, i.e., $M = M_{\mathbf{x}} M_t M_{\boldsymbol{\mu}}$. Therefore, in practice M is typically larger than millions.
6. Activation function becomes $\sigma(\cdot) = \sin(\omega_0 \cdot)$. We use $\omega_0 = 30$ throughout this work.

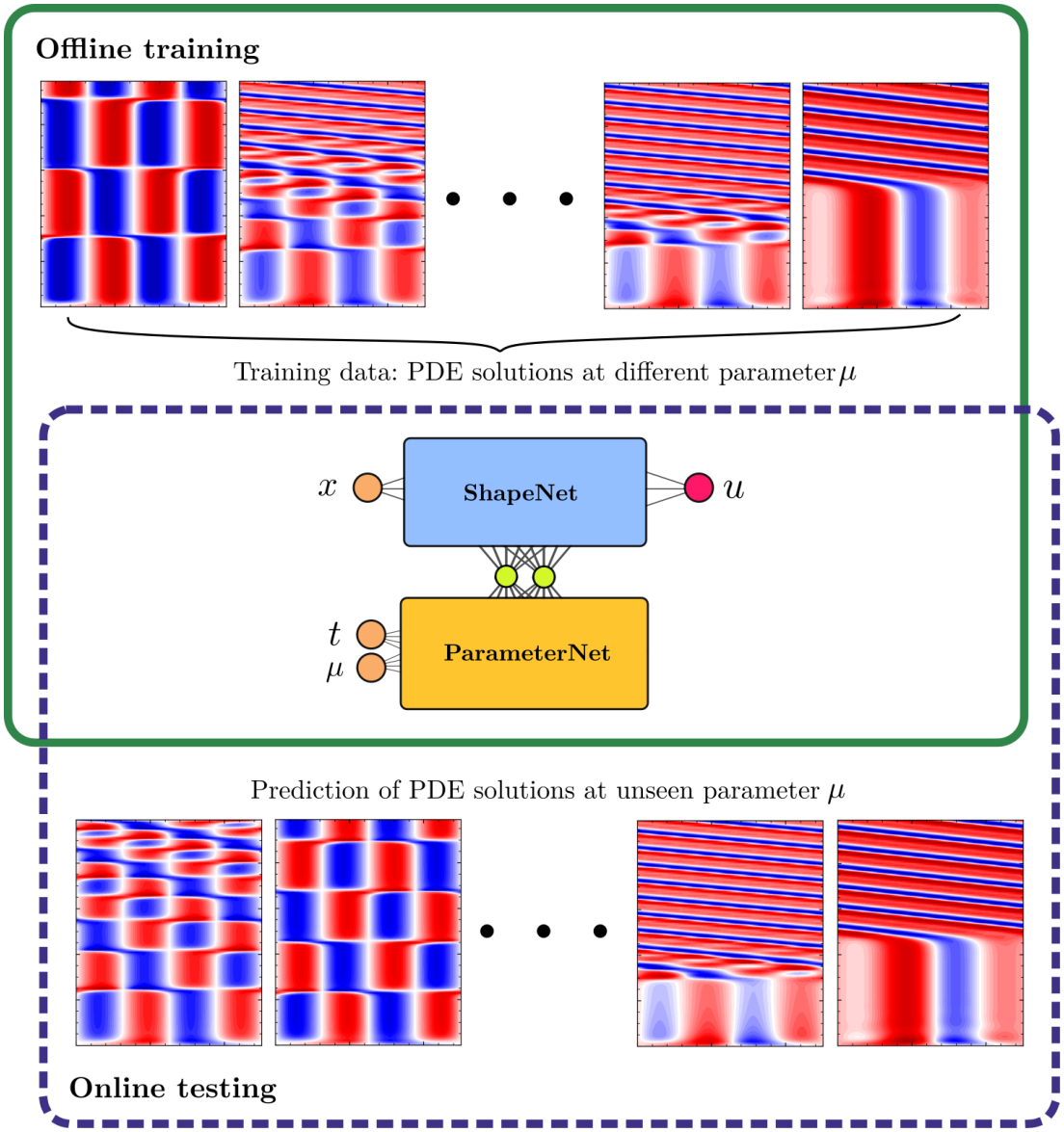

Figure 4: Application of NIF on mesh-agnostic surrogate modeling of parametric PDE. The case of 1D parametric Kuramoto-Sivashinsky equation is taken for illustration and further studied in section 3.1.

of the dataset. Implementation details are documented in appendix D. With SIREN for the ShapeNet, we can feed time t into ParameterNet to learn a compressed representation from spatial-temporal data in a mesh-agnostic way as illustrated in fig. 6. Furthermore, instead of feeding time t into ParameterNet, we can build an encoder just using sensor measurements from only a few locations as input to the ParameterNet. Such applications are demonstrated in sections 3.2 and 3.3.

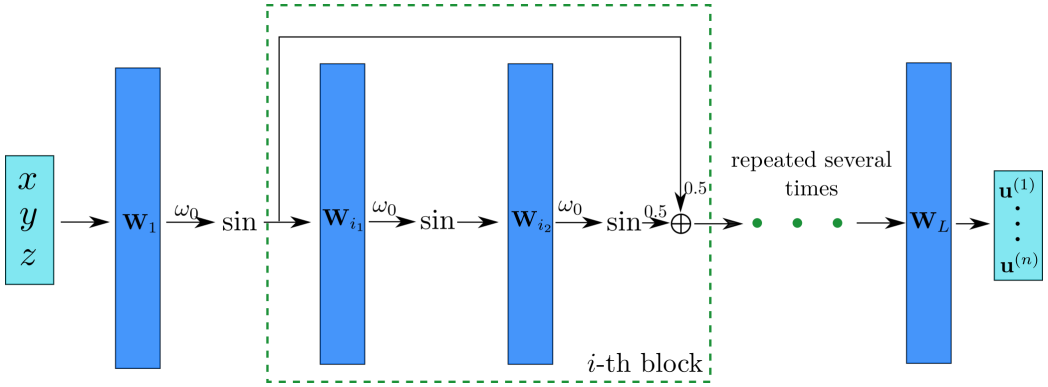

Figure 5: ShapeNet for multi-scale spatial data consists of SIREN (Sitzmann et al., 2020) with ResNet-like structure (Lu et al., 2021c) and ω_0-scaled sine function. Input is spatial coordinate x, y, z. Output is a n-dimensional vector. Biases are omitted in the figure for clarity. Green dashed line indicates the i-th block. Such structure is repeated several times downstream.

2.3 Learning mesh-agnostic linear representations

The *proper orthogonal decomposition* (POD) (Lumley, 1967) was introduced to the fluid dynamics community by Lumley in 1965 in order to give a mathematical definition of "large eddy" by applying a Karhunen-Loeve expansion (Loeve, 1955) to turbulent velocity fields. Formally, the first r POD modes are determined by the first r eigenfunctions of an integral operator with the cross-correlation kernel of the turbulent velocity field (George, 1988). In practice, the SVD is often used to obtain a discrete realization of POD modes (Brunton and Kutz, 2019). Such POD modes not only find an r-dimensional linear subspace that optimally minimizes the L^2 projection error (as shown in eq. (5)) but also provides a sequence of ordered basis weighted by their singular values (Djouadi, 2008).

$$\min_{\boldsymbol{\psi}_1,\ldots,\boldsymbol{\psi}_r} \iint \sum_{l=1}^{n} \left(\mathbf{u}^{(l)}(\mathbf{x},t) - \sum_{i=1}^{r} \alpha_i(t) \boldsymbol{\psi}_i^{(l)}(\mathbf{x}) \right)^2 d\mathbf{x}dt, \tag{5}$$

subject to the constraints $\alpha_i(t) = \int \sum_{l=1}^{n} \mathbf{u}^{(l)}(\mathbf{x},t)\boldsymbol{\psi}_i^{(l)}(\mathbf{x})d\mathbf{x}$, $\int \sum_{l=1}^{n} \boldsymbol{\psi}_i^{(l)}(\mathbf{x})\boldsymbol{\psi}_j^{(l)}(\mathbf{x})d\mathbf{x} = \delta_{ij}$, for $i = 1, \ldots, r$, where the superscript l denotes l-th component and δ_{ij} is the Kronecker delta. POD via SVD relies on a fixed mesh in order to provide a closed-form discrete approximation of the POD modes. If the mesh changes with time and/or parameters, which is the cases for many problems of interest (Teyssier, 2002; Bryan et al., 2014; Vay et al., 2004), then the SVD-based approaches are ill-suited for many downstream tasks such as modal analysis of fluid flows (Taira et al., 2017) and reduced-order modeling (Benner et al., 2015; Loiseau et al., 2021; Noack et al., 2003).

Since multi-scale features often appear in spatio-temporal data, we employ NIF with SIREN in section 2.2 for the applications considered in the rest of this paper. As shown in fig. 7, we first provide a framework based on NIF to *directly* approximate an optimal r-dimensional linear space of classical POD theory (Djouadi, 2008). The key observation

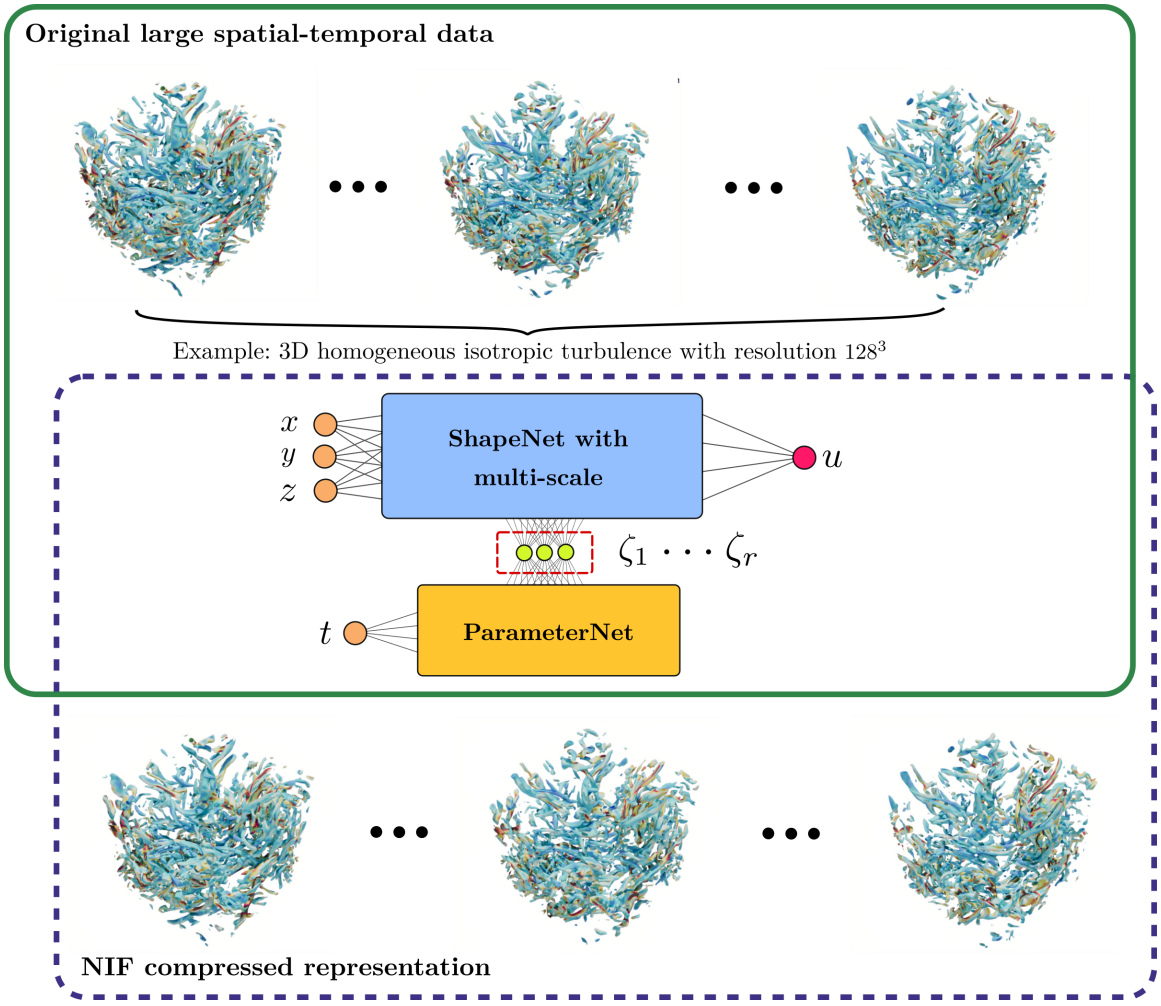

Figure 6: Application of NIF with SIREN on mesh-agnostic learning of a latent representation of large-scale spatial-temporal dataset. Iso-contour of Q-criterion colored by vorticity magnitude of 3D HIT is taken for illustration here and further studied in section 3.3.

is that we can use ParameterNet with input t to parameterize only the last layer weights and biases of ShapeNet while the rest of weights and biases of ShapeNet are determined by optimization. As shown in eq. (6), we arrive at an interpretable approximation of the original spatio-temporal field $\mathbf{u}(\mathbf{x}, t)$ as a sum of r products of spatial functions $\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_r$ and temporal modes $a_1, \ldots, a_r(t)$ parameterized by MLP,

$$\min_{\boldsymbol{\Theta}, \{\mathbf{W}_i, \mathbf{b}_i\}_{i=1}^{L-1}} \iint \sum_{l=1}^{n} \left(\mathbf{u}^{(l)}(\mathbf{x}, t) - \sum_{i=1}^{r} a_{\text{MLP},i}(t; \boldsymbol{\Theta}) \boldsymbol{\phi}_{\text{MLP},i}^{(l)}(\mathbf{x}; \{\mathbf{W}_j, \mathbf{b}_j\}_{j=1}^{L-1}) \right)^2 d\mathbf{x}dt. \quad (6)$$

The notations in eq. (6) is slightly different from eq. (5) in order to highlight that NIF only approximates the r-dimensional linear subspace rather than obtaining a set of ordered orthonormal spatial functions. Note that one needs to take the cell area into account when implementing *eq.* (6). To remove the effects of the slightly varying magnitude of the spatial basis learned with a neural network, we consider a normalizing spatial basis, such that $\tilde{\phi}^{(l)}_{\text{MLP},i} = \phi^{(l)}_{\text{MLP},i}/c_i$ where $c_i \triangleq \sqrt{\int \sum_{l=1}^{n} (\phi^{(l)}_{\text{MLP},i}(\mathbf{x}))^2 d\mathbf{x}}$ and $\zeta_i(t) \triangleq c_i a_{\text{MLP},i}(t)$. Since $\zeta(t) \in \mathbb{R}^r$ is the corresponding mesh-agnostic r-dimensional linear representation, one can effortlessly apply any existing SVD-based frameworks (e.g., SVD-DMD (Schmid, 2010)) on datasets with arbitrary varying meshes.

2.4 Mesh-agnostic data-driven non-linear sparse reconstruction

The goal of data-driven sparse reconstruction is to use limited sensor measurements to infer the entire high-dimensional system state, given *a priori* information of the low-dimensional manifold where system evolves. It has been widely applied in projection-based reduced order modeling, especially for large-scale nonlinear systems (also known as "hyper-reduction") where computing expensive nonlinear terms can be avoided. Currently POD-QDEIM (Drmac and Gugercin, 2016) is one of the most popular methods, which shows improved performance over classical compressive sensing techniques (Manohar et al., 2018). The idea of POD-DEIM (Chaturantabut and Sorensen, 2010) is to use least-square estimators for the latent representation by only measuring a few locations (e.g., sensors). Chaturantabut and Sorensen (2010) derived an error upper-bound of DEIM that indicates two contributions: 1) a spectral norm related to sensor selection and 2) projection error of the linear subspace. The idea of POD-QDEIM is to minimize the former contribution of the spectral norm with QR pivoting. Without loss of generality, given a scalar field u on $M_{\mathbf{x}}$ mesh points and M_t snapshots, the data matrix is

$$\mathbf{U} = \begin{bmatrix} u(\mathbf{x}_1, t_1) & \dots & u(\mathbf{x}_1, t_{M_t}) \\ \vdots & \vdots & \vdots \\ u(\mathbf{x}_{M_{\mathbf{x}}}, t_1) & \dots & u(\mathbf{x}_{M_{\mathbf{x}}}, t_{M_t}) \end{bmatrix} \in \mathbb{R}^{M_{\mathbf{x}} \times M_t}. \tag{7}$$

The corresponding reduced rank-r SVD is $\mathbf{U} \approx \mathbf{\Psi}_r \mathbf{\Sigma}_r \mathbf{V}_r^\top$. Near-optimal sensor placement can be achieved via QR factorization with column pivoting of the transpose of spatial basis,

$$\mathbf{\Psi}_r^\top \mathbf{C}^\top = \mathbf{Q}\mathbf{R}. \tag{8}$$

Note that the top p rows of $\mathbf{C} = \begin{bmatrix} \mathbf{e}_{\gamma_1} & \dots & \mathbf{e}_{\gamma_p} \end{bmatrix}^\top \in \mathbb{R}^{p \times M_{\mathbf{x}}}$ give a sparse measurement matrix, where $\mathbf{e}_i$ are the canonical basis vectors with a unit entry at index i and zero elsewhere. γ_i corresponds to the index of i-th best sensor location. One can recover the full field by least-square estimation of the latent representation from u at those sensors.

Given the success of POD-QDEIM, we can turn our attention to its error contribution, i.e., the projection error of the linear subspace, by replacing POD with NIF. We first use the optimized sensor location determined by POD-QDEIM. Then, as shown in fig. 8, given p sensor measurements $u(\mathbf{x}_{\gamma_1}), \dots, u(\mathbf{x}_{\gamma_p})$ as input for ParameterNet, and the ground true field u at $\mathbf{x}$, we end up with a standard supervised learning problem. Once the model

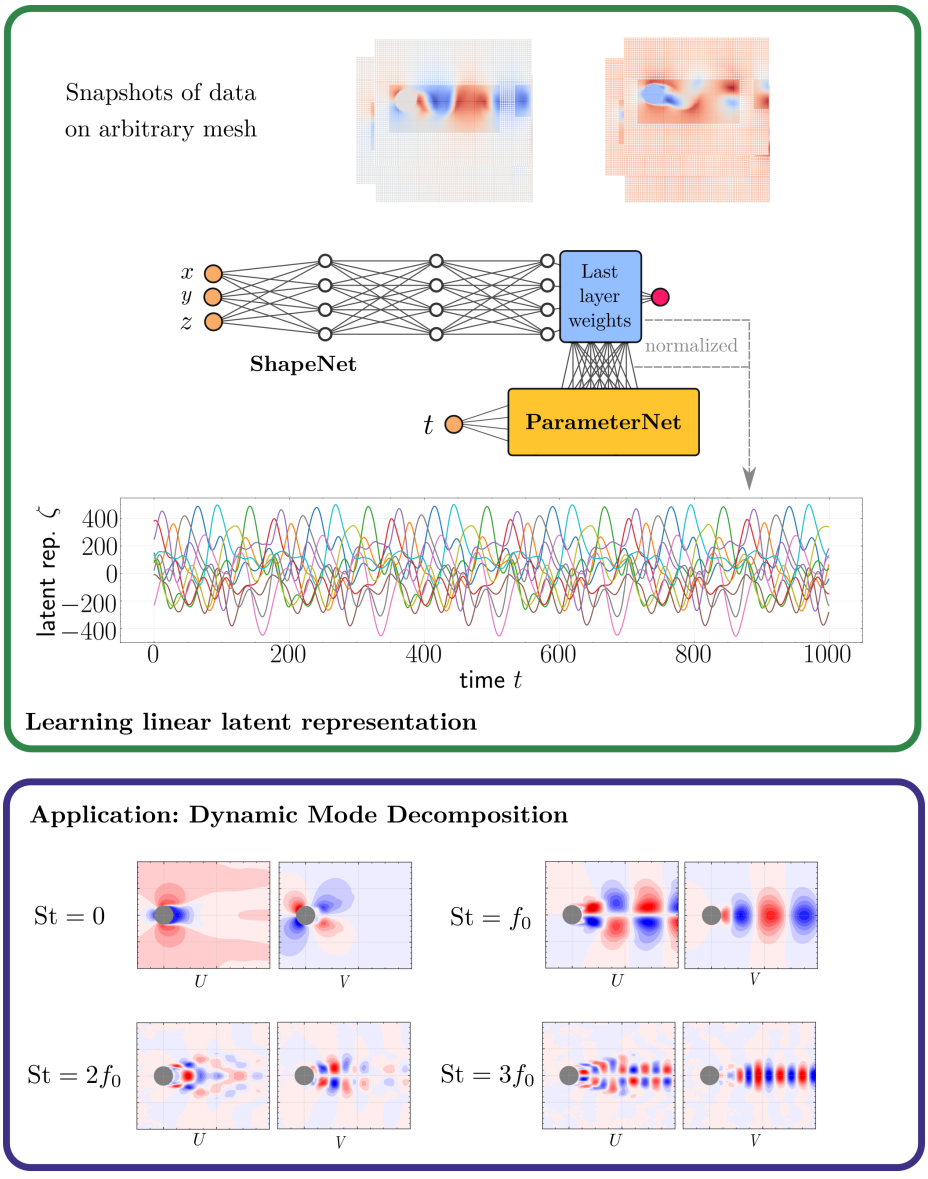

Figure 7: Application of NIF on mesh-agnostic learning of linear subspace. Here a flow over cylinder at $Re \approx 123$ is used for illustration. Once latent subspace is learned from spatio-temporal dataset with AMR. A standard DMD (Brunton and Kutz, 2019) is performed on the latent representation. f_0 denotes the fundamental frequency. When postprocessing the DMD mode shape, we choose a Cartesian grid with 500 uniform sampling points in each direction.

is trained, it can be used to predict the full spatial field u at any location for a given measurement from p sensors.

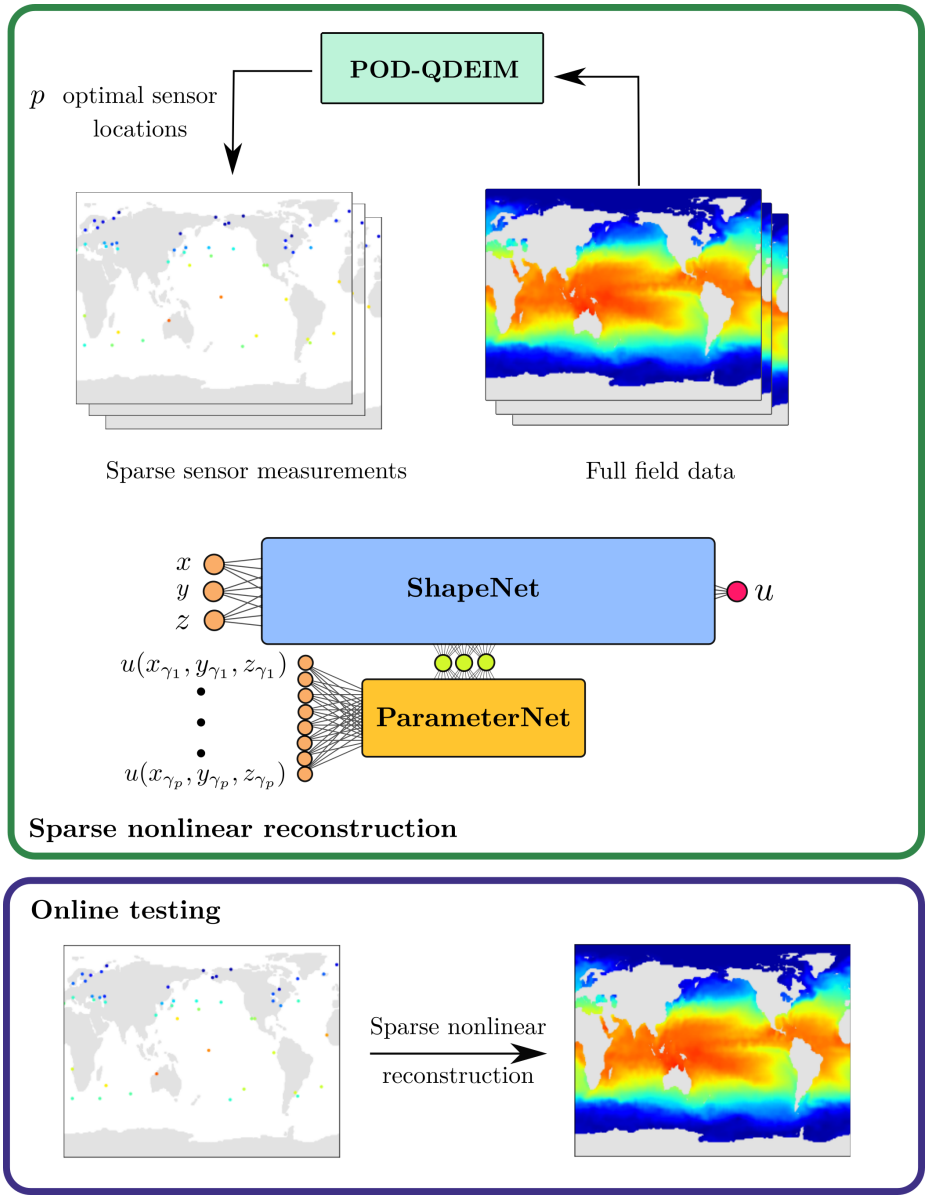

Figure 8: Application of NIF on mesh-agnostic data-driven sparse reconstruction.

3. Applications

The code and data for the following applications is available at https://github.com/pswpswpsw/paper-nif. The Python package for NIF is available at https://github.com/pswpswpsw/nif.

3.1 Learning parametric solutions of Kuramoto–Sivashinsky equation

We apply the data-fitting parametric surrogate modeling framework in eq. (4) on the 1D Kuramoto–Sivashinsky equation with periodic boundary condition,

$$u_t + uu_x + u_{xx} + \mu u_{xxxx} = 0, \quad u(0,t) = u(2\pi,t), \tag{9}$$

with varying parameter μ as shown in fig. 4. For the K-S equation, we fix the initial condition as $\sin(x)$ and vary μ from 0.2 to 0.28 which is not chaotic[7]. The training data consists of 20 points in the parameter μ space (i.e., 20 simulations with distinct μ). The testing data consists of 59 simulations with a finer sampling of μ. As shown in fig. 4, the system response when one varies μ from 0.2 to 0.28 is relatively smooth without any chaos. This makes it a well-posed problem for regression. The training data is preprocessed with standard normalization. Details are given in appendix B.1.

For NIF, we take 4 layers with units for ParameterNet as 2-30-30-2-6553 and 5 layers with units 1-56-56-56-1 with ResNet-like skip connection for ShapeNet. We empirically found such ResNet-like skip connections can help accelerate the convergence. Note that 6553 corresponds to the total number of weights and biases in the aforementioned ShapeNet. The swish activation function (Ramachandran et al., 2017) is adopted. As a comparison, we use a standard MLP with 5 layers with units 3-100-100-100-1, which is around the same number of model parameters with x, t, μ as input and output u. This can be viewed as a straightforward idea using PINNs (Raissi et al., 2020) as the regression without minimizing the PDE loss. Note that the same skip connections are employed as well. The model parameters are initialized with a truncated normal with standard deviation of 0.1 for both cases[8]. For the standard MLP case, two extra cases with tanh and ReLU activations are considered. We implemented both models in Tensorflow (Abadi et al., 2016). Note that we heavily rely on `einsum` in implementing NIF. We adopt the Adam optimizer (Kingma and Ba, 2014) with a learning rate of 1e-3, batch size of 1024 and 40000 epochs. To take training randomness into account and remove outliers, we take the average of 4 well converged trials for both network structures. As shown in table 1, NIF with Swish activations achieved better performance on both training and testing data than three MLP counterparts.

For simplicity, we fix the Swish activation function in the following. We first vary the size of the training data and retrain the models to evaluate data efficiency. We change the number of sampling points in parameter space for training data from the previous 20 to 15, 24, 29. As shown in fig. 9, NIF with Swish activation performs consistently better than MLP with Swish activation. To achieve the same level of testing error, NIF requires approximately half of the training data. Finally, we vary the number of model parameters from 7000 to 34000 while fixing the number of points in parameter space to 20. We then retrain the models to evaluate model expressiveness. As displayed in fig. 9, given the same number of parameters, NIF lowers the testing error by half compared to its MLP counterpart. Details of the comparisons are given in appendix C.1.

7. Although most often the K-S equation is simulated on chaotic regime, it is also famous for rich bifurcation phenomenon as its parameters change (Papageorgiou and Smyrlis, 1991).
8. We empirically find Such "small weights initialization" is an easy way to help the convergence of NIF. Systematic work in (Chang et al., 2019) on initialization of hypernetwork might further improve the result.

Table 1: Comparison between standard MLP and NIF in section 2.1 for surrogate modeling of 1D parametric PDE. RMSE below is averaged over all parameter μ.

Model	Train RMSE	Test RMSE
NIF (Swish)	$\mathbf{1.9 \times 10^{-2}}$	**0.64**
MLP (Swish)	2.2×10^{-2}	1.10
NIF (tanh)	3.9×10^{-2}	0.99
MLP (tanh)	3.7×10^{-2}	1.17
MLP (ReLU)	7.0×10^{-2}	1.27

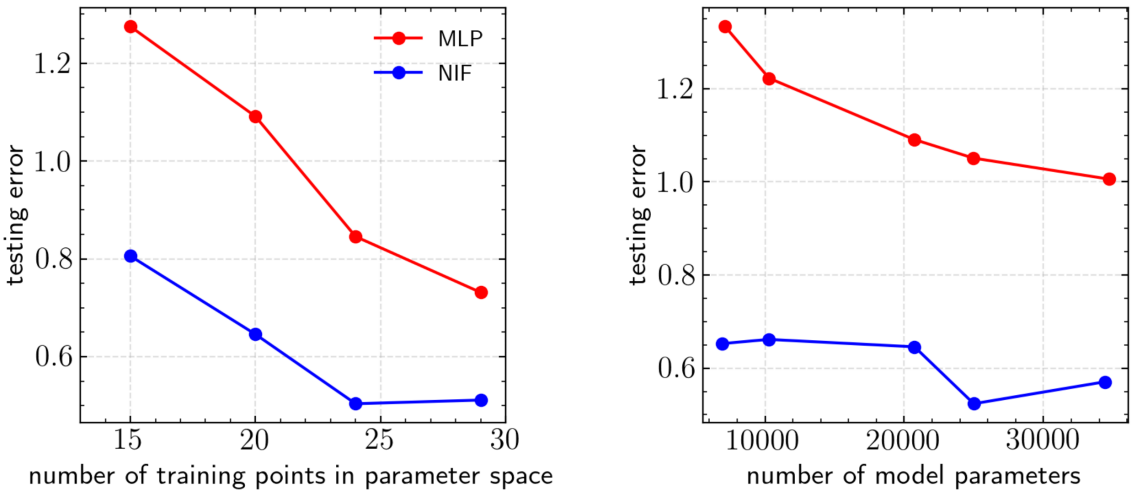

Figure 9: Trend of testing error against (Left) varying the size of training data; (Right): varying the number of model parameters.

3.2 Nonlinear dimensionality reduction of Rayleigh-Taylor instability from adaptive mesh

To highlight the advantage of NIF on multi-scale problems, here we compare the performance of SVD, CAE and a NIF-based autoencoder on reducing the dimensionality of the density field from the classical 2D Rayleigh-Taylor (R-T) instability. In this R-T problem, the interface between the high density fluid above and the low density fluid below is initially perturbed with a single mode profile of density. The CFD simulation is performed with CASTRO (Almgren et al., 2010), which is a compressible hydrodynamic code with AMR. Since the mesh changes with time, CAE or SVD can only be applied after projecting the data from the adaptive mesh onto a static fine mesh. Such preprocessing can introduce the so-called projection error and can be computationally challenging, especially in 3D. In the following, we take the static fine mesh as 128×256 for SVD and CAE. In contrast, NIF directly takes the pointwise raw data on the adaptive mesh for training.

The goal here is to encode the flow state onto an r-dimensional latent space, from which one can faithfully reconstruct the flow state. Note that r is the dimension of the latent subspace. Since we only take data from a single initial condition and the collection of data ends before the symmetry breaks, the minimal latent space dimension r that preserves the information in this case is one. We sample the flowfield uniformly in time and split such single trajectories into 84 training and 28 testing snapshots in a way that the testing snapshots fall in between the training snapshots. Details of data preparation are provided in appendix B.2.

Note that the structure of NIF in fig. 1 is only for a *decoder* rather than an *encoder*. Hence, unlike SVD and CAE, we still need to choose an encoder that feeds information to the network in order to let it discern one snapshot from the other. One can choose either a convolution layer with coarse data on Cartesian meshes just like CAE, or a neural network with a cluster of point-wise measurements at certain locations. Here we choose the latter: we consider 32 uniformly distributed sensors along the vertical axis in the middle of the flowfield.

For NIF, we take two ResNet blocks with 64 units in fig. 5 followed by a linear bottleneck layer of r units as ParameterNet. ShapeNet contains 2 ResNet-like blocks in fig. 5 with 128 units. ParameterNet takes the 32 sensor measurements as input and outputs 66,561 parameters as the weights and biases of the ShapeNet. While ShapeNet takes pointwise (x, y) coordinates as input and outputs the prediction of density u at (x, y). The output dimension of ParameterNet is 66,561 as the total number of weights and biases of the ShapeNet. To enforce a smooth latent representation, we use Jacobian and approximated Hessian regularization (Rifai et al., 2011) together with an initialization of small weights for the encoder, which we find empirically to be helpful.

For CAE, We choose a typical deep convolutional architecture used for fluid flows (Wiewel et al., 2019) with detailed setup in appendix C.2.1. Gradient-based optimization is performed with an Adam optimizer. The learning rate is 2e-5 for NIF and 1e-3 for CAE with a batch size of 3150 for NIF and 4 for CAE[9]. The total learning epoch is 10,000 for CAE and 800 for NIF.

In order to make quantitative comparisons, we project all of the predictions on testing data together with ground true data onto a very fine mesh with resolution of 256×512 by nearest-neighbor (for CAE and SVD) or direct sampling (for NIF). As shown in fig. 10 where a rank 1 reduction is performed, the prediction of NIF is better than SVD and CAE with varying static mesh resolution from 32×64, 64×128 and 128×256. When r increases from 1 to 8, it is expected that the errors from non-linear models do not change much due to the nature of single realization while that from the linear method decreases. On average, predictions of the CAE with three increasing resolutions lead to 10, 5, 1.7 times more error than that of NIF model. Further error analysis in appendix C.2.2 shows that the NIF-based autoencoder outperforms SVD mainly due to the lack of nonlinear expressiveness of SVD and it is better than CAE because of the excess projection error from the uniform Cartesian grid.

9. Note that pointwise data is fed to NIF while image snapshot data projected on a uniform Cartesian mesh is fed to CAE.

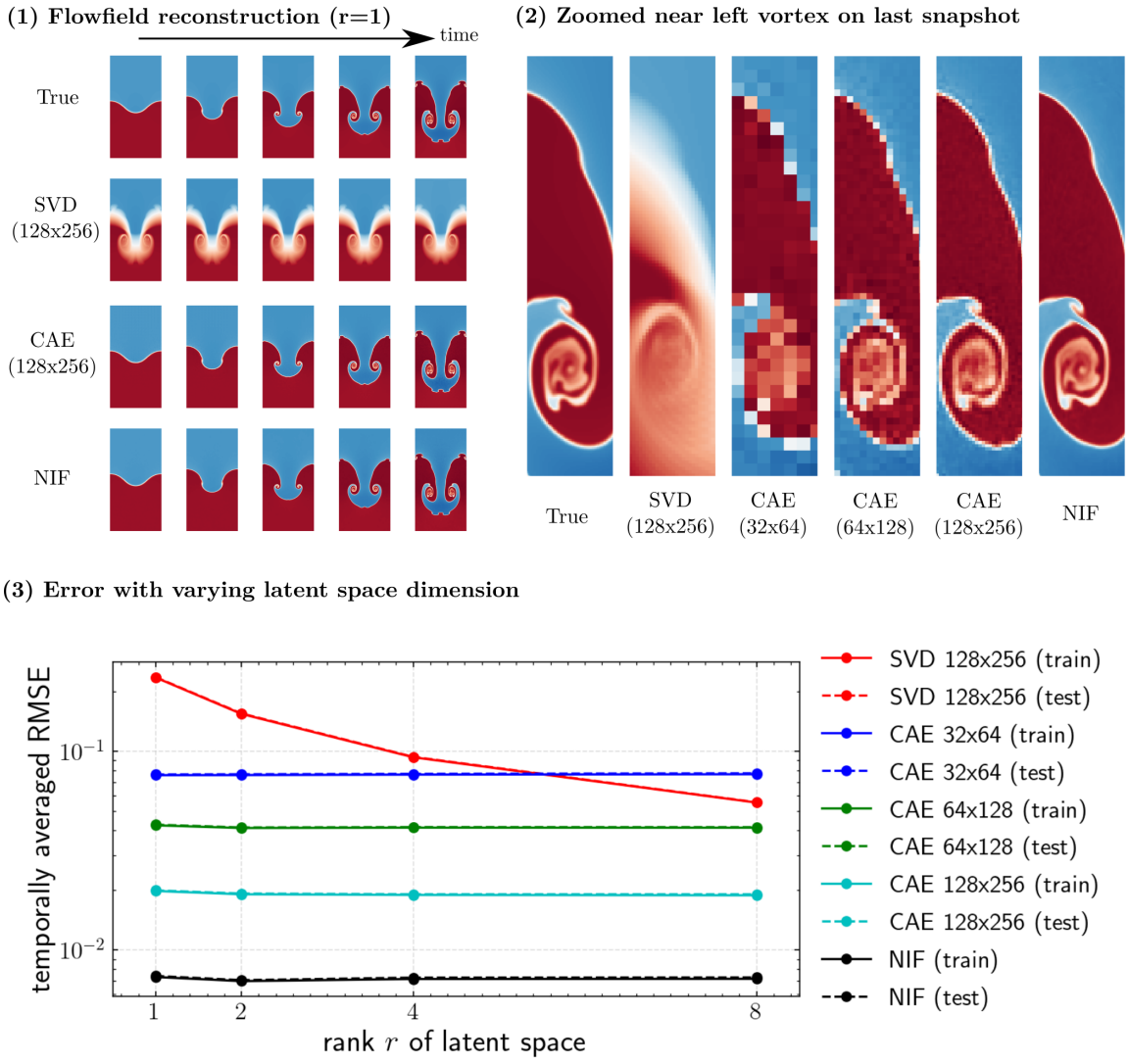

Figure 10: Comparison of SVD, CAE with 32x64 (low), 64x128 (middle) and 128x256 (high) resolution for dimensionality reduction on testing data of 2D Rayleigh-Taylor instability on an adaptive mesh. (1) evolution of flowfield reconstruction with $r = 1$ at five selected time in the testing phase. (2) Zoomed near the left vortex on the last snapshot. (3) Trend of temporally averaged RMSE with varying rank r for all models.

3.3 Learning spatially compressed representation for 3D homogeneous isotropic turbulence

Next, we apply NIF in section 2.2 to learn a spatially compressed representation of 3D multi-scale spatio-temporal field. Our goal is to find a vector-valued continuously differentiable

function $\mathbf{f}_{\text{MLP}}(\mathbf{x}; \boldsymbol{\Theta}(t))$ which "fits" the original spatial-temporal data. $\boldsymbol{\Theta}(t)$ is linearly determined by the time-dependent reduced coordinates $\zeta_1, \ldots, \zeta_r$. Note that r is several order of magnitude smaller than the number of mesh points. If it is achieved, one can efficiently send a snapshot of turbulence data at any time t by just transferring a r-dimensional vector $\boldsymbol{\Theta}(t)$ to the receiver. While the receiver just needs a "skeleton" ShapeNet and a single linear decoder layer (i.e., the last layer of ParameterNet) at local device in order to decode $\boldsymbol{\Theta}(t)$ from the sender into a continuous differentiable spatial field. It is important to note that the last layer of ParameterNet is a very wide linear layer, with the width on the order of the total number of weights and biases of ShapeNet.

As an illustrative example, we use a temporally evolving (20 snapshots) spatial (128^3) velocity field of homogeneous isotropic turbulence (HIT) from Johns Hopkins University. Details of data preparation are given in appendix B.4. It should be highlighted that distinct from CAE or SVD, the model complexity of NIF is not directly related to the resolution that the data is stored but rather the *intrinsic* spatial complexity of the data itself. In this example, as for network architecture, ShapeNet has 4 ResNet-like blocks as hidden layers with width as 200 while ParameterNet has 1 ResNet-like block with width as 50 followed by a linear layer with $r = 3$ width and a linear layer that maps the r-dimensional vector to all the weights and biases of ShapeNet. The total number of trainable parameters is 1,297,365, which is only inside ParameterNet. For training, we use an Adam optimizer with a learning rate of 1e-5 and batch size of 1600.

First, we test our model by comparing the first component of the ground true velocity field versus the reconstruction from NIF. As displayed in fig. 11, NIF reconstructs the ground true velocity even with small-scale structures very well. Since most modern studies on turbulence are descriptive from a statistical viewpoint, it is important to verify that the PDF is well preserved after compression. As shown in fig. 12, the PDFs of various quantities are approximately well preserved. For a more stringent test, we verify the model performance by visually comparing the iso-contour of Q-criterion and vorticity magnitude. As displayed in fig. 13, most of the high order quantity is well preserved by the model with only small visual difference. Lastly, it is important to highlight that the original dataset require a storage of an array with size $128^3 \times 20 = 41,943,040 \approx$ while the total number of parameters need to be trained is $1,297,365$ which is 3% of the former. Further, such compression ratio can be improved with more recent neural network pruning techniques (Han et al., 2015) and more inputs to ParameterNet. We leave this exciting topic for future study.

3.4 Efficient spatial query of 3D homogeneous isotropic dataset

When we are querying large scientific datasets, typically from PDE solvers, we perform many more queries in space than queries in time or parameter space. For example, spatial statistics in the homogeneous direction are often collected in the analysis of turbulent physics. Visualization of vortex indicators requires intensive spatial query, e.g., spatial differentiation. The primary reason is that the spatial degree of freedom is typically much larger than either the temporal or parametric degrees of freedom.

Since the structure of NIF isolates the spatial complexity independently from any other factors, we can efficiently get the spatial field data without unnecessary *repeated* computations related to other factors such as time or system parameters. On the contrary, it

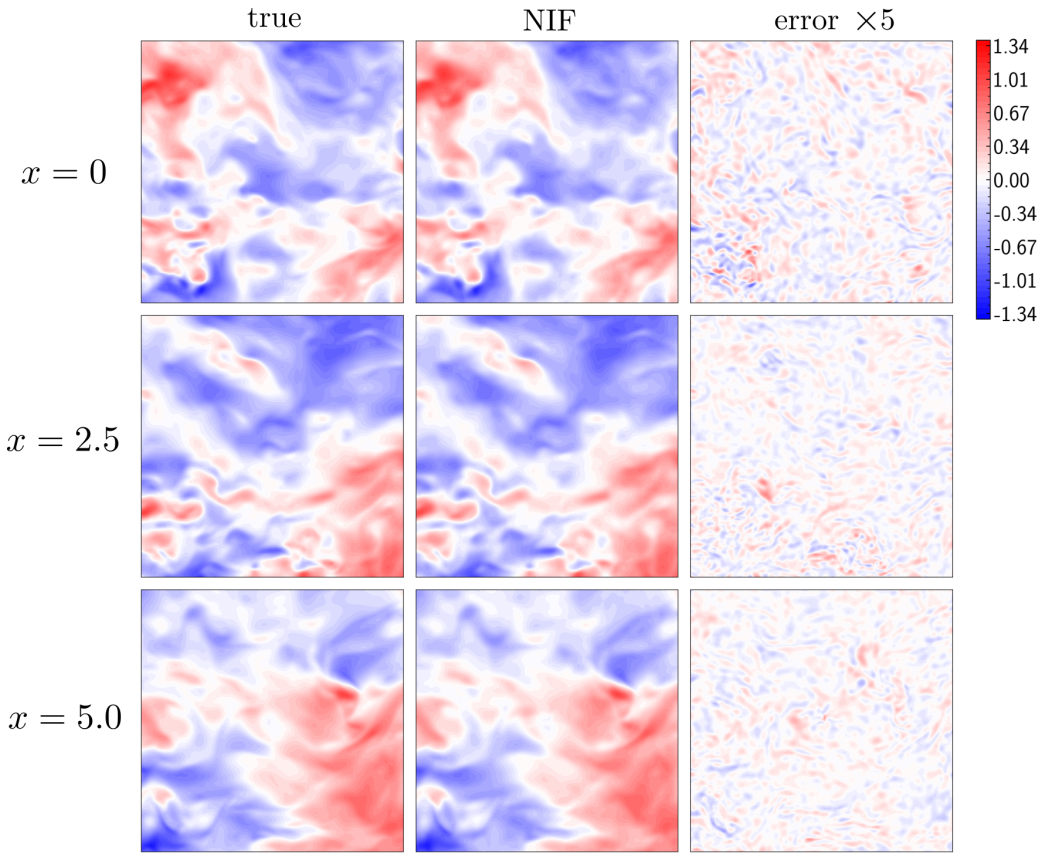

Figure 11: Evaluation of compressed reconstruction from NIF on the U velocity field of the three dimensional homogeneous isotropic turbulence at three difference slices ($x_{\max} = 9.9$) and $t = 0$ from JHU dataset (Li et al., 2008). Left: ground true sliced U field. Middle: reconstruction from NIF. Right: amplified error between ground true and reconstruction from NIF.

could be a waste of resources if one uses a single feedforward neural network with multiple SIREN layers (Sitzmann et al., 2020) that takes *all* information as input with the output still being the field of interests (here we refer it simply as "SIREN"). It is because such a single network will need to mix and learn all of the complexities, e.g., multi-scale, chaos and bifurcations. While the number of spatial query is typically on the order of 10,000 in 2D and 1 million in 3D, one has to repeatedly perform temporal and/or parametric query for different spatial points if they adopt a single network with *all* information as input for spatial query intensive tasks, e.g., learning representation of video (Sitzmann et al., 2020) or temporally evolving volumetric field (Lu et al., 2021c). Therefore, this can lead to a potentially larger network with longer inference time under the same level of accuracy.

Since once the output of ParameterNet is given, the spatial inference can be performed with only run inference on the ShapeNet. We use the same HIT data with 128^3 resolution (see section 3.3). Our model setup is the same as before except that the width of the

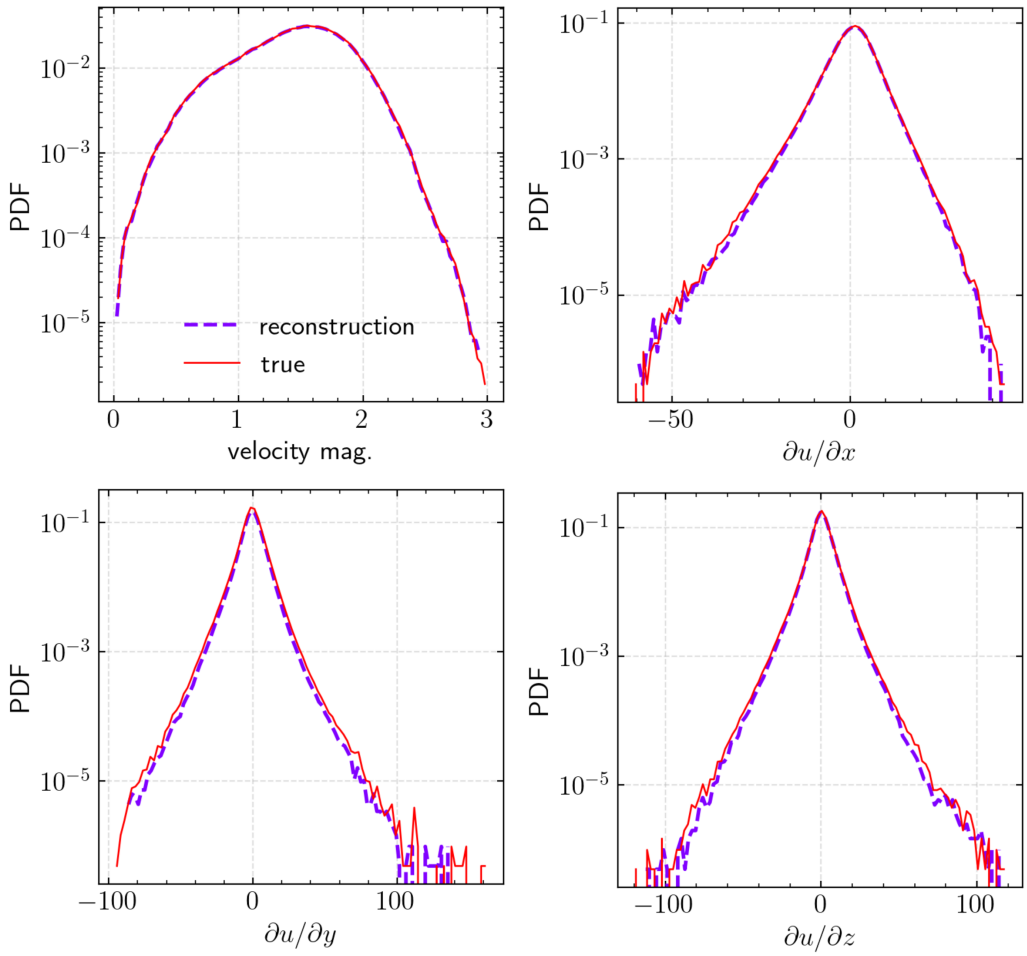

Figure 12: Evaluation of compressed reconstruction from NIF on the PDF of velocity magnitude, $\partial u/\partial x$, $\partial u/\partial y$, $\partial u/\partial z$ of the 3D HIT from JHU dataset (Li et al., 2008) at the first time instance.

ShapeNet and that of the SIREN change from 36, 52, 75, 105 to 150 and the width of the ParameterNet increases from 3 to 10. As shown in the top left of fig. 14, NIF uses a *smaller* network for spatial evaluation compared to SIREN counterpart under the same level of reconstruction error. This is because SIREN takes t, x, y, z as input so capacity of the network is also spent on learning temporal variation.

To further compare the performance, we measure CPU runtime and memory consumption of the spatial query part in the code with the standard Python package `time` and `Fil-memory-profiler` (Turner-Trauring, 2019). We denote the spatial query of three velocity components as a *forward* computation while $\partial u/\partial x$ is denoted as a *backward* computation. To make it a fair comparison, for NIF we take the inference time on ParameterNet (although it contributes less than 1% to the total computational time here) into consideration as well. Note that in the top right and bottom left subfigures of fig. 14, there is not

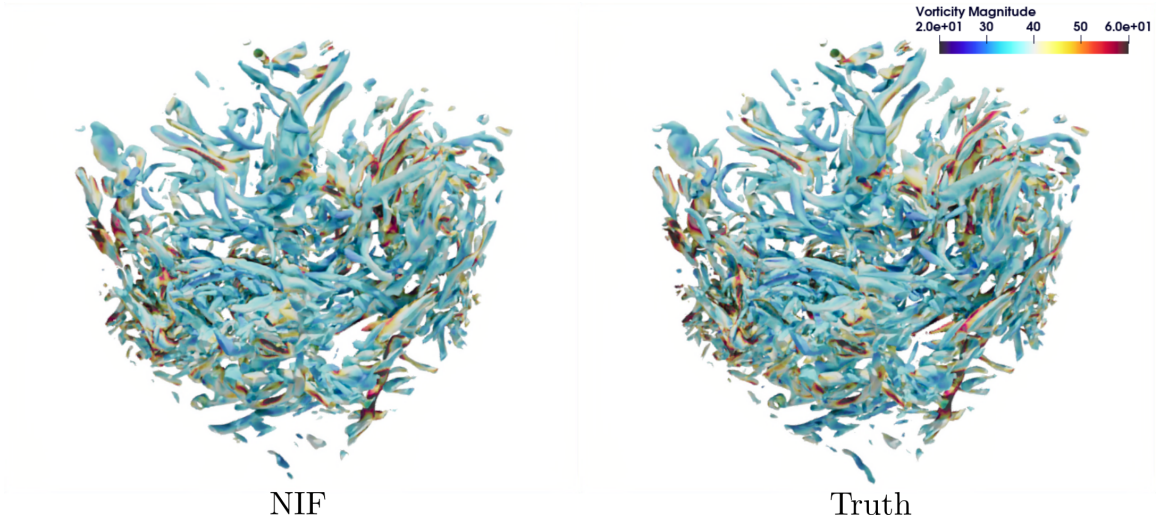

Figure 13: Comparison of compressed reconstruction of 3D HIT (left) and ground true at first snapshot (right). Flowfield is visualized with a contour of Q-criterion colored with vorticity magnitude.

much difference in terms of the scaling between NIF and SIREN, which is simply the similarity between the ShapeNet and the SIREN except the first layer of SIREN takes t, x, y, z as inputs while that of NIF takes x, y, z. We also note that the backward computation requires nearly twice the computational cost as that of forward computation. However, with the same accuracy NIF requires a smaller network than SIREN. Hence, given the same reconstruction error, the width required for NIF and SIREN can be determined. The bottom right of fig. 14 indicates that NIF leads to 27% less neural network width, 30% less CPU time in forward and backward passes and 26%/27% less memory consumption in forward/backward computations for the task of querying homogeneous isotropic turbulence data.

Finally, we compare NIF against popular frameworks *under the same computational complexity for spatial query*, i.e., the same network width associated with spatial input, for the task of reconstructing a toy 2D video of turbulence. Figure 15 qualitatively shows the comparison of our framework against standard MLPs, Fourier Features Networks (Tancik et al., 2020), and SIREN (Sitzmann et al., 2020) on a toy time-varying 2D dataset containing a slice of 3D homogeneous isotropic turbulence. From table 2, we confirm that NIF performs the best among all of the frameworks especially when the network width is limited (e.g., 36, 75) while comparable to vanilla SIREN when the network width becomes larger.

3.5 Modal analysis on adaptive mesh data: flow over cylinder

Next, we test this NIF-based framework to learn a linear subspace for a classical modal analysis problem in fluid mechanics: vortex shedding behind a cylinder. As shown in fig. 7, the simulation is performed with AMR, which is frequently seen in computational

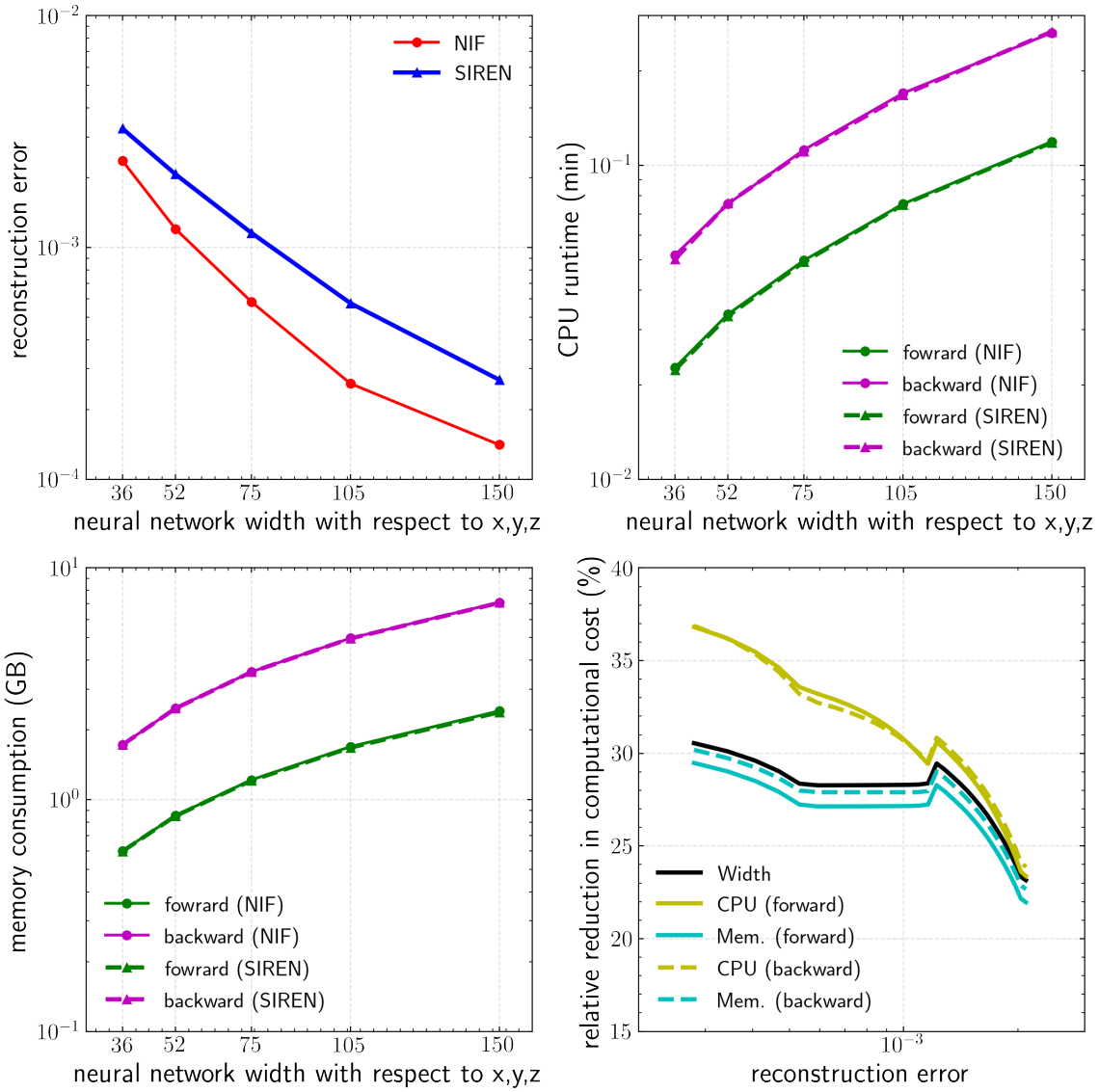

Figure 14: Efficiency comparison between NIF and SIREN in term of CPU runtime and memory consumption when performing spatial query test on half of the first snapshot of the 128^3 homogeneous isotropic turbulence. Top left: variation of reconstruction error with network size; Top right: variation of CPU runtime with network size; Bottom left: variation of memory consumption with network size. Bottom right: relative reduction in computational cost of NIF compared with that of SIREN. Note that the reconstruction error is computed based on the mean-squared error of last batch for every epoch with data shuffling.

Table 2: Normalized error between prediction and ground truth of the 2D time-varying $x-$velocity from a turbulent flow. Frobenius norm of the ground truth is the normalization factor. The configurations for the same row share the same network width, which approximately determines the computational complexity at the inference stage. We found Fourier NN requires a non-trivial tuning for the frequency σ and it doesn't outperform SIREN and NIF. Our framework NIF performs the best for small to middle-range network width while comparable to SIREN when network width reaches 150.

Network width	**MLP (tanh)**	**MLP (relu)**	**Fourier NN** ($\sigma = 1$)	**Fourier NN** ($\sigma = 10$)	**Fourier NN** ($\sigma = 100$)	**SIREN**	**NIF** (Ours)
36	0.357	0.196	0.391	0.265	0.263	0.121	**0.071**
75	0.290	0.172	0.361	0.260	0.241	0.040	**0.022**
150	0.236	0.144	0.353	0.285	0.248	**0.0116**	0.0131

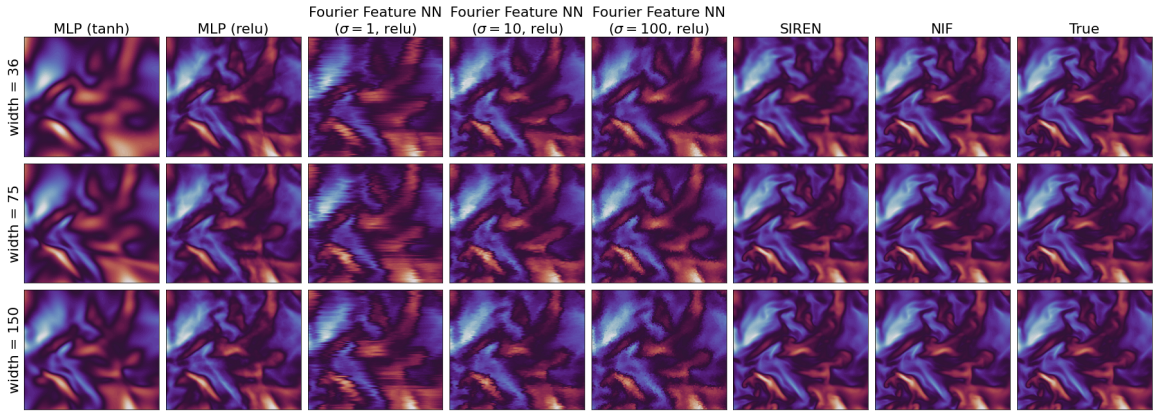

Figure 15: Comparison between NIF and other popular frameworks for reconstructing the x-velocity of a certain slice in 3D homogeneous isotropic turbulence. Each column corresponds to the reconstruction result for a certain model except last column which is the ground truth. Each row corresponds to the same network width associated with spatial input.

fluid dynamics for efficient simulation and data-storage for a wide range of flows containing moving sharp gradients (e.g., unsteady shocks/flames/vortices). Here we first collect 100 snapshots of a spatial field consisting of two velocity component u and v, together with x, y coordinate in each cell and time t. Then, we follow eq. (6) to extract a rank-10 linear subspace from the spatio-temporal field. NIF with SIREN is employed in both ParameterNet and ShapeNet. Details of the data preparation and the reconstruction performance are given in appendix B.3. Given the learned latent 10-dimensional time series $\zeta_1 = a_{\text{MLP},1}(t;\mathbf{\Theta}), \ldots, \zeta_{10} = a_{\text{MLP},10}(t;\mathbf{\Theta})$ as shown in fig. 7, we perform a standard dynamic

mode decomposition (DMD) (Schmid, 2022) to extract isolated spatio-temporal modes with distinct frequencies shown. The DMD mode shapes in fig. 7 agree qualitatively well with other existing studies (Pan et al., 2021; Chen et al., 2012). Note that the latent representation $a_{\text{MLP},i}$ contains time t and spatial functions $\boldsymbol{\phi}_{\text{MLP},i}$ contains $\mathbf{x}$ as input arguments. Thus, at the postprocessing stage, one can easily evaluate these functions above at any time t and/or spatial location $\mathbf{x}$ for any resolution one desires.

3.6 Data-driven sparse reconstruction of sea surface temperature

As shown in fig. 8, we apply the above NIF-based framework to reconstruct and predict sea surface temperature data (Reynolds et al., 2002) given sparse localized measurements. In order to compare with the state-of-the-art, we use a similar setup from Manohar et al. (2018). We take snapshots from 1990 to 2006 as training data and that of the next 15 years, until 2021, as testing data. As mentioned in the previous subsection, we first use POD-QDEIM on the training data to find the best-p sensor locations on the sea. Besides, as shown in the top left of fig. 16, we empirically find POD-QDEIM performs the best at surprisingly low 5 sensors. In order to perform an exhaustive evaluation on NIF-based framework, we vary p from 5 to 600. Due to the appearance of multi-scale structure in sea surface temperature, we use NIF with SIREN in section 2.2. For $p = 5$ to $p = 60$, we take 2 ResNet-like blocks in ShapeNet with width 60 and 2 blocks in ParameterNet with width 60. For $p > 60$, we take 2 blocks in ShapeNet still with width 60 and 2 blocks in ParameterNet with width p. For all cases, we fix $n_p = p$ in analogous to the equality between rank and number of sensors in POD-QDEIM. We use Adam optimizer for mini-batch training with learning rate as 1e-5 and batch size as 7000 for 150 epochs. Details of data preparation are in appendix B.5.

To further evaluate our framework, we make a side-by-side comparison with the state-of-the-art POD-QDIEM in both reconstruction and prediction of sea surface temperature using the same information from the best-p sparse sensor locations from POD-QDEIM. As displayed in the top right of fig. 16, the space-time mean-squared error on training data of both NIF-based framework and POD-QDEIM decrease as the number of sensors increase while that of our framework decays much more quickly than that of POD-QDEIM. The approximated decay rate is shown in the bottom left of fig. 16. We find that our NIF-based framework shows a consistent decay rate -0.74 as the number of sensors p increases. On the other hand, POD-QDEIM struggle to decrease training error only after $p > 50$ with a similar decay rate of -0.73 with a faster rate of -2.05 after $p > 300$. Also, it is interesting to note that as more and more sensors are used, the POD-QDEIM generalization is worse and worse while our framework in general performs better and better. As shown in the bottom right of fig. 16, given the number of sensors p considered in this work, our NIF-based model surpass the best POD-QDEIM model after using 200 sensors, which corresponds to using more than 0.45% of all possible sensor locations on the sea. Finally, as mentioned before, the most generalizable model of POD-QDEIM is using 5 sensors which results in a space-time M.S.E as 0.71 (additional parameter sweeps are shown in the top left of fig. 16). While the model of our NIF-based framework with smallest testing error takes 600 sensors and results in a space-time M.S.E as 0.46, which is 34% smaller than the best model of POD-QDEIM.

Apart from comparing space-time mean squared error, we also compute a standard deviation of spatially mean squared error along time axis as an indicator for robustness of model performance (see error bars in fig. 16). Note that the y axis of the top right figure in fig. 16 is in log scale. Therefore, for the range of number of sensors considered, training error bar of our framework is significantly smaller than that of POD-QDEIM, which indicates our framework can faithfully reconstruct the training data with higher confidence. This is particularly important for hyper-reduction in projection-based ROMs (Carlberg et al., 2011). The testing error bar of our framework is also smaller than that of POD-QDEIM, which means that our framework has higher robustness in predicting unseen data as well. Additional discussions are in appendix C.3.

4. Conclusions

High-dimensional spatial complexity is a major bottleneck of computational and storage costs in many physical science and engineering fields where the physics relies on a set of complex partial differential equations (PDEs). Existing frameworks, such as SVD and CAE, suffer from various challenges arising from complex data structures in real world scientific computing. In this work, a mesh-agnostic representation for parametric spatial-temporal datasets, called *neural implicit flow* (NIF), is proposed. The key feature of NIF is its ability to separate spatial complexity from other factors such as temporal and parametric complexity, which naturally follows the philosophy of manifold-based learning of PDEs. Compared to existing SVD and CAE frameworks, of which the performance is either restricted by unnecessary projection, linearity, intractable memory scaling for large-scale 3D dataset, or inefficiency for adaptive meshes, NIF enables scalable mesh-agnostic nonlinear dimensionality reduction with improved performance. As a comparison, table 3 shows a summary of the capabilities and challenges of SVD, CAE, and NIF. Specifically, we demonstrate the advantages of NIF in terms of four derived *mesh-agnostic* frameworks: data-fit surrogate modeling for parametric PDEs, compressed representation of multi-scale spatial-temporal data, learning linear representations, and nonlinear sparse sensing. To the best of our knowledge, nonlinear dimensionality reduction of 3D turbulence with over 2 million cells and complex multi-scale dynamics with an AMR solver is achieved for the first time.

4.1 Disadvantages

Intrinsic data complexity: For spatially complex data, one requires a correspondingly large ShapeNet to accommodate the spatial structures. Larger networks require longer time to converge. As a result, one has to manually decide the size of ShapeNet and ParameterNet for specific problems, whereas SVD and CAE are much easier to configure.

Long training time: Unlike SVD and CAE where best practices are well established (Maulik and Mengaldo, 2021; He et al., 2019), training a hypernetwork of a deep neural network still requires some trial and error. A typical SVD runtime takes a few seconds to minutes. Training CAE usually takes 0.5hrs to arrive at a decent accuracy. While NIF usually takes above an hour to days depending on the data complexity, which in turns affects the size of model. First, because the input of NIF is pointwise, the total number of training data can

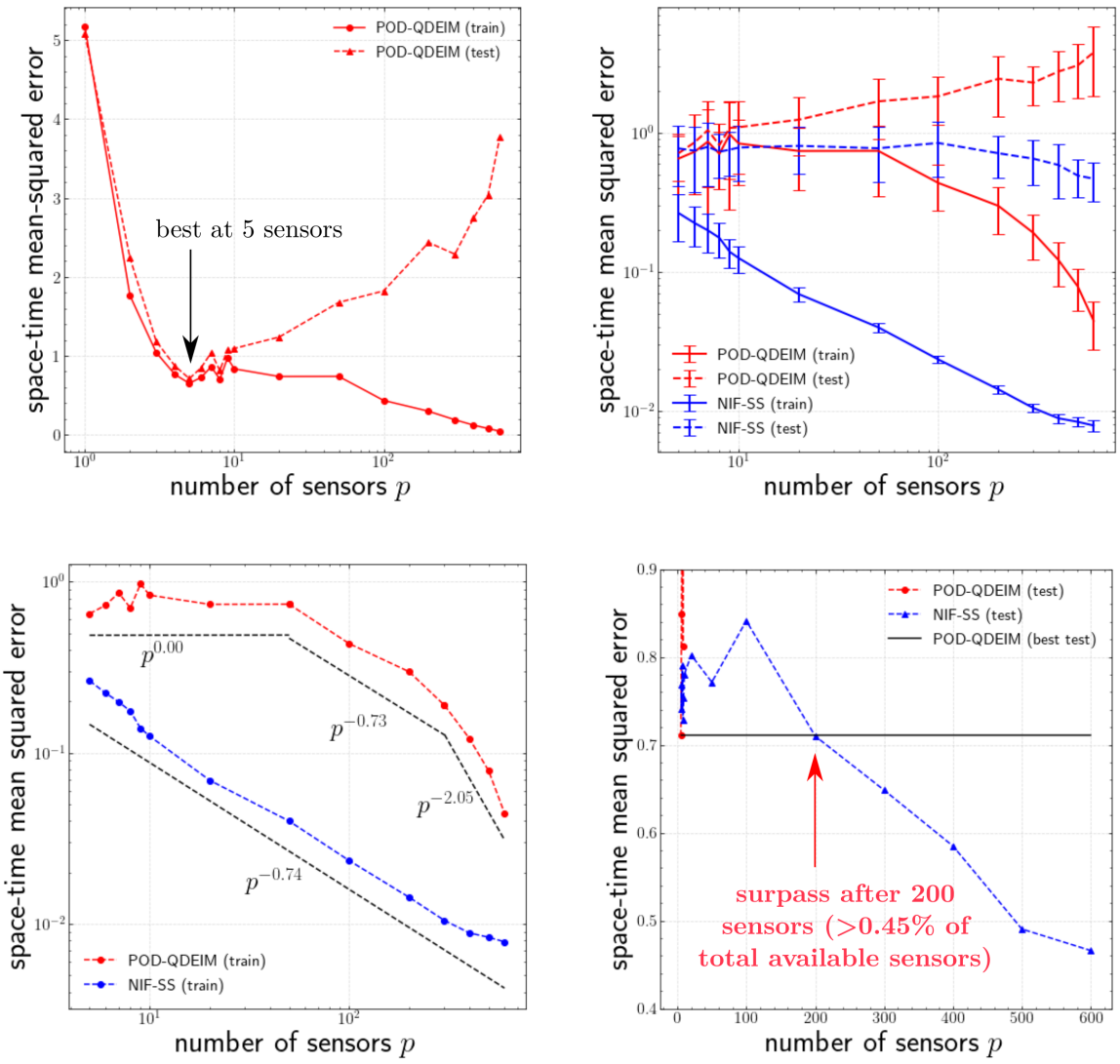

Figure 16: Comparison between NIF-based sparse sensing and POD-QDEIM. **Top left:** training and testing error of POD-QDEIM with a complete parameter sweep for the number of sensors from 1 to 600. **Top right:** mean-squared error with standard deviation for varying number of sensors from 5 to 600. Error bar represents plus and minus one temporally standard deviation of spatially mean square error. **Bottom left:** approximated decay rate of training error with respect to increasing number of sensors p (POD-QDEIM versus our NIF-based framework). **Bottom right:** comparison of testing error among NIF-SS, POD-QDEIM and best POD-QDEIM model.

be huge for 3D datasets. Second, in order to connect the two networks, there are expensive

tensor operations between 1D and a 2D tensor (output reshaped from ParameterNet). Note that both of the tensors are *batch dependent* in NIF, which can decrease cache performance. For example in the 3D turbulence case, it takes up to 1.3 hours to process one single epoch and almost 5 days on a 16GB GPU to obtain a good accuracy.

Memory intensive: The use of hypernetwork leads to more complex network topology than the standard feedforward neural net with a comparable model parameter size. As a result, it creates larger memory footprints, which unfortunately limits the maximal batch size and leads to longer training time.

Lack of invariance: Despite of some promising results (Wang et al., 2022), it is generally more challenging to embed invariance into coordinate-input neural networks than graph-based approaches (Li et al., 2020a). Such lack of invariance may worsen the generalization of data-fit regression especially when the amount of training data is small.

4.2 Advantages

Intrinsic data complexity: On the flip side, model complexity does not scale with the "superficial" complexity of the data, e.g., the number of mesh points. Finer mesh points only lead to more training data while model complexity can keep the same. Meanwhile, mesh-based models (e.g., CAE) still suffer from growing model complexity as the number of mesh points increases.

Heterogeneous data sources: Since it is mesh-agnostic in nature, it becomes very convenient to fuse information from different data sources. For example, PIV data are mostly on a perfect uniform Cartesian mesh, whereas CFD data are commonly on a well designed body-fitted mesh. Sometimes different types of CFD solver would use different mesh, e.g., multi-level Cartesian mesh for LBM/IBM (Taira and Colonius, 2007).

Well-established manifold-based ROM: Thanks to the decoupling of spatial complexity, we have a direct analogy of SVD but with a mesh-agnositic, nonlinear and scalable version for 3D datasets. It is straightforward to extend the established ROM frameworks (Carlberg et al., 2011; Bruna et al., 2022), modal analysis (Schmid, 2022; Taira et al., 2020; Rowley et al., 2009) to more realistic and mesh-complex datasets.

Efficient spatial query: Postprocessing of PDE data, e.g., turbulence (Li et al., 2008), often involves intensive spatial query than temporal or other parametric query. Our design of NIF leads to a compact network for spatial complexity, which improves the efficiency for massive spatial query on either the point value or the spatial derivative.

5. Prospects

NIF has the potential to extend existing data-driven modeling paradigms based on SVD and CAE to the real world of scientific computing, where raw spatial-temporal data can be three dimensional, large-scale, and stored on arbitrary adaptive meshes. High-fidelity large-scale data from modern scientific computing codes (Almgren et al., 2010; Nonaka et al., 2012) can be reduced to effective low dimensional spaces, where existing modal analysis (Taira

Table 3: Capabilities and challenges of representation learning: SVD, CAE, and Neural Implicit Flow.

Property/Model	**SVD**	**CAE**	**Neural Implicit Flow**
Resolution	strong Convergence to discrete data	weak Requires uniform mesh	strong Continuous field
Variable geometry	strong	weak	strong
Scalablity	strong Efficient randomized SVD available	fair/weak Affordable in 2D Resolution restricted in 3D	strong Point-wise mini-batch training; Number of parameters required only scale with intrinsic complexity
Parametric/temporal variation of domain discretization	weak Requires the same mesh throughout	weak Requires the same uniform mesh throughout	strong arbitrary mesh
Training easiness	strong	fair/strong	fair
Interpretability of representation	strong	weak	fair last-layer parameterization learns linear subspace
Expressiveness	weak Linear, not ideal for advection dominated flows	fair Nonlinear but cannot capture multi-scale efficiently	strong Nonlinear with multi-scale capability

et al., 2020; Towne et al., 2018; McKeon and Sharma, 2010) data-driven flow control (Duriez et al., 2017) and surrogate-based modeling/optimization (Koziel and Leifsson, 2013) can be performed seamlessly on the arbitrary adaptive mesh. However, as is typical for nonlinear dimensionality reduction methods (e.g., CAE), the latent representation is more difficult to interpret than its linear counter part (e.g., SVD), and each time the latent representation can be different. This raises new questions on how to guide the latent representation to be interpretable for experts. Another exciting direction is to construct projection-based ROMs in such a latent space by minimizing the residual of *known* governing equations. This has been demonstrated very recently on the material point method (Chen et al., 2021) and for high-dimensional PDE with active learning (Bruna et al., 2022). Besides dimensionality reduction, NIF can be used to design efficient "decoders" for transferring large spatial scientific data. It essentially trades reduction in storage/transfer and ease in data fusion of heterogeneous meshes with off-line training a NIF model and additional function calls on the ShapeNet.

Acknowledgments

The authors thank Kevin Carlberg, Karthik Duraisamy, Zongyi Li, and Lu Lu for valuable discussions. The authors acknowledge funding support from the Air Force Office of Scientific Research (AFOSR FA9550-19-1-0386). The authors acknowledge funding from the National Science Foundation AI Institute in Dynamic Systems grant number 2112085. We also appreciate the generous support from Prof. Karthik Duraisamy on using the following computing resources, which were provided by the NSF via the grant "MRI: Acquisition of ConFlux, A Novel Platform for Data-Driven Computational Physics". This work also used the Extreme Science and Engineering Discovery Environment (XSEDE) (Towns et al., 2014), which is supported by National Science Foundation grant number ACI-1548562. Specifically, it used the Bridges-2 system, which is supported by NSF award number ACI-1928147, at the Pittsburgh Supercomputing Center (PSC).

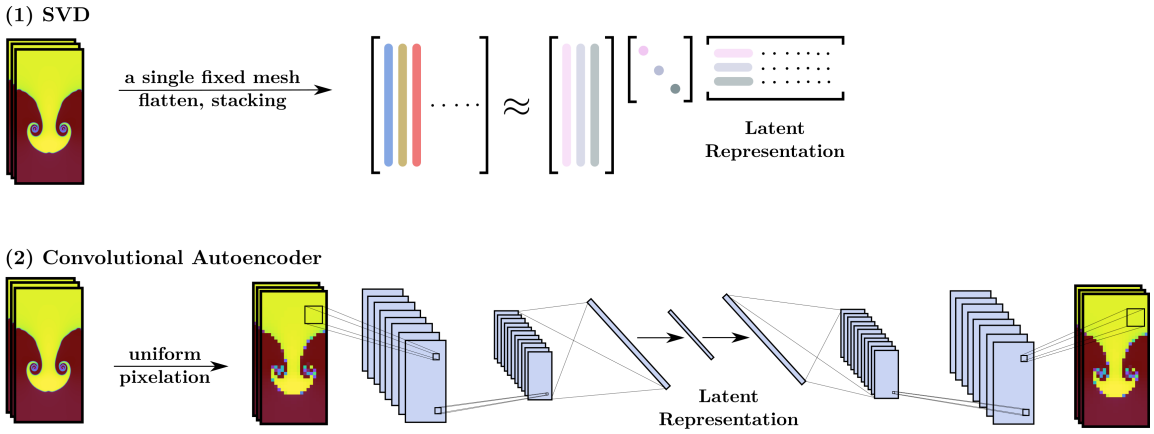

Figure 17: SVD and CAE for dimensionality reduction of spatio-temporal field from PDE. (1) SVD relies on the assumption that spatio-temporal field is sampled from a single fixed mesh for all parameters across all time. Here the latent representation is formed by the right singular vectors. (2) CAE treats spatio-temporal fields as images by pre-processing with uniform pixelation. Latent space is the flattened vector after several convolution and pooling operations.

Appendix A. POD and CAE

As shown in fig. 17, state-of-the-art methods for dimensionality reduction of parametric spatial temporal data rely on SVD and CAE. Dimensionality via SVD requires the data on a single fixed mesh. First, it flattens the spatial data into a long column vector. Second, these column vectors are stacked over time. Finally, SVD is performed on such stacked matrix and the right singular vectors are the latent representation. CAE treated the flowfield as image. First, one performs a uniform pixelation. Second, one feed the processed images into convolutional autoencoders. The corresponding latent representation is formed by the bottleneck layer in fig. 17.

Appendix B. Data preparation

B.1 1D parametric K-S

Given $x \in [0, 2\pi]$, $t \in [0, 100]$, we use ETD-RK4 (Kassam and Trefethen, 2005) method to solve 1D Kuramoto-Sivashinsky equation in eq. (9), with periodic boundary condition $u(0, t) = u(2\pi, 0)$ and a fixed initial condition $u(x, 0) = \sin(x)$. Spatial discretization is performed with 1024 points. Time integration $dt = 10^{-3}$. The raw dataset has space-time resolution as 1024×10000. We subsample 4 times in space and 1000 times in time, i.e., 256×100. Note that K-S has a rich dynamics when parameter changes (Papageorgiou and Smyrlis, 1991). In order to make the problem well-posed, we choose a regime $\mu \in [0.2, 0.28]$ where the system is not chaotic. Parametric variation of the solution in $x - t$ is displayed in fig. 18. Training data is prepared with 20 uniform sampling with $[0.2, 0.28]$, ending up

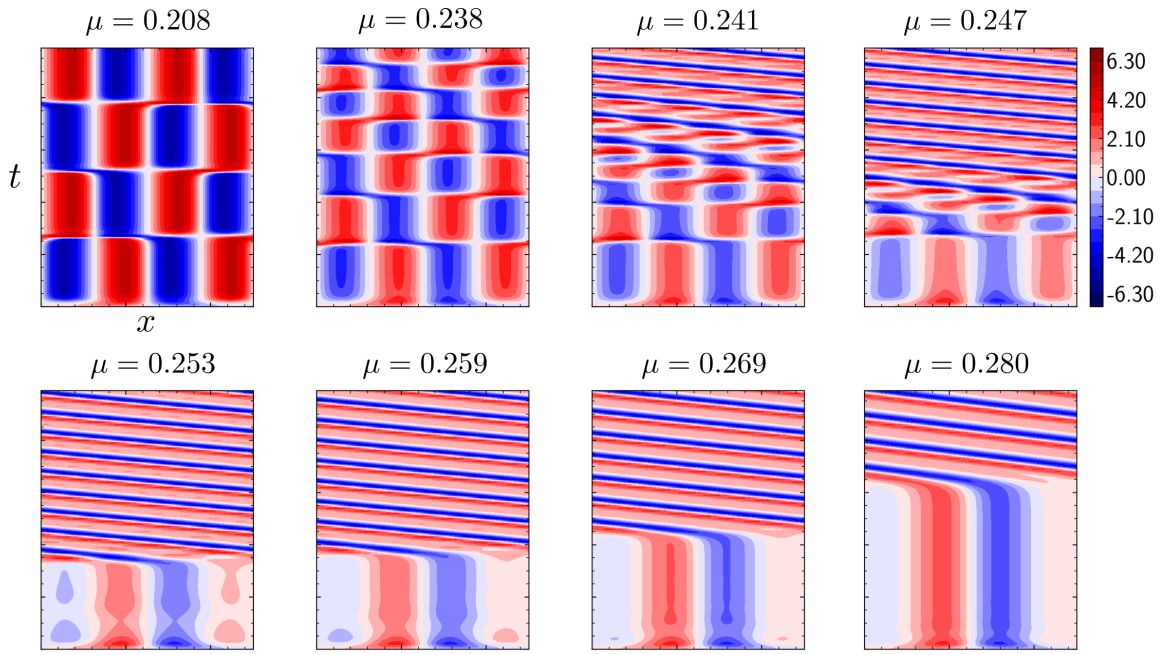

Figure 18: Parametric variation of spatial temporal field generated from Kuramoto Sivashinsky system in $x - t$ at different μ.

with $20 \times 256 \times 100 = 0.512 \times 10^6$ data points. Testing data is prepared with 40 uniform samples within $[0.2, 0.28]$, which leads to $40 \times 256 \times 100 = 1.024 \times 10^6$ data points. Still, each data point has 4 components μ, t, x, u. Each component is normalized with zero mean and unit standard deviation.

B.2 2D Rayleigh-Taylor

Rayleigh-Taylor instability happens when a heavy fluid floats above a light fluid in a gravitational field. The interface between heavy and light fluid is unstable and "mushroom"-like multi-scale structure will grow as small-scale structures grows much faster than large-scale structures. In order to efficiently simulate the system, adaptive mesh refinement is necessary. The goal of dimensionality reduction of parametric fluid system is to reduce the spatial complexity so that one may perform modeling, design or control in the reduced coordinate efficiently. However, without given the range of system parameters in the data, it is often difficult to make statement on the true dimensionality of a flow. In order to simplify our discussion, we choose only to simulate a single realization of R-T with fixed system parameter where the dimensionality can be reduced to one.

Using CASTRO, we consider one level of AMR with an initial Cartesian grid of 256 points along vertical axis and 128 points along horizontal axis during the simulation, which efficiently captures the evolution of vortex generation and advection of density fronts. AMR refinement ratio is 2 in each direction. We regrid at every 2 steps. The simulation domain is

a rectangular with 0.5 long in x-axis and 1.0 in y-axis. Top and bottom boundary condition is slip wall while left and right are periodic. Time span of the simulation is from 0 to 3.5. The density for heavy fluid is 2 while that for the light fluid is 1. They are separated by a horizontally trigonometric and vertically tanh interface, which is a perturbation to the unstable system. The exact expression follows the equation (46-47) in the original CASTRO paper (Almgren et al., 2010).

We save the simulation data at every 10 steps. Training data starts from 83rd to 249th snapshot (corresponding to time t from 1.004 to 2.879) with skipping on the even snapshot: 83, 85, ..., 247, 249. Testing data starts from 86th to 248th snapshot with an incremental of six: 86, 92, ..., 242, 248. Training and testing data are not overlapping but they have no distribution shift. In total, training data has 84 snapshots and testing data as 28 snapshot. The number of adaptive mesh points ranges from 40,976 to 72,753 across time. For POD and CAE, original data on adaptive mesh is projected onto three different Cartesian grids: 32×64, 64×128, 128×256 using the filter `ResampleToImage` in `ParaView` (Ahrens et al., 2005). Finally, original data is also projected onto a very fine Cartesian mesh 256×512 as "ground truth" in order to make quantitative comparison for predictions from different models.

As for NIF, firstly, note that we use a feedforward neural network encoder with sparse sensor as input. As shown in the left of fig. 19, we collect measurements from 32 equally spaced sensor along the vertical axis in the middle of the flowfield. At test time, input data that feed to the encoder is displayed right of fig. 19, where we can see a clear density front moving towards the bottom. Secondly, since NIF takes pointwise data, each data point is a 35-dimensional row vector. The first 32 are sensor values at t while the rest three are x, y and density at that location $u(x, y)$. Because we need to go through all grid points in the adaptive mesh at time t, there will be as many data points with repeated 32 columns as the total number of points in the adaptive mesh at time t. In fact, we end up with 4,190,928 rows with 35 columns for the training data. Thirdly, since we are using NIF with SIREN in our ShapeNet, we normalize x, y between -1 and 1 while the rest 33 columns are normalized with zero mean and unit standard deviation.

B.3 2D Cylinder Flow with $Re = 123$

We generated the data using *PeleLM* (Nonaka et al., 2012) to solve a incompressible N-S. The cylinder geometry is represented using embedding boundary method. The setup is given in our the Github repo. We set the cylinder centered at the original. The computational domain is x (streamwise) direction from -0.02 to 0.1 while y direction from -0.04 to 0.04. The boundary condition is set as inflow: Dirichlet, outflow Neumann. Side: periodic. We set $U_\infty = 3$ m/s, viscosity $\mu_\infty = 2 \times 10^{-4}$ Pa·s, $\rho_\infty = 1.175$ kg/m^3. The radius of cylinder $r = 0.0035$ m. Thus, Reynolds number is $Re = 2U_\infty r/\mu_\infty \approx 123$. The simulation is performed from $t = 0$ s to $t = 2$ s with sampling $\Delta t = 0.0005$ s. The flow is initialized with uniform streamwise flow U_∞ superimposed with a side velocity in order to break the symmetry so that the flow can quickly fall on to the limit cycle. To remove the initial transient effect, we collect the last 100 snapshots. The AMR is based on vorticity where we enable a finer mesh with half of the grid size if the magnitude of local vorticity monitored exceed 3000. Overall, we collect data (we can easily collect AMR data pointwise using

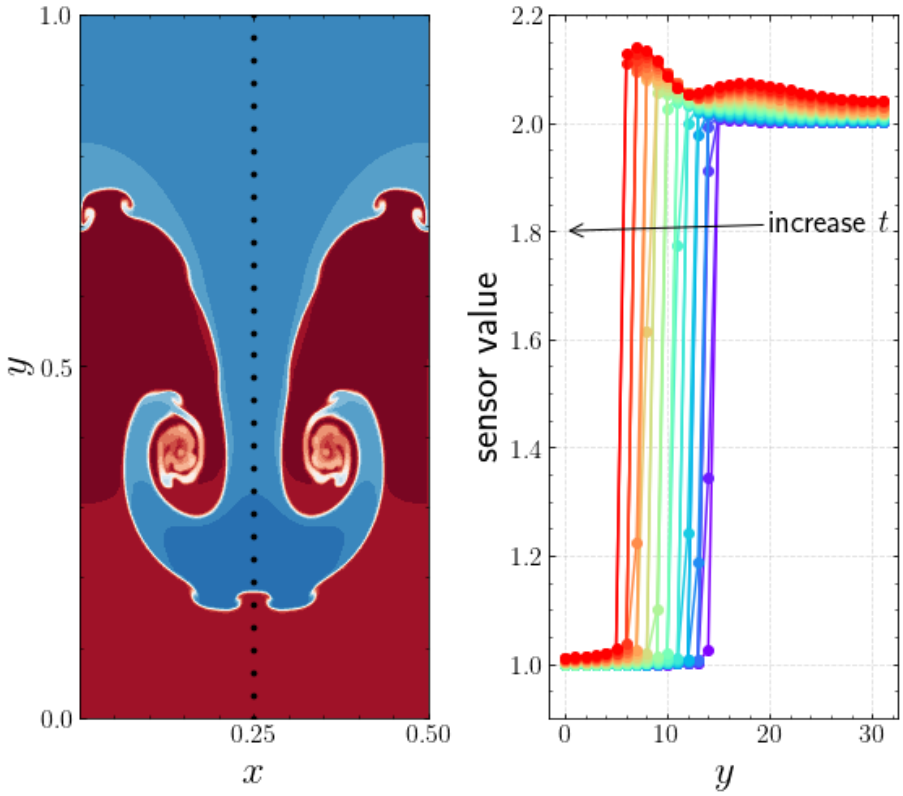

Figure 19: Illustration of sensor measurements for NIF-autoencoder in the R-T instability. Left: 32 sensor locations are distributed evenly on the middle vertical axis. Right: all 32 sensor values for different time t in the testing data.

ParaView) after the system falls on to the limit cycle and sampled the last 100 snapshots. Moreover, in order to remove the effect from the exit boundary and only focus on the region of interests, we only sample cell-centered data within the domain of interests with $x \in [-0.01, 0.04]$ m and $y \in [-0.02, 0.02]$ m as shown in fig. 20. Since we are using NIF with SIREN in appendix D, we normalize the t, x, y into uniform distribution between -1 and 1. For cell area $\mathbf{\Delta x}$, we scale it with a factor of 10^6 since we are using single precision in Tensorflow. For output velocity u, v, we first normalize them into zero mean. Next we normalize u into unit variance and normalize v with the same factor. Finally, we arrange the collected pointwise data into a big matrix, with 966,514 rows and 6 columns,

$$\begin{bmatrix} \widetilde{t}_1 & \widetilde{x}_1 & \widetilde{y}_1 & \widetilde{u}_1 & \widetilde{v}_1 & \widetilde{\mathbf{\Delta x}}_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \widetilde{t}_{966514} & \widetilde{x}_{966514} & \widetilde{y}_{966514} & \widetilde{u}_{966514} & \widetilde{v}_{966514} & \widetilde{\mathbf{\Delta x}}_{966514} \end{bmatrix},$$

where wide tilde means normalized variable.

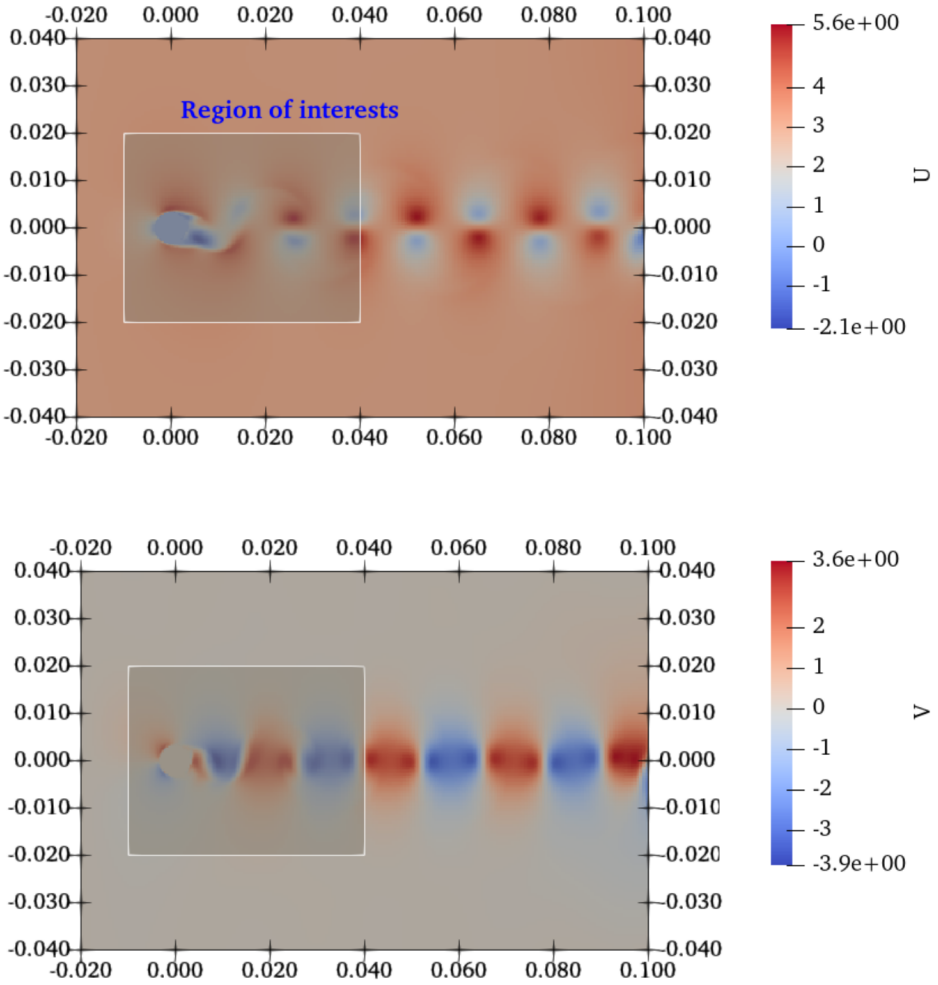

Figure 20: Configuration and mesh of CFD simulation on 2D cylinder flow at $Re \approx 123$ with AMR. Contour of magnitude of vorticity is shown. Domain of interests is marked in light gray with white edges.

B.4 3D Forced Homogeneous Isotropic Turbulence with $Re_\lambda = 433$

We use the *forced isotropic turbulence* dataset from JHU Turbulence dataset (Li et al., 2008) with Taylor-scale Reynolds number Re_λ around 433. The simulation is computed by solving 3D incompressible Navier-Stokes equation with pseudo-spectral method. To sustain the turbulence, energy is injected into the large scales. After the system reaches the equilibrium state, the original dataset consists snapshots collected at every 0.02 nondimensionalized time. Here we collect $1, 4, 7, 10, \ldots,$ 58-th snapshot, in total 20 snapshots. Then we slice a block with 128^3 resolution from the original HIT with the 20 snapshots, which is 1024^3. Since we are using NIF with SIREN in appendix D, we normalize the range of t, x, y so that

the input signal is uniformly distributed between -1 and 1. For target velocity u, v, w, we simply normalize them into zero mean and unit variance.

B.5 Sea Surface Temperature

We obtain the weekly averaged sea surface temperature data since 1990 to present from NOAA website [10]. At the time when we worked on this problem, there is in total 1642 snapshots (1990-2021) with each snapshot corresponds to a weekly averaged temperature field. Each temperature snapshot contains 180 uniformly spaced latitude and 360 uniformly spaced longitude coordinates. It should be noted coordinates correspond to land should be masked [11]. Thus, each masked snapshot contains 44219 points. We use temperature field of the first 832 weeks (16 years) as training data with a total number of 36,790,208 data points. Each data point is a $n_s + 3$-dimensional row vector, where n_s is the number of sensors and 3 corresponds to x, y, T. Locations of n_s sensors are obtained via POD-QDEIM or equivalently data-driven sensor placement in (Manohar et al., 2018). Temperature field in the rest of time are testing data. Still, spatial coordinate x and y is normalized between -1 and 1. Target temperature is normalized into zero mean and unit variance.

Appendix C. Additional discussions

C.1 Data-fit surrogate modeling for 1D Kuramoto-Sivashinsky

Recall that the advantages of NIF have been demonstrated in section 3.1. In order to further compare the model performance, we further compute the RMSE for each parameter μ for four configurations: (1) NIF-Swish, (2) MLP-Swish, (3) NIF-tanh, (4) MLP-tanh. As displayed in fig. 21, NIF with Swish generalizes better than other configurations especially when $\mu > 0.25$. As seen in appendix B.1, such parameter regime corresponds to the traveling waves solutions with a transition time increasing with μ. Meanwhile, NIF-tanh generalizes slightly better tanh MLP-tanh but similar with MLP-Swish. Hence, we only compare NIF-Swish with MLP-Swish in the rest of the paper.

Here we consider further changing the size of training data to verify if NIF-Swish still performs better than MLP-Swish. As shown in fig. 22, we prepare another three set of datasets with 15, 24, and 29 samples of parameters, which correspond to simulations of K-S system at those parameters.Figure 22 shows the same phenomenon as before that NIF (Swish) generalizes better than MLP (Swish) when $\mu > 0.25$.

Finally, we consider verifying the above phenomenon by changing the number of trainable parameters away from the baseline in section 3.1 as described in table 4. Again, we still train for 4 converged runs and compute the average error. Still, Figure 23 shows the same phenomenon that explains the difference between NIF (Swish) and MLP (Swish).

10. https://downloads.psl.noaa.gov/Datasets/noaa.oisst.v2/sst.wkmean.1990-present.nc
11. https://downloads.psl.noaa.gov/Datasets/noaa.oisst.v2/lsmask.nc

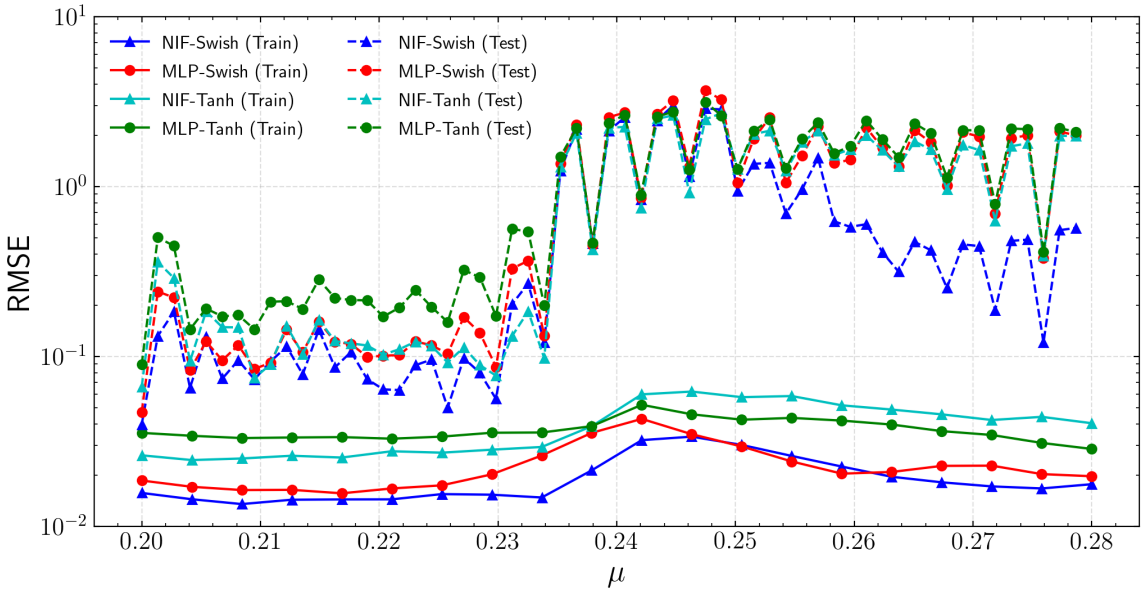

Figure 21: Comparison among four model configurations (NIF-Swish, MLP-Swish, NIF-tanh, MLP-tanh) in terms of RMSE on the dataset of 1D parametric KS system with $0.2 < \mu < 0.28$.

Table 4: Comparison between standard MLP and NIF in section 2.1 for surrogate modeling of 1D parametric PDE. RMSE below is averaged over all parameter μ.

Model	ShapeNet/Vanilla MLP	ParameterNet	Number of training parameters
NIF (Swish)-1	1-30-30-30-1	2-30-30-2-1951	6,935
MLP (Swish)-1	3-58-58-58-1		7,135
NIF (Swish)-2	1-38-38-38-1	2-29-29-2-3079	10,254
MLP (Swish)-2	3-70-70-70-1		10,291
NIF (Swish)-3	1-56-56-56-1	2-30-30-2-6553	20,741
MLP (Swish)-3	3-100-100-100-1		20,701
NIF (Swish)-4	1-60-60-60-1	2-47-47-2-7501	24,996
MLP (Swish)-4	3-110-110-110-1		24,971
NIF (Swish)-5	1-70-70-70-1	2-60-60-2-10151	34,415
MLP (Swish)-5	3-130-130-130-1		34,711

C.2 Comparison of autoencoders for 2D Rayleigh-Taylor instability

C.2.1 Detailed setup of convolutional autoencoder

The encoder first uses three consecutive stride-2 convolutional layer followed by batch normalization and Swish activations with the number of output channel as 16, 32, 64. Output of the above is followed by flatten layer and a linear dense layer with r output units. Then

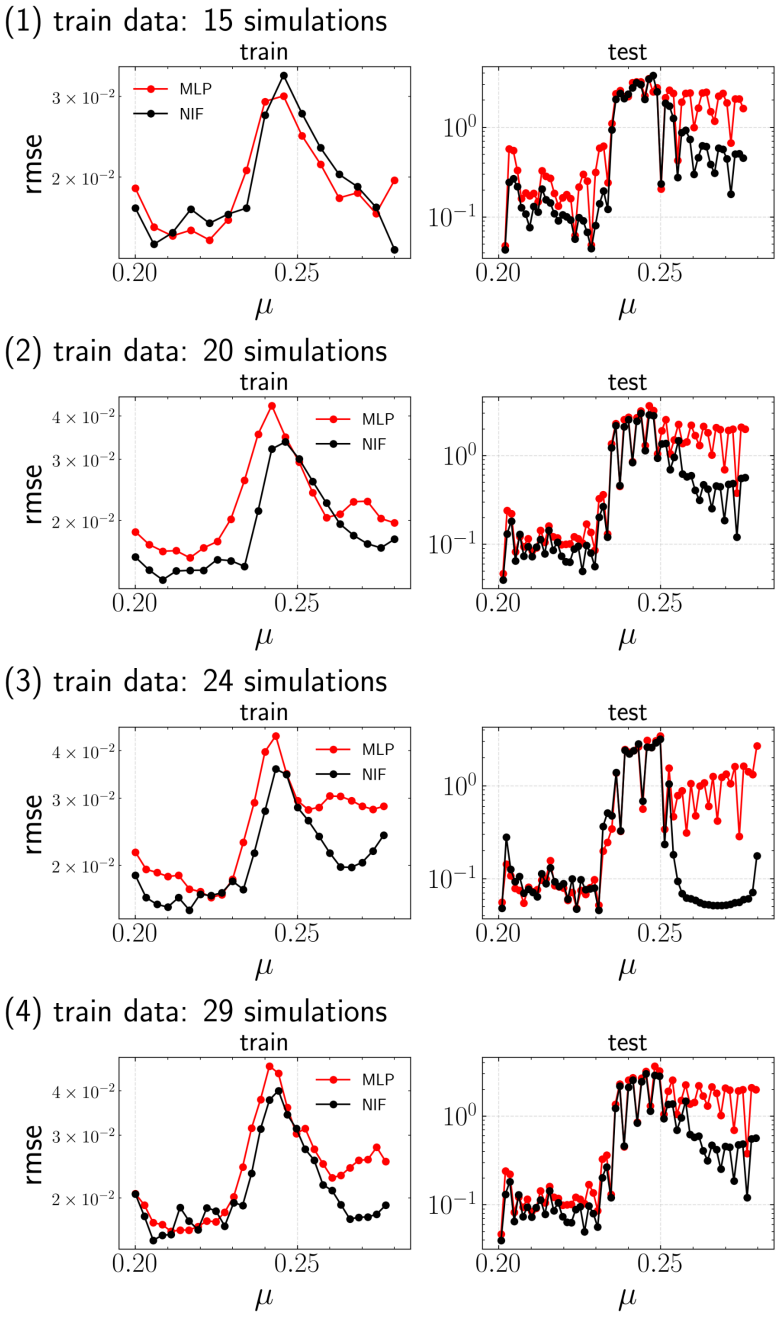

Figure 22: Comparison between NIF-Swish and MLP-Swish in terms of RMSE on the dataset of 1D parametric KS system with $0.2 < \mu < 0.28$. Training data contains different numbers of simulations from 15 to 29.

such units are feed to the decoder, which first contains a linear dense layer followed by a reshape layer and two consecutive dilated transposed convolutional layer followed by batch

Table 5: CAE architecture with input shape 128×256.

Layer (type)	Output Shape	Parameter Count
Convolution 2D	(None, 128, 64, 16)	160
Batch Normalization	(None, 128, 64, 16)	64
Swish Activation	(None, 128, 64, 16)	0
Convolution 2D	(None, 64, 32, 32)	4640
Batch Normalization	(None, 64, 32, 32)	128
Swish Activation	(None, 64, 32, 32)	0
Convolution 2D	(None, 32, 16, 64)	18496
Batch Normalization	(None, 32, 16, 64)	256
Swish Activation	(None, 128, 64, 16)	0
Flatten	(None, 32768)	0
Dense	(None, 8)	262152
Dense	(None, 32768)	294912
Reshape	(None, 32, 16, 64)	0
ConvolutionTranspose 2D	(None, 64, 32, 32)	18464
Batch Normalization	(None, 64, 32, 32)	128
Swish Activation	(None, 64, 32, 32)	0
ConvolutionTranspose 2D	(None, 128, 64, 16)	4624
Batch Normalization	(None, 128, 64, 16)	64
Swish Activation	(None, 128, 64, 16)	0
ConvolutionTranspose 2D	(None, 256, 128, 1)	145

normalization and Swish activation with the number of output channel as 32, 16. Finally, the last layer is a dilated transpose convolutional layer that maps to the same size as input snapshot. For the best results, we do not use any pooling layers throughout.

C.2.2 Evolution of fitting, projection and total error

As mentioned before, we can quantitatively compare the performance of three methods of dimensionality reduction by projecting those outputs on a very fine mesh with 256×512 resolution. Recall that the shape of POD output is 128×256, and the shape of CAE output varyes from 32×64, 64×128, 128×256. Hence, we use nearest-neighbor to obtain the projection onto 256×512 using the `transform.resize` function in `scikit-image` package (Van der Walt et al., 2014). While for NIF, we simply perform spatial query at those coordinates of cell nodes, which are $x_i = 2.5 \times 10^{-7} + (i-1) \times 1.96078333 \times 10^{-3}$, $y_j = 5 \times 10^{-7} + (j-1) \times 1.95694618 \times 10^{-3}$, for $i = 1, \ldots, 256$, $j = 1, \ldots, 512$. This can be generated in Python with `numpy.linspace(2.5e-7, 0.5, 256, endpoint=True)` and `numpy.linspace(5e-7, 1, 512, endpoint=True)`.

Unless highlighted, the following data fields are projected onto 256×512 mesh and become 2D arrays varying with time t. Value of the field at each coordinate (x_i, y_j) and time t can be indexed by $(i, j; t)$.

- u_{true}: ground true data from adaptive mesh,
- $u_{\text{true},32\times64}$: ground true data sampled on the 32×64 Cartesian mesh,
- $u_{\text{true},64\times128}$: ground true data sampled on the 64×128 Cartesian mesh,
- $u_{\text{true},128\times256}$: ground true data sampled on the 128×256 Cartesian mesh,

- $u_{\text{POD},128\times256}$: output prediction from POD on the 128×256 Cartesian mesh,
- $u_{\text{CAE},32\times64}$: output prediction from CAE on the 32×64 Cartesian mesh,
- $u_{\text{CAE},64\times128}$: output prediction from CAE on the 64×128 Cartesian mesh,
- $u_{\text{CAE},128\times256}$: output prediction from CAE on the 128×256 Cartesian mesh,
- u_{NIF} as the output prediction from NIF evaluated on the 256×512 mesh.

Without loss of generality, let's take the CAE with training data on the 32×64 mesh for example. Notice that

$$\underbrace{u_{\text{true}} - u_{\text{CAE},32\times64}}_{\text{total difference}} = \underbrace{u_{\text{true}} - u_{\text{true},32\times64}}_{\text{projection difference}} + \underbrace{u_{\text{true},32\times64} - u_{\text{CAE},32\times64}}_{\text{fitting difference}}. \tag{10}$$

Hence, we define the following three error metrics at each time t:

- *fitting error*: spatially averaged mean square of fitting difference,

$$\varepsilon^{\text{CAE},32\times64}_{\text{fitting}}(t) = \frac{1}{256 \times 512} \sum_{i=1}^{256} \sum_{j=1}^{512} (u_{\text{true},32\times64}(i,j;t) - u_{\text{CAE},32\times64}(i,j;t))^2. \tag{11}$$

- *projection error*: spatially averaged square of project difference,

$$\varepsilon^{\text{CAE},32\times64}_{\text{projection}}(t) = \frac{1}{256 \times 512} \sum_{i=1}^{256} \sum_{j=1}^{512} (u_{\text{true}}(i,j;t) - u_{\text{true},32\times64}(i,j;t))^2. \tag{12}$$

- *total error*: spatially averaged square of total difference

$$\varepsilon^{\text{CAE},32\times64}_{\text{total}}(t) = \frac{1}{256 \times 512} \sum_{i=1}^{256} \sum_{j=1}^{512} (u_{\text{true}}(i,j;t) - u_{\text{CAE},32\times64}(i,j;t))^2. \tag{13}$$

We can define the same metrics for other models. Evolution of the above three error metrics for all the models on training and testing time steps are displayed in the first row of fig. 24. Fitting error (red) contributes the most in POD while projection error is more than two orders of magnitude smaller. This implies the lack of expressiveness in POD leads to its inferior performance. In contrast, projection error (green) contributes most in CAE while fitting error remains unchanged with varying resolution of projection from 32×64 to 128×256. This indicates that CAE is mainly restricted by the projection error introduced during preprocessing in this example. Meanwhile, NIF learns the latent representation directly from the raw data on adaptive mesh. Therefore, fitting error is the same as total error. Moreover, we observe that projection error grows *near-exponentially* in time. This is because of the nature of R-T instability that energy from large scale structures transfer to small-scale as time goes. Such small-scale vortex generates even further smaller vortex. Eventually, the Cartesian grid becomes unable to resolve the flow, which ends up with a significant amount of projection error. As shown in the rest of fig. 24, such phenomenon persists even changing rank r. The fact that the dimensionality of this R-T data is one is consistent with fig. 24 from which only POD methods improves when r is increased.

C.3 Sparse reconstruction on sea surface temperature

Comparison on temporal variation of spatially mean-squared error between our framework and POD-QDEIM (Manohar et al., 2018) with different number of sensors p is shown in fig. 25. We can visually confirm the error variation is much higher in POD-QDEIM compared to our models. This in turn means the model prediction from POD-QDEIM should be accompanied with larger uncertainties. As one increases the number of sensors, training error of both models decay drastically. Meanwhile, testing error of POD-QDEIM increases and that of NIF-based framework visually stays around the same level.

Comparison on contours of sea surface temperature among ground true, our framework and POD-QDEIM for the very first week of 1990 (in training dataset) is displayed in fig. 26. We can roughly see the inferior performance of POD-QDEIM comes from at least two sources:

1. POD-QDEIM overshoots for the average temperature in the middle of the Pacific Ocean.

2. POD-QDEIM misses small scales structures while NIF-based framework captures them well.

As the number of sensors increases, both models performs better and better on training data.

In order to further analyze the model generalization performance on sea surface temperature data, we compute the temporal average of squared error on unseen testing data (2006 to 2021) and plot its corresponding spatial distribution with varying number of sensors p in fig. 27. Evidently, except at 5 sensors where NIF-SS shows a similar testing error compared to POD-QDEIM, NIF-SS generalizes much better than POD-QDEIM regardless of the number of sensors p. As p increases, the error distribution of both two frameworks tends to contain more small-scale patches.

An inspection on error distribution from NIF-based framework shows interesting correlations between regions that are difficult to predict and ocean circulation patterns. Let's focus on the NIF-SS with error magnified 5 times (third column) in fig. 27. For example, when $p = 5$, we see the regions that are most difficult to predict by NIF-SS happen mostly at the location of *Gulf stream*, *North Pacific gyre* and *Norwegian current*.

Appendix D. Implementation of NIF with SIREN

We adopt SIREN (Sitzmann et al., 2020), which is standard MLP with ω_0-scaled sine activations, as our ShapeNet. In this work we take $\omega_0 = 30$ and find it to be sufficient for all cases. The initialization of SIREN is given in appendix D.1. Based on SIREN, we also borrow the ResNet-like structure (Lu et al., 2021c) to further improve training performance, which is presented in appendix D.2. Appendix D.3 describes how we connect ParameterNet with ShapeNet. Finally, practical advice on training NIF with SIREN is given in appendix D.4.

D.1 Initialization of SIREN

First of all, one should normalize each component of input for SIREN as $\mathcal{U}(-1,1)$ approximately. $\mathcal{U}(a,b)$ denotes uniform distribution on interval $[a,b]$, $a \leq b \in \mathbb{R}$. For example, in eq. (14), we normalize one coordinate component x as

$$\tilde{x} = \frac{x - (\min(x) + \max(x))/2}{(\max(x) - \min(x))/2}. \tag{14}$$

For the output u, we choose the standard normalization (zero mean and unit variance).

Second, SIREN requires a special initialization of weights and biases to achieve superior performance. Without loss of generality, consider a standard MLP equipped with ω_0-scaled sin as activation functions and units structure as n_i-n_h-n_h-n_h-n_o. n_i/n_o denotes the dimension of input/output layer. n_h denotes the dimension of hidden layer. Next, we initialize weights and biases of input layer component-wise in eq. (15) as

$$\mathbf{W}_{1,(j,k)} \sim \mathcal{U}\left(-\frac{1}{n_i}, \frac{1}{n_i}\right), \quad \boldsymbol{b}_{1,(j)} \sim \mathcal{U}\left(-\frac{1}{\sqrt{n_i}}, \frac{1}{\sqrt{n_i}}\right), \tag{15}$$

where subscript (j,k) denotes the j-th row k-th column component. We initialize all other layer weights and biases following eq. (16),

$$\mathbf{W}_{(j,k)} \sim \mathcal{U}\left(-\frac{\sqrt{6/n_h}}{\omega_0}, \frac{\sqrt{6/n_h}}{\omega_0}\right), \quad \boldsymbol{b}_{(j)} \sim \mathcal{U}\left(-\frac{1}{\sqrt{n_h}}, \frac{1}{\sqrt{n_h}}\right). \tag{16}$$

Note that ω_0-scaled sine activation is defined as $\sigma(x) = \sin(\omega_0 x)$.

D.2 ResNet-like block

After first layer, we use a design of ResNet-like block (Lu et al., 2021c) to build ShapeNet. The configuration is displayed in fig. 5. Without loss of generality, denote η_i as the input of i-th such block, we have

$$\zeta = \sin(\omega_0 \mathbf{W}_{i_1} \eta_i + \boldsymbol{b}_{i_1}), \tag{17}$$

$$\eta_{i+1} = \frac{1}{2}\left(\eta_i + \sin(\omega_0 \mathbf{W}_{i_2} \zeta + \boldsymbol{b}_{i_2})\right), \tag{18}$$

where η_{i+1} is the input for the next $i+1$-th block.

D.3 Building ParameterNet

The final step is to connect the output of ParameterNet to ShapeNet. Note that now only ParameterNet contains undetermined parameters while the parameters in ShapeNet are subject to the output of ParameterNet. Therefore, we only need to carefully design the initialization of last layer weights and biases of ParameterNet in order to be consistent with the requirement in appendix D.1. Recall that the network structure of NIF is a special case of hypernetworks of MLP that focuses on decoupling spatial complexity away from other factors. We take the initialization scheme of hypernetworks in SIREN (Sitzmann et al., 2020). The idea is to simply multiply the last layer initial weights by a small factor, e.g., 10^{-2}, while keeping the last layer initial biases to match the distribution in appendix D.1.

D.4 Practical advice for training

We empirically found successful and stable training NIF with SIREN using Adam optimizer requires 1) *small* learning rate typically around 10^{-4} to 10^{-5}, 2) *large* batch size, which often takes to be filling up the whole GPU memory. When debugging, it is recommended to monitor training error at every 5 epoch and plot the prediction at the same time. If one found training error, e.g., mean-square error, stuck at relatively high in the first 40 epoch and the prediction is completely useless, then one should further decrease the learning rate or increasing the batch size. We found NIF with SIREN also tend to fit large scale structure first then small scales.

However, it is not always possible to increase the batch size given limited GPU memory especially in the fully 3D case. We found that if we fully parameterize the weights and biases of ShapeNet (as oppose to only parameterizing the last layer in section 2.3), the memory footprints of NIF becomes relatively large due to the tensor multiplication with `einsum` in implementation. Therefore, the maximal number of epochs affordable can be limited, which can lead to too long training time or unstable training if even one set learning rate as low as 10^{-5} in the worst case.

If one doesn't have access to good GPU resources, a more economic remedy on a single GPU is to use *small-batch* training, which is well-known to have better generalization compared to large-batch training (Masters and Luschi, 2018). However, here we are only looking for stable training of NIF with SIREN that uses small batch sizes to fit in a single GPU. We empirically found L4 optimizer (Rolinek and Martius, 2018) can have stable training performance in the *small* batch regime compared to Adam for the same batch size. Empirically, we found L4 optimizer can reduce *minimal batch size* required for stable training by at least 10 times compared to Adam optimizer, while only has a slight increase in training cost. Thus, one can increase the capacity of NIF by 10 times without sacrificing much in the overall training time and performance. Note that such capacity can be crucial in training large-scale 3D fully turbulence dataset with limited GPU resources.

For all the results shown in this paper, we have used Nvidia Tesla P100 (16 GB), Nvidia GeForce RTX 2080 GPU (12 GB), and Nvidia A6000 GPU (48 GB). If available, the simplest remedy is to use data-parallelism with multiple GPUs. We have implemented data-parallel capability for NIF in our Github repo (`https://github.com/pswpswpsw/nif`) on multiple GPUs and it scales well. We leave the complete study on distributed learning with NIF for future work.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

Shady E Ahmed, Suraj Pawar, Omer San, Adil Rasheed, Traian Iliescu, and Bernd R Noack. On closures for reduced order models—a spectrum of first-principle to machine-learned avenues. *Physics of Fluids*, 33(9):091301, 2021a.

Shady E Ahmed, Omer San, Adil Rasheed, and Traian Iliescu. Nonlinear proper orthogonal decomposition for convection-dominated flows. *Physics of Fluids*, 33(12):121702, 2021b.

James Ahrens, Berk Geveci, and Charles Law. Paraview: An end-user tool for large data visualization. *The visualization handbook*, 717(8), 2005.

Ann S Almgren, Vincent E Beckner, John B Bell, MS Day, Louis H Howell, CC Joggerst, MJ Lijewski, Andy Nonaka, M Singer, and Michael Zingale. Castro: A new compressible astrophysical solver. i. hydrodynamics and self-gravity. *The Astrophysical Journal*, 715 (2):1221, 2010.

David Amsallem, Sunil Deolalikar, Fazzel Gurrola, and Charbel Farhat. Model predictive control under coupled fluid-structure constraints using a database of reduced-order models on a tablet. In *21st AIAA Computational Fluid Dynamics Conference*, page 2588, 2013.

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

John Bell and Marcus Day. Adaptive methods for simulation of turbulent combustion. In *Turbulent Combustion Modeling*, pages 301–329. Springer, 2011.

Peter Benner, Serkan Gugercin, and Karen Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM review*, 57(4):483–531, 2015.

Jens Berg and Kaj Nyström. A unified deep artificial neural network approach to partial differential equations in complex geometries. *Neurocomputing*, 317:28–41, 2018.

Marsha J Berger and Joseph Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of computational Physics*, 53(3):484–512, 1984.

Amneet Pal Singh Bhalla, Rahul Bale, Boyce E Griffith, and Neelesh A Patankar. A unified mathematical framework and an adaptive numerical method for fluid–structure interaction with rigid, deforming, and elastic bodies. *Journal of Computational Physics*, 250:446–476, 2013.

Saakaar Bhatnagar, Yaser Afshar, Shaowu Pan, Karthik Duraisamy, and Shailendra Kaushik. Prediction of aerodynamic flow fields using convolutional neural networks. *Computational Mechanics*, 64(2):525–545, 2019.

Joan Bruna, Benjamin Peherstorfer, and Eric Vanden-Eijnden. Neural galerkin scheme with active learning for high-dimensional evolution equations. *arXiv preprint arXiv:2203.01360*, 2022.

Steven L Brunton and J Nathan Kutz. *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2019.

Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, page 201517384, 2016.

Steven L Brunton, Bernd R Noack, and Petros Koumoutsakos. Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics*, 52:477–508, 2020.

Greg L Bryan, Michael L Norman, Brian W O'Shea, Tom Abel, John H Wise, Matthew J Turk, Daniel R Reynolds, David C Collins, Peng Wang, Samuel W Skillman, et al. Enzo: An adaptive mesh refinement code for astrophysics. *The Astrophysical Journal Supplement Series*, 211(2):19, 2014.

Shengze Cai, Zhicheng Wang, Lu Lu, Tamer A Zaki, and George Em Karniadakis. Deepm&mnet: Inferring the electroconvection multiphysics fields based on operator approximation by neural networks. *Journal of Computational Physics*, 436:110296, 2021.

Kevin Carlberg, Charbel Bou-Mosleh, and Charbel Farhat. Efficient non-linear model reduction via a least-squares petrov–galerkin projection and compressive tensor approximations. *International Journal for Numerical Methods in Engineering*, 86(2):155–181, 2011.

Kevin Carlberg, Charbel Farhat, Julien Cortial, and David Amsallem. The gnat method for nonlinear model reduction: effective implementation and application to computational fluid dynamics and turbulent flows. *Journal of Computational Physics*, 242:623–647, 2013.

Kathleen Champion, Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*, 116(45):22445–22451, 2019.

William M Chan. Overset grid technology development at nasa ames research center. *Computers & Fluids*, 38(3):496–503, 2009.

Oscar Chang, Lampros Flokas, and Hod Lipson. Principled weight initialization for hypernetworks. In *International Conference on Learning Representations*, 2019.

Saifon Chaturantabut and Danny C Sorensen. Nonlinear model reduction via discrete empirical interpolation. *SIAM Journal on Scientific Computing*, 32(5):2737–2764, 2010.

Kevin K Chen, Jonathan H Tu, and Clarence W Rowley. Variants of dynamic mode decomposition: boundary condition, koopman, and fourier analyses. *Journal of nonlinear science*, 22(6):887–915, 2012.

Peter Yichen Chen, Maurizio Chiaramonte, Eitan Grinspun, and Kevin Carlberg. Model reduction for the material point method via learning the deformation map and its spatial-temporal gradients. *arXiv preprint arXiv:2109.12390*, 2021.

Seddik M Djouadi. On the optimality of the proper orthogonal decomposition and balanced truncation. In *2008 47th IEEE Conference on Decision and Control*, pages 4221–4226. IEEE, 2008.

Zlatko Drmac and Serkan Gugercin. A new selection operator for the discrete empirical interpolation method—improved a priori error bound and extensions. *SIAM Journal on Scientific Computing*, 38(2):A631–A648, 2016.

Karthik Duraisamy, Gianluca Iaccarino, and Heng Xiao. Turbulence modeling in the age of data. *Annual Review of Fluid Mechanics*, 51:357–377, 2019.

Thomas Duriez, Steven L Brunton, and Bernd R Noack. *Machine learning control-taming nonlinear dynamics and turbulence*. Springer, 2017.

Soheil Esmaeilzadeh, Kamyar Azizzadenesheli, Karthik Kashinath, Mustafa Mustafa, Hamdi A Tchelepi, Philip Marcus, Mr Prabhat, Anima Anandkumar, et al. Meshfreeflownet: A physics-constrained deep continuous space-time super-resolution framework. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–15. IEEE, 2020.

James L Flanagan. *Speech analysis synthesis and perception*, volume 3. Springer Science & Business Media, 2013.

M. Frangos, Y. Marzouk, K. Willcox, and B. van Bloemen Waanders. *Surrogate and Reduced-Order Modeling: A Comparison of Approaches for Large-Scale Statistical Inverse Problems*, chapter 7, pages 123–149. John Wiley & Sons, Ltd, 2010. ISBN 9780470685853. doi: https://doi.org/10.1002/9780470685853.ch7. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470685853.ch7.

Nicholas Geneva and Nicholas Zabaras. Modeling the dynamics of pde systems with physics-constrained deep auto-regressive networks. *Journal of Computational Physics*, 403:109056, 2020.

William K George. Insight into the dynamics of coherent structures from a proper orthogonal decomposition dy. In *International seminar on wall turbulence*, 1988.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 558–567, 2019.

Jan S Hesthaven, Gianluigi Rozza, Benjamin Stamm, et al. *Certified reduced basis methods for parametrized partial differential equations*, volume 590. Springer, 2016.

Philip Holmes, John L Lumley, Gahl Berkooz, and Clarence W Rowley. *Turbulence, coherent structures, dynamical systems and symmetry*. Cambridge university press, 2012.

Kurt Hornik, Maxwell Stinchcombe, Halbert White, et al. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Thomas JR Hughes. *The finite element method: linear static and dynamic finite element analysis*. Courier Corporation, 2012.

Manu Kalia, Steven L Brunton, Hil GE Meijer, Christoph Brune, and J Nathan Kutz. Learning normal form autoencoders for data-driven discovery of universal, parameter-dependent governing equations. *arXiv preprint arXiv:2106.05102*, 2021.

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

Aly-Khan Kassam and Lloyd N Trefethen. Fourth-order time-stepping for stiff pdes. *SIAM Journal on Scientific Computing*, 26(4):1214–1233, 2005.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces. *arXiv preprint arXiv:2108.08481*, 2021.

Slawomir Koziel and Leifur Leifsson. *Surrogate-based modeling and optimization*. Springer, 2013.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25: 1097–1105, 2012.

J Nathan Kutz. Deep learning in fluid dynamics. *Journal of Fluid Mechanics*, 814:1–4, 2017.

Henning Lange, Steven L Brunton, and J Nathan Kutz. From fourier to koopman: Spectral methods for long-term time series prediction. *J. Mach. Learn. Res.*, 22:41–1, 2021.

Sangseung Lee and Donghyun You. Data-driven prediction of unsteady flow over a circular cylinder using deep learning. *Journal of Fluid Mechanics*, 879:217–254, 2019.

Yi Li, Eric Perlman, Minping Wan, Yunke Yang, Charles Meneveau, Randal Burns, Shiyi Chen, Alexander Szalay, and Gregory Eyink. A public turbulence database cluster and applications to study lagrangian evolution of velocity increments in turbulence. *Journal of Turbulence*, 9:N31, 2008. doi: 10.1080/14685240802376389. URL https://doi.org/10.1080/14685240802376389.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020a.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020b.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Andrew Stuart, Kaushik Bhattacharya, and Anima Anandkumar. Multipole graph neural operator for parametric partial differential equations. *Advances in Neural Information Processing Systems*, 33:6755–6766, 2020c.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Markov neural operators for learning chaotic systems. *arXiv preprint arXiv:2106.06898*, 2021a.

Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *arXiv preprint arXiv:2111.03794*, 2021b.

Michel Loeve. *Probability theory: foundations, random sequences*. New York, NY: Van Nostrand, 1955.

Jean-Christophe Loiseau, Steven Brunton, and Bernd Noack. From the pod-galerkin method to sparse manifold models, 2021.

Zichao Long, Yiping Lu, Xianzhong Ma, and Bin Dong. Pde-net: Learning pdes from data. In *International Conference on Machine Learning*, pages 3208–3216. PMLR, 2018.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021a.

Lu Lu, Xuhui Meng, Shengze Cai, Zhiping Mao, Somdatta Goswami, Zhongqiang Zhang, and George Em Karniadakis. A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data. *arXiv preprint arXiv:2111.05512*, 2021b.

Yuzhe Lu, Kairong Jiang, Joshua A Levine, and Matthew Berger. Compressive neural representations of volumetric scalar fields. *arXiv preprint arXiv:2104.04523*, 2021c.

Hugo FS Lui and William R Wolf. Construction of reduced-order models for fluid flows using deep feedforward neural networks. *Journal of Fluid Mechanics*, 872:963–994, 2019.

John Leask Lumley. The structure of inhomogeneous turbulent flows. *Atmospheric turbulence and radio wave propagation*, 1967.

Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature Communications*, 9(1):4950, 2018. doi: 10.1038/s41467-018-07210-0.

Krithika Manohar, Bingni W Brunton, J Nathan Kutz, and Steven L Brunton. Data-driven sparse sensor placement for reconstruction: Demonstrating the benefits of exploiting known patterns. *IEEE Control Systems Magazine*, 38(3):63–86, 2018.

Zhiping Mao, Lu Lu, Olaf Marxen, Tamer A Zaki, and George Em Karniadakis. Deepm&mnet for hypersonics: Predicting the coupled flow and finite-rate chemistry behind a normal shock using neural-network approximation of operators. *Journal of Computational Physics*, 447:110698, 2021.

Andreas Mardt, Luca Pasquali, Hao Wu, and Frank Noé. Vampnets for deep learning of molecular kinetics. *Nature communications*, 9(1):1–11, 2018.

Andreas Mardt, Luca Pasquali, Frank Noé, and Hao Wu. Deep learning markov and koopman models with physical constraints. In *Mathematical and Scientific Machine Learning*, pages 451–475. PMLR, 2020.

Dominic Masters and Carlo Luschi. Revisiting small batch training for deep neural networks. *arXiv preprint arXiv:1804.07612*, 2018.

Amrita Mathuriya, Deborah Bard, Peter Mendygral, Lawrence Meadows, James Arnemann, Lei Shao, Siyu He, Tuomas Kärnä, Diana Moise, Simon J Pennycook, et al. Cosmoflow: Using deep learning to learn the universe at scale. In *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 819–829. IEEE, 2018.

Romit Maulik and Gianmarco Mengaldo. Pyparsvd: A streaming, distributed and randomized singular-value-decomposition library. In *2021 7th International Workshop on Data Analysis and Reduction for Big Scientific Data (DRBSD-7)*, pages 19–25. IEEE, 2021.

Romit Maulik, Arvind Mohan, Bethany Lusch, Sandeep Madireddy, Prasanna Balaprakash, and Daniel Livescu. Time-series learning of latent-space dynamics for reduced-order model closure. *Physica D: Nonlinear Phenomena*, 405:132368, 2020.

Beverley J McKeon and Ati S Sharma. A critical-layer framework for turbulent pipe flow. *Journal of Fluid Mechanics*, 658:336–382, 2010.

Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer, 2020.

Arvind Mohan, Don Daniel, Michael Chertkov, and Daniel Livescu. Compressed convolutional lstm: An efficient deep learning framework to model high fidelity 3d turbulence. *arXiv preprint arXiv:1903.00033*, 2019.

Arvind T Mohan and Datta V Gaitonde. A deep learning based approach to reduced order modeling for turbulent flow control using lstm neural networks. *arXiv preprint arXiv:1804.09269*, 2018.

Takaaki Murata, Kai Fukami, and Koji Fukagata. Nonlinear mode decomposition with convolutional neural networks for fluid dynamics. *Journal of Fluid Mechanics*, 882, 2020.

Bernd R Noack, Konstantin Afanasiev, MAREK MORZYŃSKI, Gilead Tadmor, and Frank Thiele. A hierarchy of low-dimensional models for the transient and post-transient cylinder wake. *Journal of Fluid Mechanics*, 497:335–363, 2003.

A Nonaka, JB Bell, MS Day, C Gilet, AS Almgren, and ML Minion. A deferred correction coupling strategy for low mach number flow with complex chemistry. *Combustion Theory and Modelling*, 16(6):1053–1088, 2012.

Samuel E Otto and Clarence W Rowley. Linearly recurrent autoencoder networks for learning dynamics. *SIAM Journal on Applied Dynamical Systems*, 18(1):558–593, 2019.

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

Shaowu Pan and Karthik Duraisamy. Long-time predictive modeling of nonlinear dynamical systems using neural networks. *Complexity*, 2018, 2018b.

Shaowu Pan and Karthik Duraisamy. Physics-informed probabilistic learning of linear embeddings of nonlinear dynamics with guaranteed stability. *SIAM Journal on Applied Dynamical Systems*, 19(1):480–509, 2020.

Shaowu Pan, Nicholas Arnold-Medabalimi, and Karthik Duraisamy. Sparsity-promoting algorithms for the discovery of informative koopman-invariant subspaces. *Journal of Fluid Mechanics*, 917, 2021.

Demetrios T Papageorgiou and Yiorgos S Smyrlis. The route to chaos for the kuramoto-sivashinsky equation. *Theoretical and Computational Fluid Dynamics*, 3(1):15–42, 1991.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019.

Benjamin Peherstorfer and Karen Willcox. Data-driven operator inference for nonintrusive projection-based model reduction. *Computer Methods in Applied Mechanics and Engineering*, 306:196–215, 2016.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W Battaglia. Learning mesh-based simulation with graph networks. *arXiv preprint arXiv:2010.03409*, 2020.

Elizabeth Qian, Boris Kramer, Benjamin Peherstorfer, and Karen Willcox. Lift & learn: Physics-informed machine learning for large-scale nonlinear dynamical systems. *Physica D: Nonlinear Phenomena*, 406:132401, 2020.

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

Maziar Raissi, Alireza Yazdani, and George Em Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367(6481):1026–1030, 2020.

Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

Richard W Reynolds, Nick A Rayner, Thomas M Smith, Diane C Stokes, and Wanqiu Wang. An improved in situ and satellite sst analysis for climate. *Journal of climate*, 15 (13):1609–1625, 2002.

Salah Rifai, Grégoire Mesnil, Pascal Vincent, Xavier Muller, Yoshua Bengio, Yann Dauphin, and Xavier Glorot. Higher order contractive auto-encoder. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 645–660. Springer, 2011.

Michal Rolinek and Georg Martius. L4: Practical loss-based stepsize adaptation for deep learning. *arXiv preprint arXiv:1802.05074*, 2018.

Clarence W Rowley, Tim Colonius, and Richard M Murray. Model reduction for compressible flows using pod and galerkin projection. *Physica D: Nonlinear Phenomena*, 189(1-2): 115–129, 2004.

Clarence W Rowley, Igor Mezić, Shervin Bagheri, Philipp Schlatter, and Dan S Henningson. Spectral analysis of nonlinear flows. *Journal of fluid mechanics*, 641:115–127, 2009.

Omer San, Romit Maulik, and Mansoor Ahmed. An artificial neural network framework for reduced order modeling of transient flows. *Communications in Nonlinear Science and Numerical Simulation*, 77:271–287, 2019.

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

Peter J Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of fluid mechanics*, 656:5–28, 2010.

Peter J Schmid. Dynamic mode decomposition and its variants. *Annual Review of Fluid Mechanics*, 54, 2022.

Jürgen Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992.

Justin Sirignano and Konstantinos Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375:1339–1364, 2018.

Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33, 2020.

Jaroslaw Sobieszczanski-Sobieski and Raphael T Haftka. Multidisciplinary aerospace design optimization: survey of recent developments. *Structural optimization*, 14(1):1–23, 1997.

Luning Sun, Han Gao, Shaowu Pan, and Jian-Xun Wang. Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Computer Methods in Applied Mechanics and Engineering*, 361:112732, 2020.

Mark Sussman. A parallelized, adaptive algorithm for multiphase flows in general geometries. *Computers & structures*, 83(6-7):435–444, 2005.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Proc. NIPS*, Montreal, CA, 2014. URL http://arxiv.org/abs/1409.3215.

Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

Kunihiko Taira and Tim Colonius. The immersed boundary method: a projection approach. *Journal of Computational Physics*, 225(2):2118–2137, 2007.

Kunihiko Taira, Steven L Brunton, Scott TM Dawson, Clarence W Rowley, Tim Colonius, Beverley J McKeon, Oliver T Schmidt, Stanislav Gordeyev, Vassilios Theofilis, and Lawrence S Ukeiley. Modal analysis of fluid flows: An overview. *Aiaa Journal*, pages 4013–4041, 2017.

Kunihiko Taira, Maziar S Hemati, Steven L Brunton, Yiyang Sun, Karthik Duraisamy, Shervin Bagheri, Scott TM Dawson, and Chi-An Yeh. Modal analysis of fluid flows: Applications and outlook. *AIAA Journal*, 58(3):998–1022, 2020.

Naoya Takeishi, Yoshinobu Kawahara, and Takehisa Yairi. Learning koopman invariant subspaces for dynamic mode decomposition. In *Advances in Neural Information Processing Systems*, pages 1130–1140, 2017.

Matthew Tancik, Pratul P Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *arXiv preprint arXiv:2006.10739*, 2020.

Romain Teyssier. Cosmological hydrodynamics with adaptive mesh refinement-a new high resolution code called ramses. *Astronomy & Astrophysics*, 385(1):337–364, 2002.

Aaron Towne, Oliver T Schmidt, and Tim Colonius. Spectral proper orthogonal decomposition and its relationship to dynamic mode decomposition and resolvent analysis. *Journal of Fluid Mechanics*, 847:821–867, 2018.

John Towns, Timothy Cockerill, Maytal Dahan, Ian Foster, Kelly Gaither, Andrew Grimshaw, Victor Hazlewood, Scott Lathrop, Dave Lifka, Gregory D Peterson, et al. Xsede: accelerating scientific discovery. *Computing in science & engineering*, 16(5):62–74, 2014.

Itamar Turner-Trauring. The fil memory profiler for python. https://github.com/pythonspeed/filprofiler, 2019.

Stefan Van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. scikit-image: image processing in python. *PeerJ*, 2:e453, 2014.

J-L Vay, P Colella, JW Kwan, P McCorquodale, DB Serafini, A Friedman, DP Grote, G Westenskow, J-C Adam, A Heron, et al. Application of adaptive mesh refinement to particle-in-cell simulations of plasmas and beams. *Physics of Plasmas*, 11(5):2928–2934, 2004.

Hengjie Wang, Robert Planas, Aparna Chandramowlishwaran, and Ramin Bostanabad. Mosaic flows: A transferable deep learning framework for solving pdes on unseen domains. *Computer Methods in Applied Mechanics and Engineering*, 389:114424, 2022.

Qian Wang, Nicolò Ripamonti, and Jan S Hesthaven. Recurrent neural network closure of parametric pod-galerkin reduced-order models based on the mori-zwanzig formalism. *Journal of Computational Physics*, 410:109402, 2020.

Sifan Wang, Hanwen Wang, and Paris Perdikaris. On the eigenvector bias of fourier feature networks: From regression to solving multi-scale pdes with physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 384:113938, 2021a.

Sifan Wang, Hanwen Wang, and Paris Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed deeponets. *arXiv preprint arXiv:2103.10974*, 2021b.

Steffen Wiewel, Moritz Becher, and Nils Thuerey. Latent space physics: Towards learning the temporal evolution of fluid flow. In *Computer graphics forum*, volume 38, pages 71–82. Wiley Online Library, 2019.

Jiayang Xu and Karthik Duraisamy. Multi-level convolutional autoencoder networks for parametric prediction of spatio-temporal dynamics. *Computer Methods in Applied Mechanics and Engineering*, 372:113379, 2020.

Jiayang Xu, Aniruddhe Pradhan, and Karthikeyan Duraisamy. Conditionally parameterized, discretization-aware neural networks for mesh-based modeling of physical systems. *Advances in Neural Information Processing Systems*, 34, 2021.

Zhi-Qin John Xu, Yaoyu Zhang, Tao Luo, Yanyang Xiao, and Zheng Ma. Frequency principle: Fourier analysis sheds light on deep neural networks. *arXiv preprint arXiv:1901.06523*, 2019.

Enoch Yeung, Soumya Kundu, and Nathan Hodas. Learning deep neural network representations for koopman operators of nonlinear dynamical systems. In *2019 American Control Conference (ACC)*, pages 4832–4839. IEEE, 2019.

Ke Yu, Benedikt Dorschner, and Tim Colonius. Multi-resolution lattice green's function method for incompressible flows. *Journal of Computational Physics*, page 110845, 2022.

Matthew J Zahr and Charbel Farhat. Progressive construction of a parametric reduced-order model for pde-constrained optimization. *International Journal for Numerical Methods in Engineering*, 102(5):1111–1135, 2015.

Linfeng Zhang, Jiequn Han, Han Wang, Roberto Car, and E Weinan. Deep potential molecular dynamics: a scalable model with the accuracy of quantum mechanics. *Physical review letters*, 120(14):143001, 2018.

Yinhao Zhu and Nicholas Zabaras. Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification. *Journal of Computational Physics*, 366:415–447, 2018.

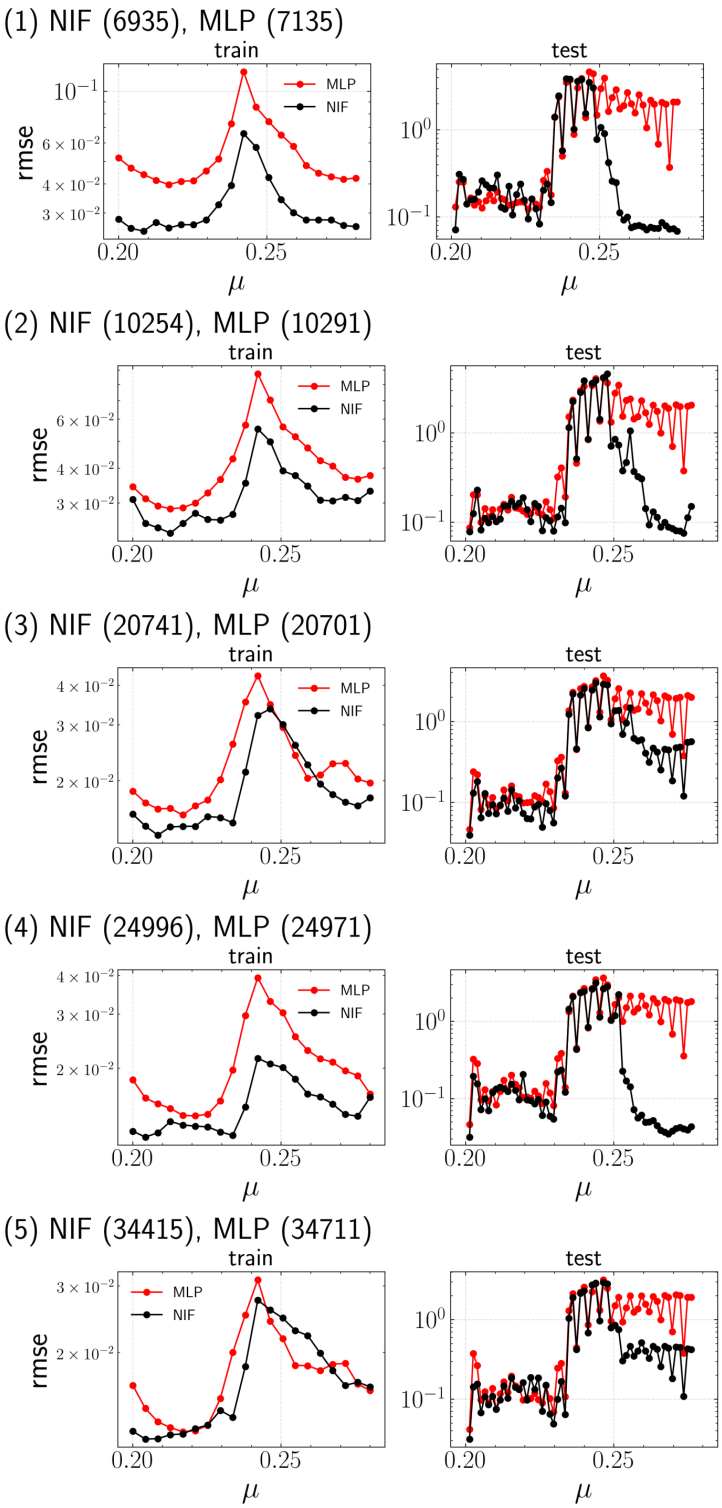

Figure 23: Comparison between NIF-Swish and MLP-Swish in terms of RMSE on the dataset of 1D parametric KS system with $0.2 < \mu < 0.28$. Training data contains 20 simulations but Model parameters vary from 7k to 34k.

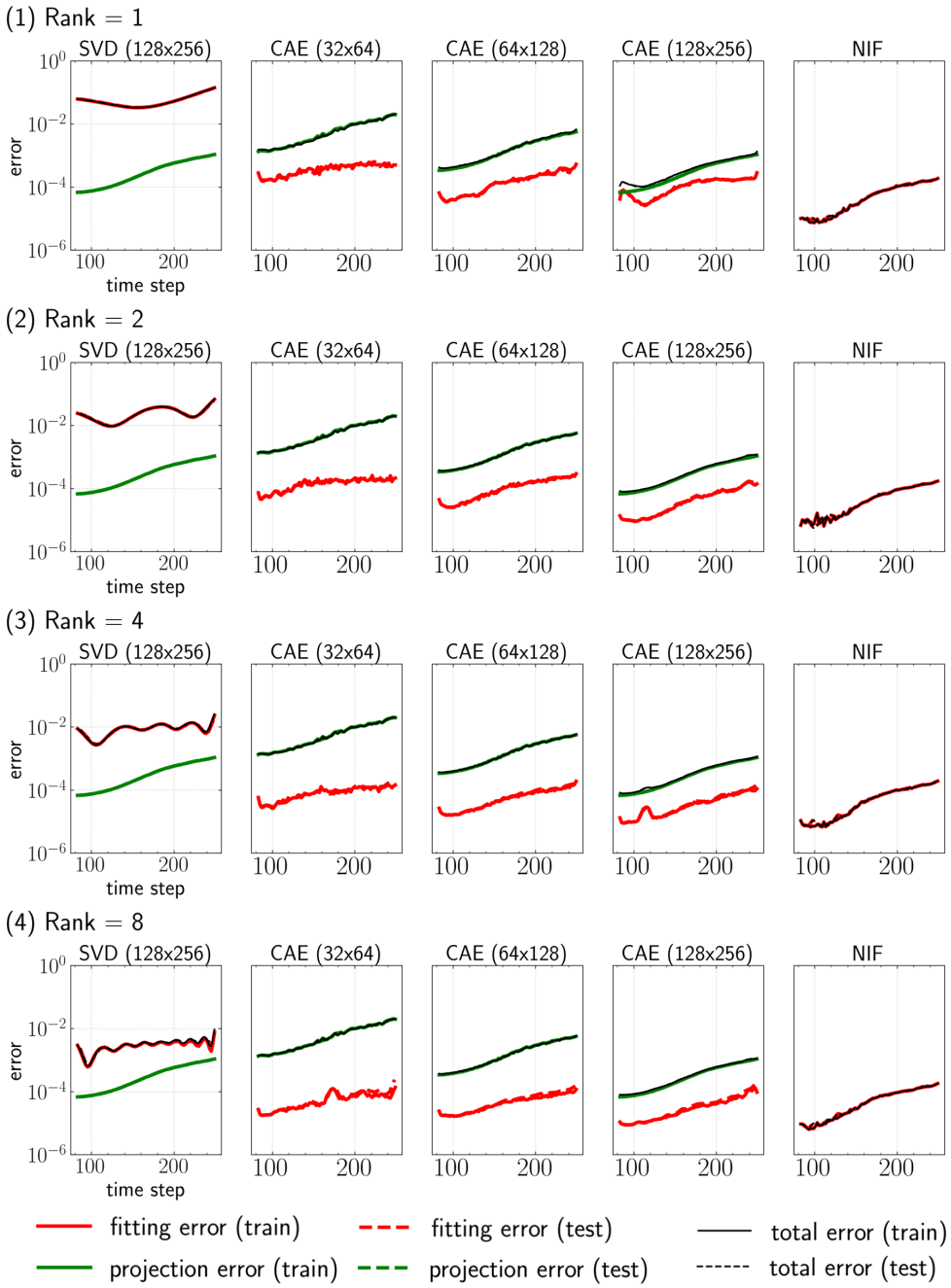

Figure 24: Evolution of fitting, projection and total error for all five dimensionality reductions methods on learning low dimensional latent subspace with $r = 2$ (top), $r = 4$ (middle), $r = 8$ (bottom).

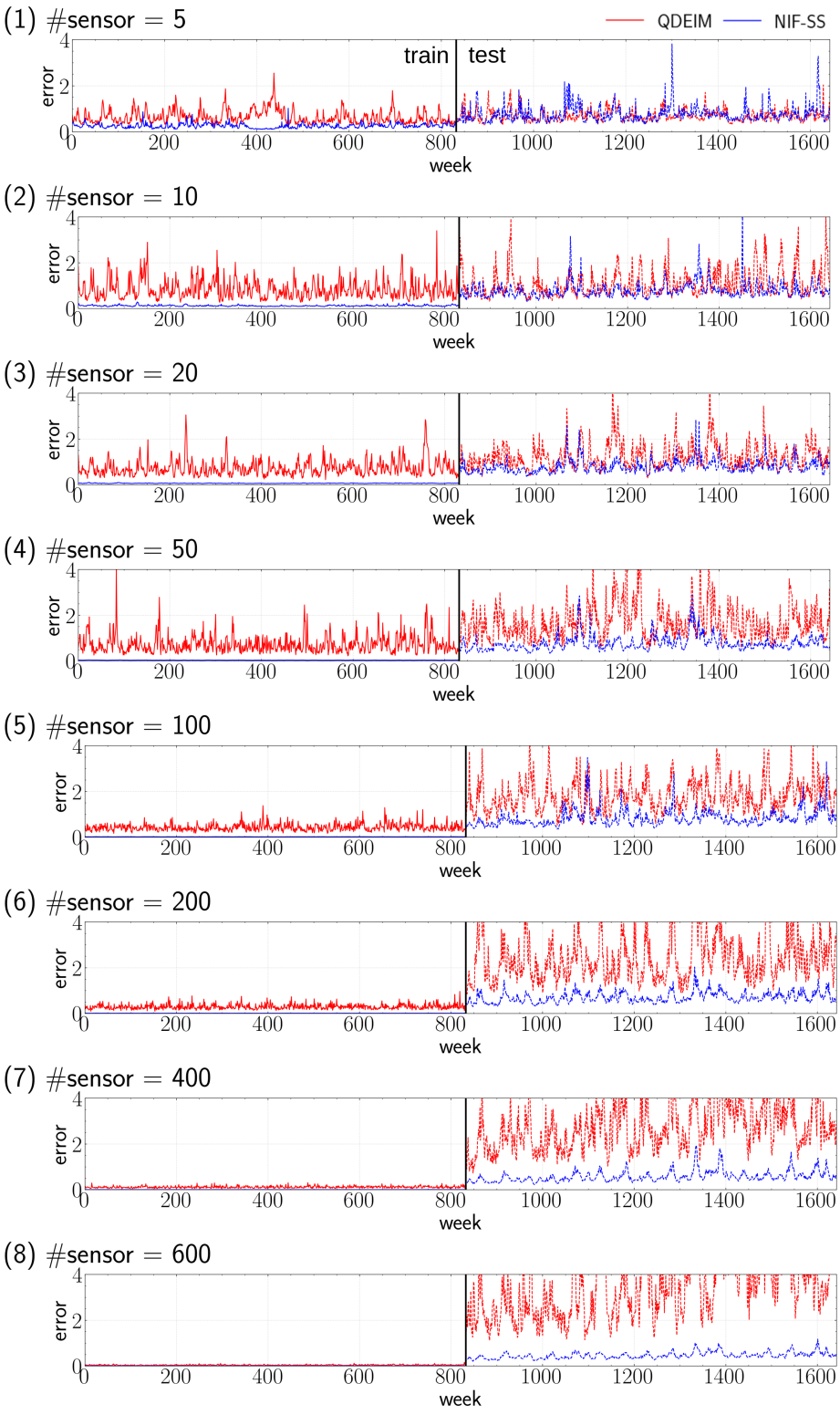

Figure 25: History of spatially mean-squared error of NIF-based framework and POD-QDEIM on sea surface temperature from 1990 to 2021 with different number of sensors p.

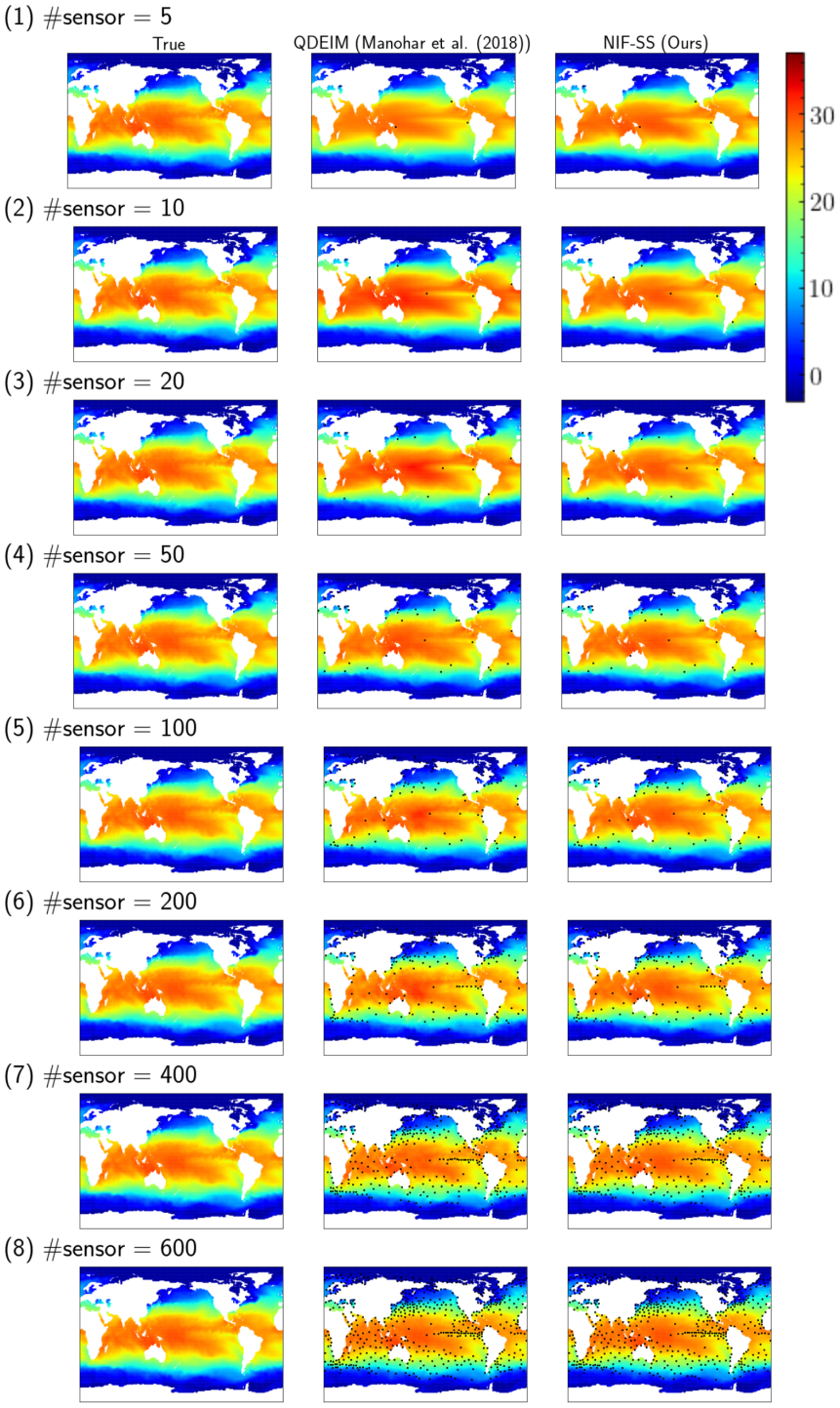

Figure 26: Sea surface temperature contour of the first week of 1990 from ground true (left column), POD-QDEIM (middle column) and our framework (right column) with varying number of sensors. Black dots indicates optimized sensor location from POD-QDEIM.

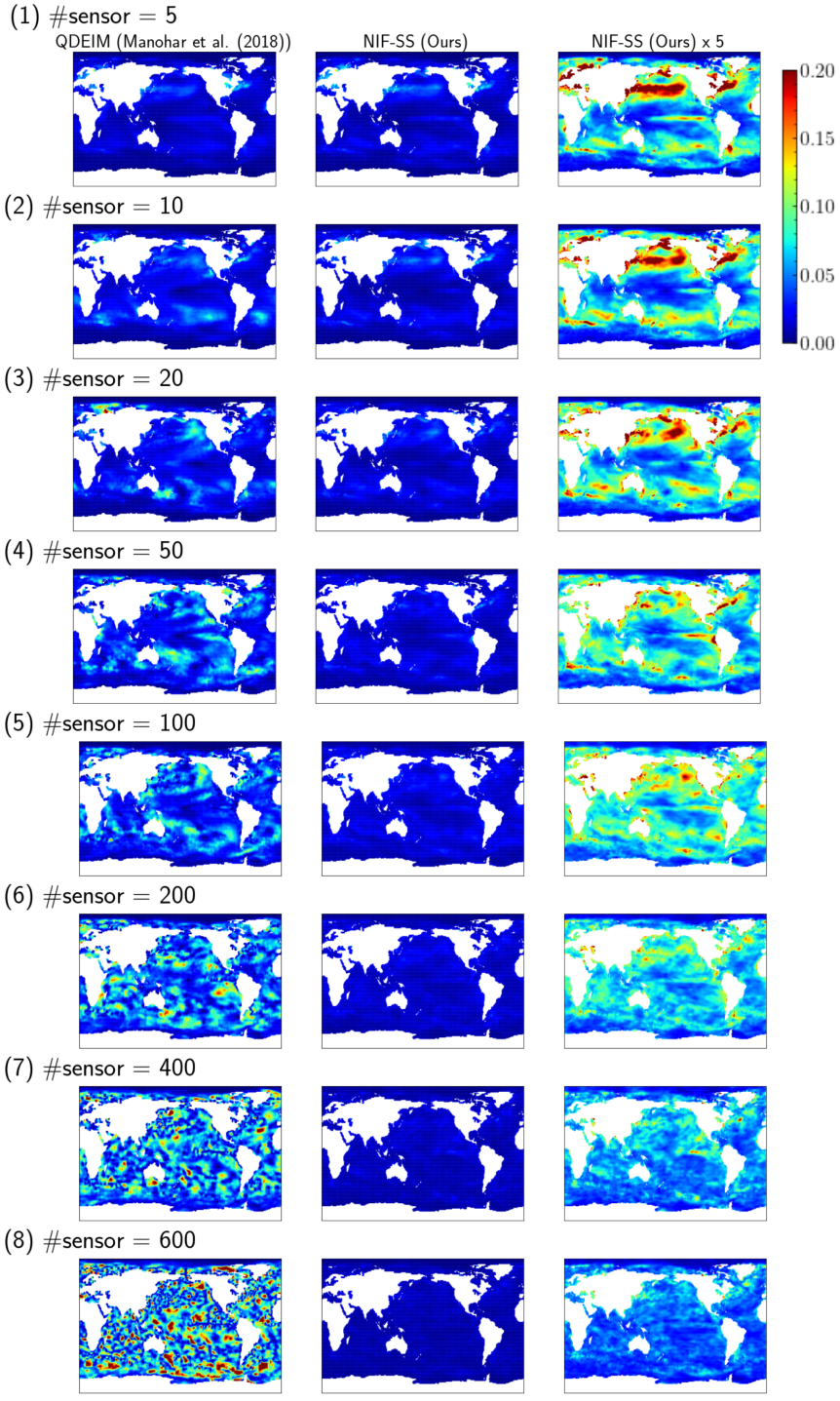

Figure 27: Spatial distribution of temporally mean squared error of NIF-based framework and POD-QDEIM with varying number of sensors. "NIF-SS x 5" means the error is magnified 5 times to increase visibility.