Latent Neural PDE Solver for Time-dependent Systems

Zijie Li[†]**, Saurabh Patil**[†]**, Dule Shu, Amir Barati Farimani**
Carnegie Mellon University
Mechanical Engineering Department
{zijieli, ssp2, dules}@andrew.cmu.edu & barati@cmu.edu

Abstract

Neural networks have shown promising potential in accelerating the numerical simulation of systems governed by partial differential equations (PDEs). Different from many existing neural network surrogates operating on the high-dimensional discretized field, we propose to learn the dynamics of the system in the latent space with much coarser discretization. In our proposed framework, a non-linear autoencoder is first trained to project the full-order representation of the system onto the mesh-reduced space, then a temporal model is trained to predict the future state in this mesh-reduced space. This reduction process simplifies the training of the temporal model by greatly reducing the computational cost with a fine discretization. We study the capability of the proposed framework on 2D/3D fluid flows and showcase that it has competitive performance compared to the model that operates on full-order space.

1 Introduction

Many intricate physical processes, from the interaction of protein dynamics to the movement of a celestial body, can be described by time-dependent partial differential equations (PDEs). The simulation of these processes is often conducted by solving these equations numerically, which requires fine discretization to resolve the necessary spatiotemporal domain to reach convergence. Deep neural network surrogates [5, 42, 48, 61, 65, 71] recently emerged as a computationally less-expensive alternative, with the potential to improve the efficiency of simulation by relaxing the requirement for fine discretization and attaining a higher accuracy on coarser grids compared to classical numerical solver [5, 42, 75].

For time-dependent systems, many neural-network-based models address the problem by approximating the solution operator $\mathcal{G}$ that maps the state u_t to $u_{t+\Delta t}$, where the input and output are sampled on discretization grid $\{D_i, D_h\}$ respectively. The input discretization grid can either remain unchanged between every layer inside the network [5, 9, 42], or fit into a hierarchical structure [19, 44, 58, 64, 79, 87] that resembles the V-Cycle in classical multi-grid method. Hierarchical network structures have been a common model architectural choice in the field of image segmentation [70] and generation [25] given their capability for utilizing multi-scale information.

In contrast to the aforementioned approaches especially those that utilize a hierarchical grid structure, our work studies the effect of decoupling dynamics prediction from upsampling/downsampling processes. Specifically, the neural network for predicting the forward dynamics (which we defined as a propagator) only operates on the coarsest resolution, while a deep autoencoder is pre-trained to compress the data from the original discretization grid D_i to the coarse grid D_l (e.g. from a 64×64 grid to an 8×8 grid). As the propagator network operates on a lower dimensional space, the training

[†]Equal contribution.

NeurIPS 2023 AI for Science Workshop.

cost is greatly reduced and can be potentially adapted to unrolled training with a longer rollout, which is often observed to be helpful to long-term stability [14, 21]. We parameterize the model with a convolutional neural network along with several other components that are popular in neural PDE solvers, including spectral convolution and several variants of attention. We test the proposed framework on 2D and 3D time-dependent PDEs with uniform grids and showcase that the model can achieve efficient data compression and accurate prediction of forward dynamics.

2 Related works

Neural PDE solver Neural PDE solvers can be categorized into the following groups based on their model design. The first group employs neural networks with mesh-specific architectures, such as convolutional layers for uniform meshes or graph layers for irregular meshes. These networks learn spatiotemporal correlations within PDE data without the knowledge of the underlying equations [5, 19, 28, 36, 38, 47, 58, 62, 63, 72, 75, 79, 82, 86]. Such a data-driven approach is useful for systems with unknown or partially known physics, such as large-scale climate modeling [33, 53, 59, 66]. The second group, known as Physics-Informed Neural Networks (PINNs) [8, 9, 20, 22, 30, 40, 41, 49, 56, 65, 76, 93], treats neural networks as a representation of the solution function. PINNs incorporate knowledge of governing equations into the loss function, including PDE residuals and consistency with boundary and initial conditions. Unlike the first group, PINNs can be trained solely on equation loss and do not necessarily require input-target data pairs. The third group, known as the neural operators[1, 3, 4, 9, 17, 22, 29, 31, 32, 42, 44, 45, 48, 50, 54], is designed to learn the mapping between function spaces. For a certain family of PDEs, neural operators can generalize and adapt to multiple discretizations without retraining. DeepONet [48] presents a pragmatic implementation of the universal operator approximation theorem[10]. Meanwhile, the concurrent research [43] in the form of the graph neural operator proposes a trainable kernel integral for approximating solution operators in parametric PDEs. Their follow-up work, Fourier Neural Operator (FNO) [42], has demonstrated high accuracy and efficiency in solving specific types of problems. Different function bases such as Fourier[15, 42, 80, 89] / wavelet bases[17], the column vectors from attention layers[9, 40], or Green's function approximation[2, 78], have been be used for operator learning. For more physically consistent predictions[46, 88], neural operator training can be combined with PINN principles.

Two-stage model for image compression and synthesis The utilization of a two-stage model for image synthesis has gained significant attention in the field of computer vision in recent years. VQ-VAEs[67] adopts a two-stage approach for generating images within a latent space. In the initial stage, the approach compresses images into this latent space, using model components such as an encoder, a codebook, and a decoder. Subsequently, in the second stage, a latent model is introduced to predict the latent characteristics of the compressed images, and the decoder from the first stage is used to transform the predicted latent representation back into image pixels. VQ-GANs[13] is developed to scale autoregressive transformers to large image generation by employing adversarial and perceptual objectives for first-stage training. Most recently, several works have developed latent diffusion models with promising results ranging from image[68], point clouds[92] to text generation[35]. Within the domain of neural PDE solvers, the widely employed Encoder-Process-Decoder (EPD) scheme, used to map the input solution at time t to the subsequent time step, stands as a conventional and direct method [6, 27, 57, 61, 71, 74]. As an alternative, researchers have explored propagating the system dynamics in the latent space, aiming to diminish computational complexity and minimize memory usage [34, 90]. Evolving the system dynamics in latent space can involve utilization of recurrent neural networks like LSTM [90], linear propagators grounded in the assumptions of the Koopman operator [37, 51, 52, 55, 77], attention mechanism [24], recurrent MLPs [39] or state-space model [60]. In this work, we propose to use an autoencoder to embed inputs into the latent space, and a simple yet effective convolutional propagator is employed to learn the dynamics of the time-dependent system within this latent space.

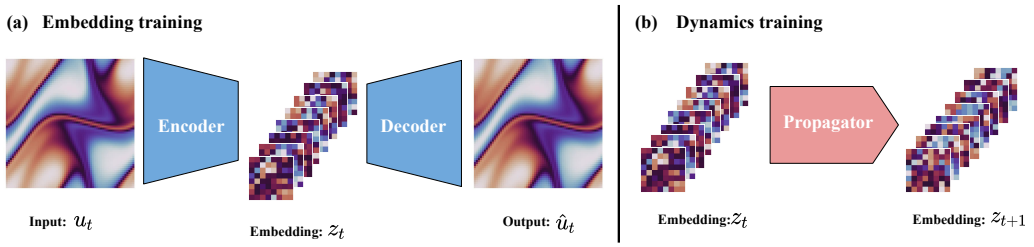


Figure 1: (a) An autoencoder is trained to project the input field to latent field with much coarser discretization. (b) A neural network is trained to predict the latent field at different time steps.

3 Methodology

3.1 Problem definition

We are interested in solving time-dependent PDEs of the following form:

$$\frac{\partial u(\mathbf{x},t)}{\partial t} = F(u(\mathbf{x},t),t), \quad \mathbf{x}\in\Omega, t\in[0,T] \tag{1}$$

$$u(\mathbf{x},0) = u_0(\mathbf{x}), \quad \mathbf{x}\in\Omega, \tag{2}$$

where T denotes the time horizon and some boundary condition for $\mathbf{x}\in\partial\Omega$ is provided *a priori*. To solve this initial value problem, a neural network is trained to approximate the following mapping:

$$u(\mathbf{x},t+\Delta t) = \mathcal{A}(u(\mathbf{x},t)), \tag{3}$$

with a fixed Δt, and the system is assumed to be Markovian such that $u(\mathbf{x},t+2\Delta t) = \mathcal{A}(\mathcal{A}(u(\mathbf{x},t)))$.

In practice, the function of interest at a particular time step $u(\cdot,t)$ is sampled on a m-point discretization grid D. For a hierarchical model like U-Net, the grid will be altered internally between different layers and the mapping $\mathcal{A}$ is a composition of a sequence of mapping $\{\mathcal{A}_0,\dots,\mathcal{A}_l\}$ which operates on grids $\{D_0,\dots,D_l\}$ with $D_0 = D$ and the number of grid points $m_l < m_{l-1} < \cdots < m_0$. In contrast to the aforementioned hierarchical model, we propose to learn $\mathcal{A}$ on the coarsest grid D_l.

3.2 Autoencoder for dimension reduction

One of the most straightforward ways to project the function from the original grid to a coarser grid is interpolation (*e.g.*, bicubic interpolation). However, interpolation can result in significant information loss about the function, as a coarser grid can only evaluate a limited bandwidth and cannot distinguish frequencies that are higher than the Nyquist frequency. To achieve a less lossy compression of the input, we train an encoder network ϕ to project the input into latent space when coarsening its spatial grid. In the meantime, we train the decoder network ψ to recover the input from the latent embedding that are represented on the coarse grid. The goal of training these two networks is to achieve data compression without too much loss of information such that their composition approximates an identity mapping: $I \approx \phi\circ\psi$.

In this work, we exploit the fact that the grid structure we are dealing with is uniform and that the majority part of the autoencoder is parameterized with convolutional neural networks (CNN) which are effective for compressing imagery data [13, 69, 84]. On top of the CNNs modules, we also introduced several other modules that have been shown to be effective for PDE surrogate modeling.

Spectral convolution Spectral convolution layer is first proposed in Fourier Neural Operator [42] as a parameterization of the learnable kernel integral [32]. It applies a discrete Fourier transform to the input and then multiplies the k-lowest modes with learnable complex weights. Given input function u_l, the spectral convolution computes the kernel integral as follows:

$$u_{l+1}(x) = \int_\Omega \kappa(x,y)u_l(y)dy = \sum_{\xi_1=0}^{\xi_1^{\max}} \dots \sum_{\xi_n=0}^{\xi_n^{\max}} W_j \mathbf{c}_j f_j(x), \quad j = \xi_1\xi_2\dots\xi_n \tag{4}$$

where $W \in \mathbb{C}^{(\xi_1^{\max} \times \xi_2^{\max} \times \ldots \xi_n^{\max}) \times d_c \times d_c}$ is the learnable weight, f_j is the j-th Fourier basis function: $\exp\left(2i\pi \sum_d \frac{x_d \xi_d}{m_d}\right)$ with m_d being the resolution along the d-th dimension, x_d being the coordinate for d-th dimension, and $\mathbf{c}_j =< u_l, f_j >$ denotes the channel-wise inner product between input function and Fourier series. Unlike the CNN layer, spectral convolution is able to capture multi-scale features that correspond to different frequencies within a single layer. It is also computationally efficient on a uniform grid as the c_j can be efficiently computed via fast Fourier Transformation (FFT). In addition, Gupta and Brandstetter [19] hypothesized that suppressing high-frequency modes with spectral convolution before downsampling can further improve the performance of the network.

Attention Scaled-dot product attention [85] has become the state-of-the-art models for natural language processing [7, 11] and computer vision tasks [12] with its capability to capture non-local interactions and compute data-dependent weights. Attention is also closely related to the kernel integral [32] defined in the previous subsection, with its theoretical property on specific PDE problems analyzed in several prior works[9, 16, 31]. Given the i-th input feature vector $\mathbf{u}_i$ with channel size d_c, the (self-)attention can be defined as:

$$\mathbf{z}_i = \sum_{j=1}^{m} \alpha_{ij} \mathbf{v}_j, \quad \alpha_{ij} = \frac{\exp\left(\mathbf{q}_i \cdot \mathbf{k}_j / \sqrt{d_c}\right)}{\sum_{s=1}^{m} \exp\left(\mathbf{q}_i \cdot \mathbf{k}_s / \sqrt{d_c}\right)}, \tag{5}$$

where: $\mathbf{q}_i = W_q \mathbf{u}_i, \mathbf{k}_i = W_k \mathbf{u}_i, \mathbf{v}_i = W_v \mathbf{u}_i$ respectively, and $\{W_q, W_k, W_v\} \in \mathbb{R}^{d_c \times d_c}$ are learnable weights. We plug the self-attention layer into the decoder and investigate its effect on learning the latent embedding.

4 Experiments

We test out the proposed model on two time-dependent fluid problems and compared our model to a state-of-the-art neural PDE solver Fourier Neural Operator [42]. For all the problems we sample the data on a spatial grid of resolution 64 along each axis.

4.1 Datasets

2D incompressible flow The 2D incompressible flow we considered here is the 2D flow dataset proposed in Li et al. [42], which is based on 2D Navier-Stokes equation under vorticity formulation. The voriticity form reads as:

$$\begin{aligned} \frac{\partial \omega(\mathbf{x},t)}{\partial t} + \mathbf{u}(\mathbf{x},t) \cdot \nabla \omega(\mathbf{x},t) &= \nu \nabla^2 \omega(\mathbf{x},t) + f(\mathbf{x}), && \mathbf{x} \in (0,1)^2, t \in (0,T], \\ \nabla \cdot \mathbf{u}(\mathbf{x},t) &= 0, && \mathbf{x} \in (0,1)^2, t \in [0,T], \\ \omega(\mathbf{x},0) &= \omega_0(\mathbf{x}), && \mathbf{x} \in (0,1)^2, \end{aligned} \tag{6}$$

where ω denotes vorticity: $\omega := \nabla \times u$, the initial condition ω_0 is sampled from the Gaussian random field, the boundary condition is periodic, the viscosity coefficient ν is $1e-4$ and the forcing term is defined as: $f(\mathbf{x}) = 0.1(\sin 2\pi(x_1 + x_2) + \cos 2\pi(x_1 + x_2))$. We are interested in learning to simulate the system (by predicting vorticity) from $t = 5$ to $t = 35$ with 30 seconds of time duration. The reference numerical simulation data is generated via the pseudo-spectral method. The dataset contains 1000 trajectories where we use 900 for training and 100 for testing.

3D smoke buoyancy The second benchmark problem is 3D Navier-Stokes equation coupled with advection equation proposed in Li et al. [41] and similar 2D cases have been studied in prior works [3, 18, 81]. The equation describes the motion of rising smoke in a closed box,

$$\begin{aligned} \frac{\partial \mathbf{u}(\mathbf{x},t)}{\partial t} + \mathbf{u}(\mathbf{x},t) \cdot \nabla \mathbf{u}(\mathbf{x},t) &= \nu \nabla^2 \mathbf{u}(\mathbf{x},t) - \frac{1}{\rho} \nabla p(\mathbf{x},t) + \mathbf{f}(\mathbf{x},t), && \mathbf{x} \in (0,L)^3, t \in (0,T], \\ \frac{\partial d(\mathbf{x},t)}{\partial t} + \mathbf{u}(\mathbf{x},t) \cdot \nabla d(\mathbf{x},t) &= 0, && \mathbf{x} \in (0,L)^3, t \in (0,T], \\ \nabla \cdot \mathbf{u}(\mathbf{x},t) &= 0, && \mathbf{x} \in (0,L)^3, t \in [0,T], \\ \mathbf{u}(\mathbf{x},0) = 0, \quad \mathbf{d}(\mathbf{x},0) &= d_0(\mathbf{x}), && \mathbf{x} \in (0,L)^3, \end{aligned} \tag{7}$$

where d depicts a marker field for smoke and is subjects to the Neumann boundary condition: $\partial d/\partial n = 0$, the velocity field $\mathbf{u}$ is under Dirichlet boundary condition: $\mathbf{u}(\mathbf{x}, t) = 0, \mathbf{x} \in \partial\Omega$, the initial condition of the marker field d is sampled from a random field, the forcing term is based on the Bousinessq model $\mathbf{f}(\mathbf{x}, t) = [0, 0, \eta d(\mathbf{x}, t)]$ with η being the buoyancy factor. We study the case with viscosity coefficient $\nu = 0.003$ and buoyancy factor $\eta = 0.50$. The goal is to predict the marker field and velocity field from $t = 0$ to $t = 12$, with domain size $L = 8$. The reference simulation data is generated using *phiflow* [26] with pressure projection and Maccormack advection scheme [73]. The dataset contains 2200 trajectories amoung which we use 2000 for training and 200 for testing.

4.2 Implementation

Autoencoder The encoder and decoder are mainly built upon convolutional layers. Internally they comprise a stack of downsampling/upsampling blocks, where each block downsamples/upsamples the spatial resolution by a factor of 2. Each block contains a residual convolution block and a downsampling/upsampling layer. The residual convolution block consists of group normalization [91] and two 3×3 convolution layers. The downsampling layer uses a 3×3 convolution layer with a stride of 2, and the upsampling layer upsamples the resolution by using nearest interpolation followed by a 3×3 convolution layer. We also investigate the influence of inserting spectral convolution layers into each downsampling block and add self-attention layers to the lowest resolution following prior works on image synthesis [13, 68]. For the 2D problem, we set the latent resolution to 8×8 and the latent dimension to 16. For the 3D problem, we set the latent resolution to $16 \times 16 \times 16$ and the latent dimension to 64. In addition, on the 3D problem, we use the multi-dimensional factorized attention [41] instead of standard attention to reduce the computational cost.

Propagator We use a simple residual convolution network [23] to forecast the forward dynamics in the latent space, where each residual block contains a group normalization layer and three convolution layers with 3×3 convolution kernels. We also employ dilated convolution for the middle convolution layer to capture longer-range interaction. For the 2D problem, we use 3 residual blocks with network width 128. For the 3D problem, we use 4 residual blocks.

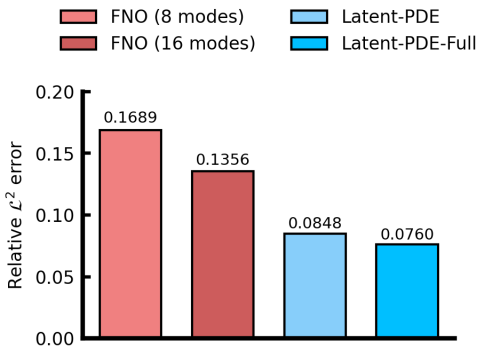
Baseline On the 2D problem, we tested out two versions of the FNO. The first version is based on the hyperparameter provided in the original paper [42], where the model width is 32 and 8 lowest modes are used at each spectral convolution layer. We also test out a larger version with a width of 64 and use a mode number of 16. On the 3D problem, we use a width of 64 and a mode number of 12 as increasing the mode number for 3D spectral convolution will drastically increase the model parameter (by cubic).

Training We first train the autoencoder by minimizing the relative L^2 reconstruction loss for around 150k iterations with constant learning rate $3e-5$ using batch size of $64/16$ respectively for 2D/3D. We then train the propagator by minimizing the mean squared error between predicted embeddings and embedding of reference data for another 150k iteration with a learning rate of $5e-4$ and a cosine annealing schedule. For FNO we train it with a learning rate $5e-4$ and a cosine annealing scheduling to minimize the relative L^2 prediction loss. The total training iterations are also set to 150k. Different from the original FNO paper, we do not use full rollout during training as we observe reducing the rollout steps during training can significantly improve the performance on NS2D. * We rollout for 2 steps for all models unless stated otherwise.

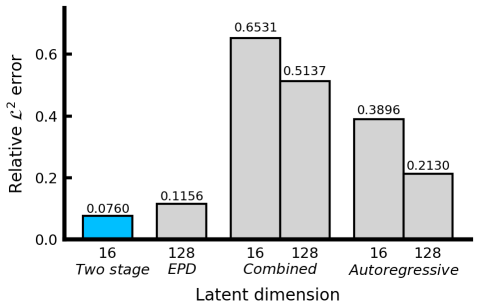
4.3 Results

In this section, we present the comparison between the proposed framework and other models. We observe that the proposed model consistently outperforms FNO which operates on the full mesh space and for lower-dimensional problems like 2D fluid flow the performance gap is more significant. On more complex 3D flow, the model is able to compress the original data to a much coarser (4 times coarser) resolution and learns to predict with accuracy on par with full-order models. Furthermore, as the temporal model operates on a much coarser discretization, we can afford longer rollout training to allow gradient propagated from farther future which can further improve the model's performance on

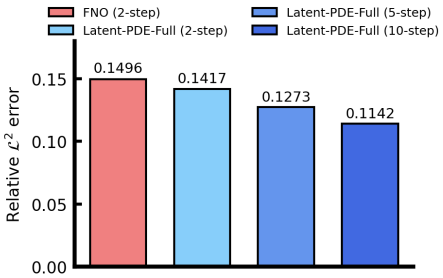
*On 2D Navier-Stokes, FNO (8 modes) has a prediction error of 0.2596 if using fully rollout training, whereas rolling out for 2 steps yields an error of 0.1689.



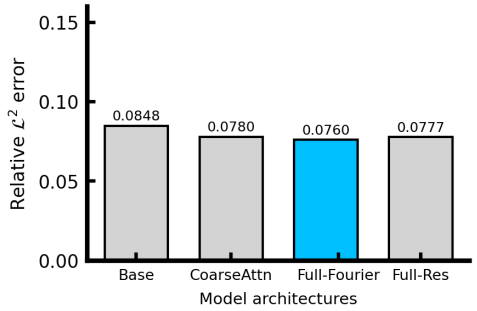
(a) Comparison of different models on 2D Navier-Stokes



(b) Ablation on different training strategies on 2D Navier-Stokes



(c) Comparison of different models on 3D smoke buoyancy



(d) Ablation on different architectural design on 2D Navier-Stokes

Figure 2: Quantitative study on model's performance. Latent-PDE denotes our proposed latent neural PDE solver. "Base" model contains only residual convolutional blocks and fully-connected layers. "CoarseAttn" means we add self-attention to the bottleneck part of the model. "Full-Fourier" means we add spectral convolution layers at the top two downsampling blocks in the encoder and decoder of "CoarseAttn" model. "Res" means we replace spectral convolution layer with residual convolutional blocks. x-step models are rollout for x steps during the training.

predicting the equilibriuim state of the smoke marker field (Figure 2c). (Sample visualization of the best model's prediction are presented in Appendix A)

We also study how different training strategies will influence the model's performance (Figure 2b). We maintain consistent hyperparameters and explore three training strategies: the two-stage method discussed in the previous subsection (referred to as "two-stage"), training the autoencoder and propagator simultaneously by minimizing both reconstruction and prediction loss jointly (referred to as "combined"), and considering the autoencoder and propagator as an unified entity to predict the subsequent step (referred to as "autoregressive"). We also compare two-stage training to Dilated-ResNet [74] that employs a Encode-Process-Decode (EPD) scheme [6, 61, 71] . We find that two-stage training yields the best performance compared to other strategies, which indicates the advantage of two-stage training in obtaining high-quality coarse-graining of the system.

	FNO2D	Latent-PDE 2D		FNO3D	Latent-PDE 3D	
		Autoencoder	Propagator		Autoencoder	Propagator
Fwd + Bwd time (sec)	0.067	0.103	0.013	2.223	1.375	0.372
Memory (GB)	1.87	2.54	0.25	33.33	37.15	8.10
# of params (M)	16.8	9.7	1.4	226.5	38.8	5.4

Table 1: Computational cost of different models' training. 2D benchmark is carried out on RTX 3090, using a batch size of 64. 3D benchmark is carried out on A6000, using a batch size of 16.

Compared to FNO that has log-linear complexity with respect to the grid size, the training of the proposed model is relatively slower when combining the time cost for autoencoder training and temporal model training. However, since the temporal model training is much more efficient in latent-pde solver, its training can be less costly on system that requires rolling out for more steps during training.

5 Conclusion

In this work, we study a straightforward yet effective data-driven framework for predicting time-dependent PDEs. We show that training the temporal model in the mesh-reduced space improves the computation efficiency and is beneficial for problems that feature latent dynamics distributed on a low-dimensional manifold. The observation in this study is also in alignment with the recent success of a series of image synthesis models that learn the generative model in the latent space instead of pixel space [13, 68, 83]. As this work only considers uniform mesh, an interesting future direction would be the extension to arbitrary meshes and geometries.

References

[1] Kaushik Bhattacharya, Bamdad Hosseini, Nikola B. Kovachki, and Andrew M. Stuart. Model reduction and neural networks for parametric pdes, 2021.

[2] Nicolas Boullé, Seick Kim, Tianyi Shi, and Alex Townsend. Learning green's functions associated with time-dependent partial differential equations. *Journal of Machine Learning Research*, 23(218):1–34, 2022.

[3] Johannes Brandstetter, Rianne van den Berg, Max Welling, and Jayesh K Gupta. Clifford neural layers for pde modeling. *arXiv preprint arXiv:2209.04934*, 2022.

[4] Johannes Brandstetter, Max Welling, and Daniel E Worrall. Lie point symmetry data augmentation for neural PDE solvers. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 2241–2256. PMLR, 17–23 Jul 2022. URL `https://proceedings.mlr.press/v162/brandstetter22a.html`.

[5] Johannes Brandstetter, Daniel Worrall, and Max Welling. Message passing neural pde solvers. *arXiv preprint arXiv:2202.03376*, 2022.

[6] Johannes Brandstetter, Daniel E. Worrall, and Max Welling. Message passing neural PDE solvers. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=vSix3HPYKSU`.

[7] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[8] Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin, and George Em Karniadakis. Physics-informed neural networks (pinns) for fluid mechanics: A review. *Acta Mechanica Sinica*, 37(12):1727–1738, 2021.

[9] Shuhao Cao. Choose a transformer: Fourier or galerkin. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 24924–24940. Curran Associates, Inc., 2021. URL `https://proceedings.neurips.cc/paper_files/paper/2021/file/d0921d442ee91b896ad95059d13df618-Paper.pdf`.

[10] Tianping Chen and Hong Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Transactions on Neural Networks*, 6(4):911–917, 1995. doi: 10.1109/72.392253.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[13] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis, 2021.

[14] Nicholas Geneva and Nicholas Zabaras. Transformers for modeling physical systems. *Neural Networks*, 146:272–289, 2022.

[15] John Guibas, Morteza Mardani, Zongyi Li, Andrew Tao, Anima Anandkumar, and Bryan Catanzaro. Adaptive fourier neural operators: Efficient token mixers for transformers. *arXiv preprint arXiv:2111.13587*, 2021.

[16] Ruchi Guo, Shuhao Cao, and Long Chen. Transformer meets boundary value inverse problems. *arXiv preprint arXiv:2209.14977*, 2022.

[17] Gaurav Gupta, Xiongye Xiao, and Paul Bogdan. Multiwavelet-based operator learning for differential equations, 2021.

[18] Jayesh K. Gupta and Johannes Brandstetter. Towards multi-spatiotemporal-scale generalized pde modeling, 2022.

[19] Jayesh K Gupta and Johannes Brandstetter. Towards multi-spatiotemporal-scale generalized pde modeling. *arXiv preprint arXiv:2209.15616*, 2022.

[20] Jiequn Han, Arnulf Jentzen, and Weinan E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34): 8505–8510, 2018.

[21] Xu Han, Han Gao, Tobias Pfaff, Jian-Xun Wang, and Li-Ping Liu. Predicting physics in mesh-reduced space with temporal attention. *arXiv preprint arXiv:2201.09113*, 2022.

[22] Zhongkai Hao, Chengyang Ying, Zhengyi Wang, Hang Su, Yinpeng Dong, Songming Liu, Ze Cheng, Jun Zhu, and Jian Song. Gnot: A general neural operator transformer for operator learning. *arXiv preprint arXiv:2302.14376*, 2023.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[24] AmirPouya Hemmasian and Amir Barati Farimani. Reduced-order modeling of fluid flows with transformers. *Physics of Fluids*, 35(5), 2023.

[25] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020.

[26] Philipp Holl, Vladlen Koltun, Kiwon Um, and Nils Thuerey. Phiflow: A Differentiable PDE Solving Framework for Deep Learning via Physical Simulations. In *NeurIPS Workshop*, 2020. URL `http://montrealrobotics.ca/diffcvgp/assets/papers/3.pdf`.

[27] Jun-Ting Hsieh, Shengjia Zhao, Stephan Eismann, Lucia Mirabella, and Stefano Ermon. Learning neural PDE solvers with convergence guarantees. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=rklaWn0qK7`.

[28] Steeven JANNY, Aurélien Bénéteau, Madiha Nadri, Julie Digne, Nicolas THOME, and Christian Wolf. EAGLE: Large-scale learning of turbulent fluid dynamics with mesh transformers. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=mfIX4QpsARJ`.

[29] Pengzhan Jin, Shuai Meng, and Lu Lu. Mionet: Learning multiple-input operators via tensor product. *SIAM Journal on Scientific Computing*, 44(6):A3490–A3514, 2022.

[30] 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.

[31] Georgios Kissas, Jacob Seidman, Leonardo Ferreira Guilhoto, Victor M. Preciado, George J. Pappas, and Paris Perdikaris. Learning operators with coupled attention, 2022.

[32] 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.

[33] Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Alexander Pritzel, Suman Ravuri, Timo Ewalds, Ferran Alet, Zach Eaton-Rosen, Weihua Hu, Alexander Merose, Stephan Hoyer, George Holland, Jacklynn Stott, Oriol Vinyals, Shakir Mohamed, and Peter Battaglia. Graphcast: Learning skillful medium-range global weather forecasting, 2022.

[34] Kookjin Lee and Kevin T. Carlberg. Deep conservation: A latent-dynamics model for exact satisfaction of physical conservation laws. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(1):277–285, May 2021. URL `https://ojs.aaai.org/index.php/AAAI/article/view/16102`.

[35] Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. Diffusion-lm improves controllable text generation. *Advances in Neural Information Processing Systems*, 35:4328–4343, 2022.

[36] Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B. Tenenbaum, and Antonio Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids, 2019.

[37] Yunzhu Li, Hao He, Jiajun Wu, Dina Katabi, and Antonio Torralba. Learning compositional koopman operators for model-based control. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=H1ldzA4tPr`.

[38] Zijie Li and Amir Barati Farimani. Graph neural network-accelerated lagrangian fluid simulation. *Computers & Graphics*, 103:201–211, 2022. ISSN 0097-8493. doi: https://doi.org/10.1016/j.cag.2022.02.004. URL `https://www.sciencedirect.com/science/article/pii/S0097849322000206`.

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

[40] Zijie Li, Kazem Meidani, and Amir Barati Farimani. Transformer for partial differential equations' operator learning. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL `https://openreview.net/forum?id=EPPqt3uERT`.

[41] Zijie Li, Dule Shu, and Amir Barati Farimani. Scalable transformer for pde surrogate modeling, 2023.

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

[43] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations, 2020.

[44] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Multipole graph neural operator for parametric partial differential equations, 2020.

[45] 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*, 2020.

[46] 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, 2023.

[47] Winfried Lötzsch, Simon Ohler, and Johannes Otterbach. Learning the solution operator of boundary value problems using graph neural networks. In *ICML 2022 2nd AI for Science Workshop*, 2022. URL https://openreview.net/forum?id=4vx9FQA7wiC.

[48] Lu Lu, Pengzhan Jin, and George Em Karniadakis. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.

[49] Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. Deepxde: A deep learning library for solving differential equations. *SIAM review*, 63(1):208–228, 2021.

[50] 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. *Computer Methods in Applied Mechanics and Engineering*, 393:114778, apr 2022. doi: 10.1016/j.cma.2022.114778. URL https://doi.org/10.1016%2Fj.cma.2022.114778.

[51] Bethany Lusch, J. Nathan Kutz, and Steven L. Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature Communications*, 9(1):4950, Nov 2018. ISSN 2041-1723. doi: 10.1038/s41467-018-07210-0. URL https://doi.org/10.1038/s41467-018-07210-0.

[52] Jeremy Morton, Antony Jameson, Mykel J Kochenderfer, and Freddie Witherden. Deep dynamical modeling and control of unsteady fluid flows. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/2b0aa0d9e30ea3a55fc271ced8364536-Paper.pdf.

[53] Tung Nguyen, Johannes Brandstetter, Ashish Kapoor, Jayesh K. Gupta, and Aditya Grover. Climax: A foundation model for weather and climate, 2023.

[54] Oded Ovadia, Adar Kahana, Panos Stinis, Eli Turkel, and George Em Karniadakis. Vito: Vision transformer-operator. *arXiv preprint arXiv:2303.08891*, 2023.

[55] 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. doi: 10.1137/19M1267246. URL https://doi.org/10.1137/19M1267246.

[56] Guofei Pang, Lu Lu, and George Em Karniadakis. fpinns: Fractional physics-informed neural networks. *SIAM Journal on Scientific Computing*, 41(4):A2603–A2626, 2019.

[57] Pranshu Pant, Ruchit Doshi, Pranav Bahl, and Amir Barati Farimani. Deep learning for reduced order modelling and efficient temporal evolution of fluid simulations. *Physics of Fluids*, 33(10):107101, 2021. doi: 10.1063/5.0062546. URL https://doi.org/10.1063/5.0062546.

[58] Pranshu Pant, Ruchit Doshi, Pranav Bahl, and Amir Barati Farimani. Deep learning for reduced order modelling and efficient temporal evolution of fluid simulations. *Physics of Fluids*, 33(10):107101, oct 2021. doi: 10.1063/5.0062546. URL https://doi.org/10.1063%2F5.0062546.

[59] Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, Pedram Hassanzadeh, Karthik Kashinath, and Animashree Anandkumar. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators, 2022.

[60] Saurabh Patil, Zijie Li, and Amir Barati Farimani. Hno: Hyena neural operator for solving pdes. *arXiv preprint arXiv:2306.16524*, 2023.

[61] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia. Learning mesh-based simulation with graph networks. 2020. doi: 10.48550/ARXIV.2010.03409. URL https://arxiv.org/abs/2010.03409.

[62] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia. Learning mesh-based simulation with graph networks, 2021.

[63] Lukas Prantl, Benjamin Ummenhofer, Vladlen Koltun, and Nils Thuerey. Guaranteed conservation of momentum for learning particle-based fluid dynamics, 2022.

[64] Md Ashiqur Rahman, Zachary E. Ross, and Kamyar Azizzadenesheli. U-no: U-shaped neural operators, 2023.

[65] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

[66] Stephan Rasp, Peter D. Dueben, Sebastian Scher, Jonathan A. Weyn, Soukayna Mouatadid, and Nils Thuerey. WeatherBench: A benchmark data set for data-driven weather forecasting. *Journal of Advances in Modeling Earth Systems*, 12(11), nov 2020. doi: 10.1029/2020ms002203. URL https://doi.org/10.1029%2F2020ms002203.

[67] Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 32, 2019.

[68] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[69] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022.

[70] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[71] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W. Battaglia. Learning to simulate complex physics with graph networks, 2020. URL https://arxiv.org/abs/2002.09405.

[72] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W. Battaglia. Learning to simulate complex physics with graph networks, 2020.

[73] Andrew Selle, Ronald Fedkiw, Byungmoon Kim, Yingjie Liu, and Jarek Rossignac. An unconditionally stable maccormack method. *Journal of Scientific Computing*, 35:350–371, 2008. URL https://api.semanticscholar.org/CorpusID:10522058.

[74] Kim Stachenfeld, Drummond Buschman Fielding, Dmitrii Kochkov, Miles Cranmer, Tobias Pfaff, Jonathan Godwin, Can Cui, Shirley Ho, Peter Battaglia, and Alvaro Sanchez-Gonzalez. Learned simulators for turbulence. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=msRBojTz-Nh.

[75] Kimberly Stachenfeld, Drummond B. Fielding, Dmitrii Kochkov, Miles Cranmer, Tobias Pfaff, Jonathan Godwin, Can Cui, Shirley Ho, Peter Battaglia, and Alvaro Sanchez-Gonzalez. Learned coarse models for efficient turbulence simulation, 2022.

[76] 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.

[77] Naoya Takeishi, Yoshinobu Kawahara, and Takehisa Yairi. Learning koopman invariant subspaces for dynamic mode decomposition. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 1130–1140, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

[78] Jingwei Tang, Vinicius C Azevedo, Guillaume Cordonnier, and Barbara Solenthaler. Neural green's function for laplacian systems. *Computers & Graphics*, 107:186–196, 2022.

[79] Nils Thuerey, Konstantin Weißenow, Lukas Prantl, and Xiangyu Hu. Deep learning methods for reynolds-averaged navier–stokes simulations of airfoil flows. *AIAA Journal*, 58(1):25–36, 2020.

[80] Alasdair Tran, Alexander Mathews, Lexing Xie, and Cheng Soon Ong. Factorized fourier neural operators, 2023.

[81] Kiwon Um, Robert Brand, Yun Raymond Fei, Philipp Holl, and Nils Thuerey. Solver-in-the-loop: Learning from differentiable physics to interact with iterative pde-solvers. *Advances in Neural Information Processing Systems*, 33:6111–6122, 2020.

[82] Benjamin Ummenhofer, Lukas Prantl, Nils Thuerey, and Vladlen Koltun. Lagrangian fluid simulation with continuous convolutions. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=B1lDoJSYDH.

[83] Arash Vahdat, Karsten Kreis, and Jan Kautz. Score-based generative modeling in latent space, 2021.

[84] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning, 2018.

[85] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[86] Rui Wang, Karthik Kashinath, Mustafa Mustafa, Adrian Albert, and Rose Yu. Towards physics-informed deep learning for turbulent flow prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, page 1457–1466, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450379984. doi: 10.1145/3394486.3403198. URL https://doi.org/10.1145/3394486.3403198.

[87] Rui Wang, Karthik Kashinath, Mustafa Mustafa, Adrian Albert, and Rose Yu. Towards physics-informed deep learning for turbulent flow prediction, 2020.

[88] Sifan Wang, Hanwen Wang, and Paris Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed deeponets. *Science Advances*, 7(40): eabi8605, 2021. doi: 10.1126/sciadv.abi8605. URL https://www.science.org/doi/abs/10.1126/sciadv.abi8605.

[89] Gege Wen, Zongyi Li, Kamyar Azizzadenesheli, Anima Anandkumar, and Sally M Benson. U-fno—an enhanced fourier neural operator-based deep-learning model for multiphase flow. *Advances in Water Resources*, 163:104180, 2022.

[90] Steffen Wiewel, Moritz Becher, and Nils Thürey. Latent space physics: Towards learning the temporal evolution of fluid flow. *Computer Graphics Forum*, 38, 2019.

[91] Yuxin Wu and Kaiming He. Group normalization, 2018.

[92] Xiaohui Zeng, Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, and Karsten Kreis. Lion: Latent point diffusion models for 3d shape generation. *arXiv preprint arXiv:2210.06978*, 2022.

[93] Min Zhu, Handi Zhang, Anran Jiao, George Em Karniadakis, and Lu Lu. Reliable extrapolation of deep neural operators informed by physics or sparse observations. *Computer Methods in Applied Mechanics and Engineering*, 412:116064, 2023.

A Sample visualization

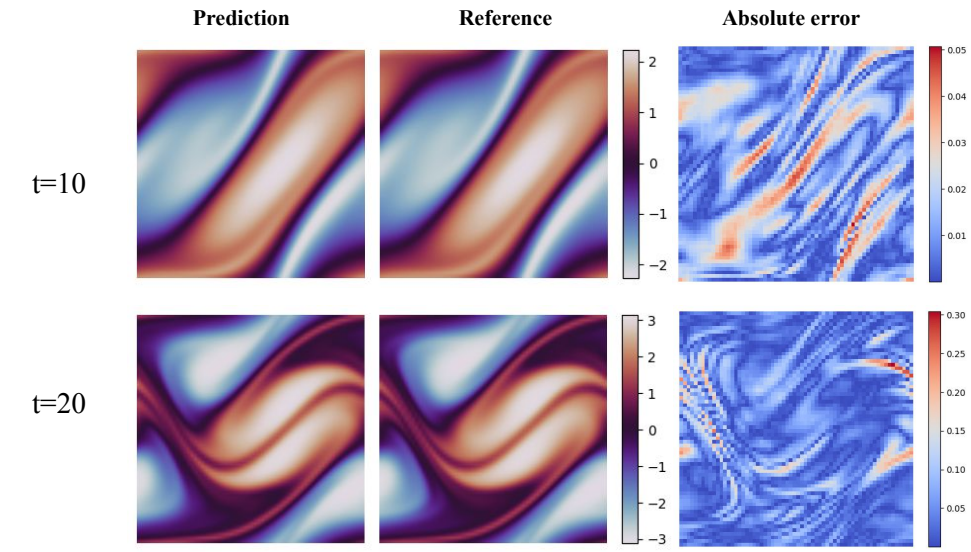


Figure 3: Visualization of model's prediction on 2D Navier-Stokes equation

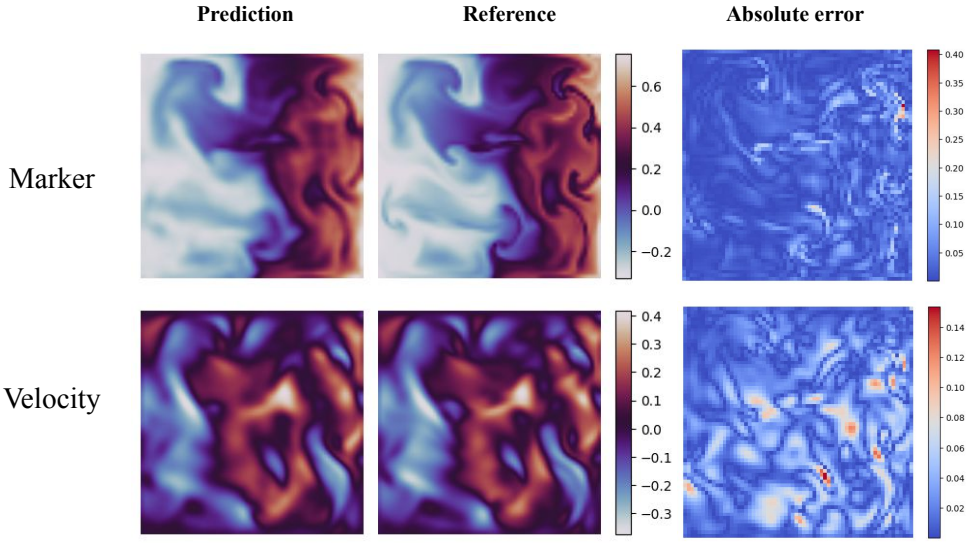


Figure 4: Visualization of model's prediction on 3D smoke buoyancy at cross-section plane $x = 4m$ and time $t = 9s$.