Tadpole: Autoencoders as Foundation Models for 3D PDEs with Online Learning

Qiang Liu[1], **Felix Koehler**[1], **Benjamin Holzschuh**[1], **Nils Thuerey**[1,2]
TUM School of Computation, Information and Technology[1]
Technical University of Munich
Garching, Germany 85748
MCML, Munich Center for Machine Learning[2]
Munich, Germany 80804
{qiang7.liu,nils.thuerey}@tum.de

Abstract

We introduce Tadpole, a novel foundation model for three-dimensional partial differential equations (PDEs) that addresses key challenges in transferability, scalability to high dimensionality, and multi-functionality. Tadpole is pre-trained as an autoencoder on synthetic 3D PDE data generated by an efficient online data-generation framework. This enables large-scale, diverse training without storage or I/O overhead, demonstrated by scaling to an equivalent of hundreds of terabytes of training data. By autoencoding single-channel spatial crops, Tadpole learns rich and transferable representations across heterogeneous physical systems with varying numbers of state variables and spatial resolutions. Although pre-trained solely as an autoencoder, Tadpole can be efficiently applied for multiple downstream tasks beyond reconstruction, including dynamics learning and generative modeling. For dynamics learning, we propose a novel parameter-efficient fine-tuning strategy that integrates low-rank adaptation, latent-space transformations, and reintroduced skip connections, achieving accurate temporal modeling with a minimal number of trainable parameters. Tadpole demonstrates strong fine-tuning performance across various downstream tasks, highlighting its versatility and effectiveness as a foundation model for 3D PDE learning. Source code and pre-trained weights of Tadpole are available at `https://tum-pbs.github.io/tadpole/`.

1 Introduction

The foundation model paradigm has achieved transformative success in Natural Language Processing (NLP) and Computer Vision (CV) [1, 2]. Recently, it has been adapted to scientific machine learning to solve Partial Differential Equations (PDEs) [3–6]. Unlike specialized solvers, these foundation models aim to learn transferable representations across diverse physical systems to allow efficient fine-tuning for new dynamics.

The prevailing strategy for building PDE foundation models is to learn the PDE dynamics by pre-training on large-scale trajectory datasets that capture a rich diversity of physical phenomena. These datasets are composed of numerous simulations, each representing a unique system defined by its governing equations, boundary conditions, and parameters (for instance, fluid viscosities or material stiffnesses). The model's task is to approximate the mapping from a system's past states to its future states. Formally, for a given phenomenon $\mathbb{P}^i$, the model learns to predict the state $\mathbf{u}^i_{t+\Delta t}$ from prior states $\mathbf{u}^i_{\leq t}$. The ambition is that by learning from many such examples, the model will distill universal physical principles, allowing it to *zero-shot* generalize to new phenomena and adapt to specific tasks with minimal additional *fine-tuning*.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Despite the attractiveness and potential of PDE foundation models, three fundamental challenges remain. First, there is a notable shortage of PDE foundation models for *three-dimensional* data. Most existing PDE foundation models focus on 1D or 2D problems, and a few notable exceptions that support 3D often rely on datasets that combine 3D with 2D/1D data [7, 8], or purely rely on 2D data [9]. Besides the heavily increased computational cost, a key reason for this lack of 3D models is the difficulty of collecting diverse and large-scale 3D PDE datasets for pre-training. Generating, storing, reading, and processing 3D data is significantly more expensive than 2D data, which fundamentally limits the diversity and scale of precomputed 3D PDE datasets. Since many real-world applications (e.g., weather forecasting, fluid dynamics, and material science) inherently involve 3D spatial domains, developing effective 3D PDE foundation models is crucial for advancing scientific machine learning.

In addition, transferability and generalization remain inconsistent. Ideally, most parameters of a foundation model can be reused without retraining, since the network should have learned general, transferable representations. For example, zero-shot evaluation and Parameter-Efficient Fine-Tuning (PEFT) have become standard benchmarks for the quality of NLP and CV foundation models [10–14]. However, most PDE foundation models still rely on full-parameter fine-tuning (FPFT), and preliminary zero-shot/PEFT experiments have shown limited success [15, 16, 7]. The reliance on FPFT casts doubt on the paradigm of training PDE foundation models: whether a model can really learn a generalizable representation through pre-training on PDE dynamics with extreme variability.

Finally, current PDE foundation models focus solely on dynamics learning, neglecting the ability to extend to other functionality. For example, generative modeling has emerged as a powerful paradigm in scientific machine learning [17–19]. Achieving *multi-functionality* across diverse downstream tasks, such as generative modeling, is a new challenge for PDE foundation models.

Therefore, developing a foundation model for 3D PDEs that efficiently and reliably generalizes across different tasks remains an open problem. Our work makes important steps to address these challenges with **Tadpole**, **t**hree-dimensional **a**utoenco**d**ers for **P**DEs with **o**nline **le**arning. It challenges the widespread notion that PDE foundation models require pretraining on the PDE dynamics of massive precomputed local data. We instead establish that foundation models can be trained for representation learning using simple, synthetic data generated on-the-fly during training. In contrast to foundation models in NLP, where representations emerge implicitly from next-token prediction, we achieve representation learning via autoencoding, explicitly optimizing a continuous latent space to capture the underlying data manifold. Our key innovations are:

- **A synthetic online learning framework:** We propose an efficient online learning framework with highly accurate, yet efficient GPU-based pseudo-spectral solvers and a novel buffer strategy, effectively bypassing I/O bottlenecks and storage limits at training time.
- **Transferable representations:** By pre-training Tadpole as an autoencoder on cropped individual fields, our models learn rich, transferable representations, enabling them to process varying PDE systems across different resolutions.
- **Efficient dynamics fine-tuning:** We propose a novel PEFT method for dynamics learning that integrates latent transformations, re-introduced skip connections, and LoRA [20] fine-tuning, which better utilizes the pre-trained representations and achieves high accuracy.
- **Multi-task versatility:** We demonstrate that Tadpole excels across different downstream tasks, including autoencoding, dynamics learning, and generative modeling, and at resolutions up to 1024^3 (i.e., on more than one billion degrees of freedom).

2 Related work

The potential of pre-trained neural networks to generalize across diverse physical systems was first characterized by Subramanian et al. [3]. Subsequent research has prioritized architectural scalability, from traditional U-Net structures [21] to modern vision-transformer (ViT) designs [22, 9, 16]. Poseidon [22] utilizes a multiscale transformer with time-conditioned layer norms to achieve continuous-in-time evaluations, while DPOT [9] scales to 1 billion parameters using a Fourier-attention-based architecture. Other similar works include MPP [15] and Walrus [8], the latter introducing compute-adaptive tokenization to maintain stability.

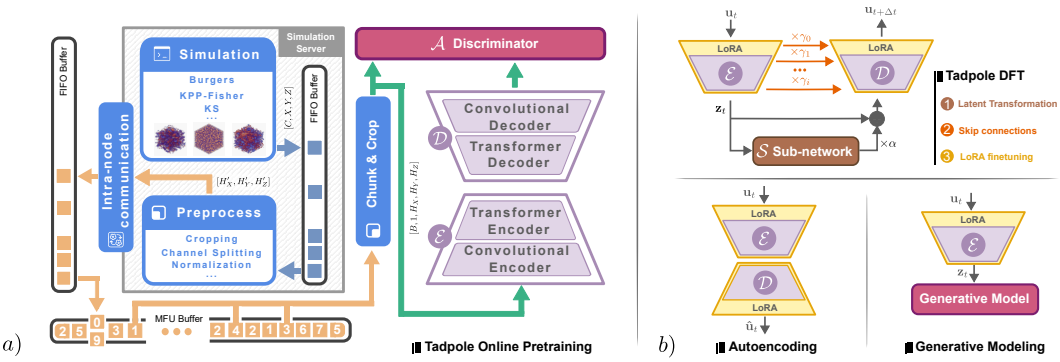

Figure 1: Overview of Tadpole: a) Tadpole is pre-trained as an autoencoder on single-channel crops of 3D PDE data generated on-the-fly by a GPU-based solver with an efficient buffer strategy. b) The pre-trained Tadpole can be used for various downstream tasks, including autoencoding, dynamics learning with the novel Tadpole-DFT method, and generative modeling via latent flow matching.

A central line of research focuses on the representation and embedding of heterogeneous PDE systems. Researchers have explored encoding PDEs as computational graphs to capture symbolic and numerical information simultaneously [23, 24], introducing point-wise deep conditions to guide the global attention of transformers [25], and utilizing SymPy-based libraries for automated symbolic tokenization [26]. To overcome the limitations of single-modality inputs, multimodal frameworks such as PROSE-PDE [27] and UPS [28] integrate numerical states with symbolic or textual descriptions [29, 30]. In addition, UPS [28] warm-starts from pre-trained Large Language Models (LLMs) to explicitly align data and improve computational efficiency.

Drawing inspiration from LLMs, recent studies investigated In-Context Learning (ICL) for PDE foundation models [31–33]. Zebra [34] and VICON [32] leverage prompt-based trajectories to solve parametric PDEs, while Liu et al. [35] utilize a block causal transformer to treat historical frames as contextual priors for next-frame prediction. Parallel to these ICL methods, PhysiX [36] utilizes discrete tokenization and autoregressive next-token prediction to model physical processes.

Beyond these themes, the field is advancing on several adjacent topics. PreLowD [37], MORPH [7] and OmniArch [38] have proposed lower dimensional pre-training. Frequency-adaptive fine-tuning was proposed [39], while constraint-aware pre-training [40] and physics-informed temporal alignment [41] incorporate PDE residuals to ensure physical consistency. A recent work [42] also pre-trains autoencoders for 2D PDEs, where the decoder is removed for dynamics fine-tuning, similar to previous latent-space learners [43, 44]. Finally, frontiers such as operator discovery [45, 46] and reward-model-driven reasoning [47] represent the latest efforts for scientific foundation models.

3 Self-supervised pre-training

3.1 Training objective

In traditional pre-training for PDE foundation models, the models learn the dynamics mapping a previous state $\mathbf{u}_t$ to a future state $\mathbf{u}_{t+\Delta t}$. In contrast, we pre-train Tadpole as an autoencoder that reconstructs $\mathbf{u}_t$ to learn rich, transferable spatial features of $\mathbf{u}_t$ itself. Specifically, Tadpole is pre-trained as a Variational Autoencoder (VAE) with an adversarial loss to encourage sharper reconstructions, following the success of representation learning paradigms in CV [48, 49]. Tadpole consists of an encoder $\mathcal{E}$ and a decoder $\mathcal{D}$. The encoder transforms the input $\mathbf{u}_t$ into a latent distribution $p_{\mathcal{E}}(\mathbf{z}_t|\mathbf{u}_t)$, while the decoder reconstructs the input from a sampled latent representation $\mathbf{z}_t$. A discriminator network $\mathcal{A}$ is optimized simultaneously to distinguish between real and reconstructed inputs and send feedback to the backbone training. Details of the pre-training target are provided in Appendix C.2. We choose reconstruction as the pre-training target over the dynamics target for the following reasons:

- In dynamics pre-training, a single $\mathbf{u}_t$ may evolve into significantly different future states depending on PDE type, boundary conditions, and physical parameters. This necessitates

high architectural complexity, as the network must distinguish between different physical systems by embedding a diverse set of parameters.

- The dynamics pre-training target can usually only be applied to dynamics downstream tasks. Instead, reconstruction pre-training will provide a meaningful latent space of the solution domain, enabling more diverse applications in different types of downstream tasks.
- Reconstruction only requires learning the low-dimensional manifold of admissible PDE solutions, which is often smooth and highly structured due to spatial correlations induced by differential operators. In contrast, predicting the mapping from $\mathbf{u}_t$ to $\mathbf{u}_{t+\Delta t}$ entails learning the nonlinear flow on this manifold. This is inherently more difficult to learn than reconstruction, as it must accurately capture both the geometry of the solution space and the vector field governing its evolution. A more detailed discussion can be found at Appendix A.1.

We show below that Tadpole, with its reconstruction-based pretraining, can be effectively fine-tuned for various downstream tasks (including the dynamics prediction) thanks to the generalization learned during pre-training. Details of fine-tuning methods will be discussed in Section 4.

3.2 Online learning framework

Another significant difference between Tadpole and other PDE foundation models is the efficient online learning [50–52] pipeline for pre-training, which (i) exposes the model to a diverse training distribution; (ii) and eliminates I/O and storage challenges induced by large-scale 3D PDE datasets while sustaining high-throughput training without stalling. An overview is shown in Fig. 1 a).

Data Generation: All pre-training data are generated on-the-fly using a PyTorch-based GPU solver, TorchFSM [53]. Spatial derivatives are computed via pseudo-spectral methods based on Fast Fourier Transforms (FFTs), and time integration is performed using Exponential Time Differencing Runge–Kutta (ETDRK) schemes [54, 55], providing a highly efficient simulation backbone. Details of the solver can be found in Appendix B.1.2. Although Fourier-spectral solvers impose periodic boundary conditions at the global domain level, Tadpole is trained exclusively on randomly sampled crops as will be discussed in Section 3.3. These crops correspond to local regions with non-periodic boundaries, thereby preventing the model from learning spurious periodic structures. Meanwhile, we also impose various initial conditions for different PDEs, further increasing data diversity (cf. Appendix B.1.3).

Buffering: To avoid the simulation speed affecting the training speed, we employ a three-stage buffering strategy. Simulation outputs are first written to a small First-In-First-Out (FIFO) buffer and asynchronously forwarded to the training processes. Each training process maintains a second FIFO buffer and a larger cache governed by a Most-Frequently-Used (MFU) replacement policy, from which batches are drawn directly during training. Background threads continuously replenish the MFU cache from newly arriving samples, effectively hiding simulation and communication latency behind training computation. The designed communication and buffer strategy can be effectively extended to multi-node HPC setups, and more details are provided in Appendix B.2.

3.3 Dataset structure

Another significant challenge in pre-training PDE foundation models is the variability in spatial resolutions and state variable counts across different systems, which complicates batching for diverse datasets. We address this in Tadpole by pre-training on single-channel crops of 3D PDE data. Specifically, given training data of shape $[B, C, X, Y, Z]$ where B is the batch size, C is the channel dimension representing the number of state variables, and X, Y, Z are spatial dimensions, we collapse the channel dimension into the batch dimension and randomly sample contiguous crops of shape $[B \times C, 1, H_X, H_Y, H_Z]$ for training. During inference, large domains are processed by encoding crops for each state variable in mini-batches, thereby avoiding the memory overhead of jointly processing all variables and spatial locations (cf. Appendix A.8). In this work, we set the crop sizes to $H_{X,Y,Z} = 64$ for pre-training. Meanwhile, to reduce data transfer volume and increase sample diversity, we apply an intermediate pre-cropping step before sending simulation data to the training process. Each simulation output is first cropped to an intermediate spatial size $H'_{X,Y,Z} = 96$, smaller than the full simulation resolution but larger than the final training crop size

$H_{X,Y,Z}$, enabling multiple distinct random crops to be drawn from the same transmitted sample. Notably, in conventional dynamics pre-training, learning from single-channel crops is challenging due to dynamics dependencies across variables (e.g., velocity and pressure) and error accumulation at crop boundaries during long rollouts. In contrast, Tadpole pre-training avoids these issues due to its reconstruction learning target.

3.4 Network architecture

We employ P3D [4], a state-of-the-art transformer for 3D PDEs, as the backbone for Tadpole. We adapt it to the reconstruction objective while preserving its core design principles. Specifically, we remove all embeddings related to PDE parameters, as the autoencoder focuses on input reconstruction rather than modeling complex dynamics. Furthermore, we eliminate the skip connections between the encoder and decoder to ensure that all information necessary for reconstruction is contained in the bottleneck latent space. An additional projection layer is appended to the encoder to map its output to the latent distribution parameters (mean and log-variance). Detailed architectural specifications are provided in Appendix C.1.

A primary motivation for choosing P3D, beyond its scalability for 3D data, is its hybrid architecture, which combines convolutional layers with a transformer-based bottleneck. This configuration leverages the translation equivariance of convolutions. Consequently, during inference, crops with different sizes than those used in training can be directly processed thanks to the convolutional layers. This design not only provides greater flexibility for processing data with different spatial resolutions but also lays a firm foundation for downstream dynamics fine-tuning, as discussed in the following sections. It is worth noting that the proposed training and fine-tuning strategy is not limited to the P3D architecture but also applies to other architectures with translation equivariance.

4 Flexible fine-tuning on downstream tasks

Below, we outline our fine-tuning methodology for dynamics, autoencoding, and generative modeling. Importantly, we will introduce Tadpole dynamics fine-tuning (*Tadpole-DFT*) methods for the dynamics task. An overview of the fine-tuning pipeline is shown in Fig. 1 b).

Dynamics Learning: With a pre-trained autoencoder like Tadpole, a natural approach for downstream dynamics tasks is to learn the PDE dynamics in the latent space rather than the high-dimensional physical space [43, 44]. However, since the latent representation $\mathbf{z}_t$ is significantly more compact than the original input $\mathbf{u}_t$, capturing precise dynamics purely in the latent space is often challenging. To address this, we propose the novel Tadpole-DFT approach, which encompasses:

- **Latent transformation:** Tadpole-DFT introduces a lightweight sub-network $\mathcal{S}$ between the pre-trained Tadpole encoder and decoder with a residual connection. As discussed in Section 3.3, capturing cross-variable interactions is crucial for dynamics learning. To solve this issue, we aggregate the latent spaces of all state variables after the encoder, enabling $\mathcal{S}$ to learn correlations between state variables.
- **Re-introduced skip connections:** During fine-tuning, the skip connections between the encoder and decoder are re-established, each governed by a trainable scale factor γ. This allows the model to leverage both the latent dynamics from the sub-network and high-resolution spatial information from the skip connections to predict the future state.
- **LoRA fine-tuning:** The pre-trained encoder $\mathcal{E}$ and decoder $\mathcal{D}$ are fine-tuned using LoRA [20] while their core weights remain frozen. For a pre-trained weight matrix $W_0 \in \mathbb{R}^{d\times k}$, LoRA approximates the update ΔW via a low-rank decomposition: $\Delta W = AB$, where $A \in \mathbb{R}^{d\times r}$, $B \in \mathbb{R}^{r\times k}$, and the rank $r \ll \min(d, k)$. During fine-tuning, the effective weight matrix is computed as $W = W_0 + AB$, where only the low-rank matrices A and B are trainable while W_0 remains frozen. The updated weights will be merged back into the original weights during inference to avoid additional computational overhead. LoRA enables the backbone to adapt to the new information flow from the skip connections and the sub-network $\mathcal{S}$ while preserving the robust representations acquired during pre-training.

In addition, to prevent error accumulation at crop boundaries during rollouts, we exploit the translation equivariance of the Tadpole encoder to construct latent representations from uncropped data. Despite

removing cropping, Tadpole-DFT remains easier to scale to larger datasets than conventional setups, as inputs are still encoded in mini-batches along the channel axis. Meanwhile, we zero-initialize the weights of the sub-network and backbone LoRA layers, as well as the scale factors for skip connections, which ensures that the model starts as a standard pre-trained autoencoder (outputting the current state $\mathbf{u}_t$) and gradually learns to transform $\mathbf{u}_t$ into the future state $\mathbf{u}_{t+\Delta t}$ during training, which further maintains the prior knowledge learned during pre-training.

Autoencoding: The pre-trained Tadpole model can be applied directly to unseen PDE systems for zero-shot autoencoding tasks. Alternatively, it can be fine-tuned on specific downstream systems to further enhance reconstruction quality, either by updating all model parameters or using LoRA to reduce the number of trainable parameters.

Generative Modeling: With a fine-tuned Tadpole, we can efficiently build a latent generative model [49] for 3D PDEs. For this, a new generative component is trained in the $\mathbf{z}_t$ space using a standard flow matching [56] objective. At inference time, new samples are drawn from the latent flow matching model and transformed into high-fidelity 3D PDE data by the Tadpole decoder $\mathcal{D}$. Operating in the compact latent space rather than the high-dimensional pixel space enables high-quality generative modeling of complex 3D physical systems while minimizing memory and processing requirements.

5 Experiments

In this section, we first summarize the pre-training statistics and then evaluate Tadpole across various downstream tasks that involve challenging 3D phenomena. We assess its zero-shot capabilities and fine-tuning performance, comparing it against state-of-the-art PDE foundation models.

5.1 Pre-training statistics

With the proposed online-learning framework, we pre-train Tadpole on 7 distinct PDE systems at four spatial resolutions (64^3, 128^3, 256^3, and 384^3). Details of the PDEs and corresponding configurations can be found in Appendix B.1.1. We perform a spectral distribution analysis on the pre-training dataset, illustrating its large spectral diversity, cf. Appendix B.1.4. Meanwhile, we also scale Tadpole models of varying sizes. Specifically, we investigate S, B, and L sizes of 8.8, 38.1, and 152.1 million parameters, with corresponding compression ratios of 16, 8, and 4. The pre-trained Tadpole achieves average reconstruction RMSEs of 5.48×10^{-3}, 2.83×10^{-3}, and 2.06×10^{-3}, for the S-, B- and L-size models across all pre-training PDEs. The online data generation pipeline yields approximately 202 TB of training data, with no local storage required. While naturally dependent on hardware specifics, the online training pipeline achieves an overall $1.8\times$ speedup over an offline setup with pre-generated data in our high-speed SSD environment. In HPC environments, which are typically used for FM training and have lower I/O bandwidths for data loading, we have measured differences that are an order of magnitude larger. Thus, the online training is especially effective in such cases.

5.2 Autoencoding

Let $D = C \times X \times Y \times Z$ denote the dimension of a 3D PDE state. To evaluate Tadpole's autoencoding performance for unseen data, we consider four representative 3D PDE systems: isotropic turbulence (`Iso`, $D = 4 \times 1024^3$), turbulent channel flow (`TCF`, $D = 3 \times 96^2 \times 192$), magnetohydrodynamics (`MHD`, $D = 10 \times 512^3$), and transitional boundary layer flows (`TBL`, $D = 4 \times 224^3$). These systems exhibit diverse physical characteristics and significantly higher resolutions that Tadpole has not encountered during pre-training. Details of these datasets are provided in Appendix B.3.

Zero-Shot Performance w.r.t Model and Dataset Sizes. We first evaluate Tadpole in zero-shot settings. The results indicate a clear scaling trend on model size: larger models consistently outperform smaller ones, as shown in Fig. 2 a). Besides, a comparison of models pre-trained on datasets with varying diversity is presented in Fig. 2 b). Three successively simpler online training setups are introduced: pre-training with only three PDEs (KS, Burgers, and KPP-Fisher), with one PDE (KS), or only with PDE initial conditions. The findings show that incorporating additional PDEs during pre-training continuously improves zero-shot performance. In contrast, the reduced performance of the pre-training solely with initial conditions suggests that the PDE dynamics generate novel features that enhance reconstruction generalization. Additionally, we also introduce a model pre-trained on

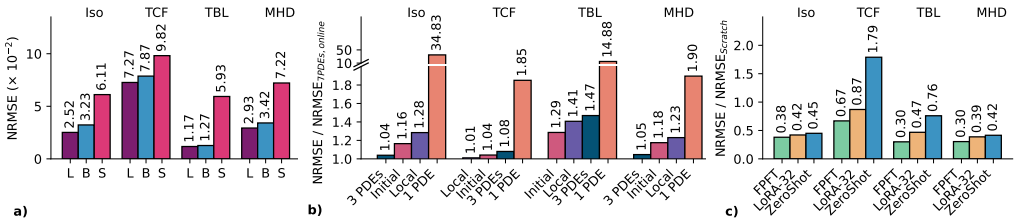

Figure 2: Performance of Tadpole on the downstream autoencoding task (exact NRMSE values in Table 2). a) Zero-shot reconstruction NRMSE of Tadpole with different model sizes. Tadpole shows consistent scaling with respect to model size. b) Zero-shot relative NRMSE of Tadpole-B models pre-trained on datasets with varying diversity compared to the full-size online setup (at 1.0). All variants perform worse, with relative NRMSE values larger than 1.0, demonstrating that greater PDE diversity improves reconstruction generalization. c) Relative NRMSE of Tadpole-B models with different fine-tuning methods compared to models trained from scratch. Pre-training consistently improves performance. E.g., LoRA-32 fine-tuning reduces errors by more than 60% for MHD compared to training from scratch.

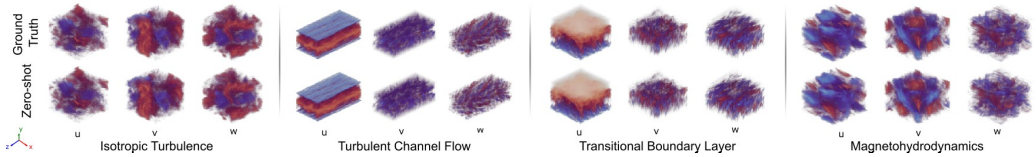

Figure 3: Visualizations of Tadpole-B zero-shot reconstruction on different datasets. Only velocity channels are shown here; additional ones are provided in Appendix A.9. The datasets feature high resolutions, ranging from $96^2 \times 192$ for `TCF` to 1024^3 for `Iso`.

a 500GB local dataset generated with the same PDEs and parameter distributions (cf. Table 13). The online pre-trained model outperforms the local variant, indicating that the online learning strategy further increases data diversity while eliminating the training I/O and data storage bottleneck. Visualizations of zero-shot reconstructions are shown in Fig. 3.

Fine-Tuning with Pre-trained Model: To verify the effect of pre-training, we fine-tune Tadpole under different settings and compare the performance with models trained from scratch. The results are summarized in Fig. 2 c). Pre-trained Tadpole shows substantial advantages, where both FPFT and LoRA-based PEFT consistently outperform their from-scratch counterparts. Notably, the zero-shot Tadpole B clearly surpasses from-scratch models on the `Iso`, `MHD`, and `TBL` tasks, which further highlights the effectiveness of pre-training.

Effect of LoRA Rank: Fig. 4 further illustrates the impact of the LoRA rank on fine-tuning performance. Even with a small LoRA rank, Tadpole achieves lower reconstruction error than models trained from scratch (NRMSE$= 7.17 \times 10^{-2}$), highlighting the efficacy of pre-trained representations. As the LoRA rank increases, performance continues to improve and approaches that of FPFT, while maintaining substantially fewer trainable parameters.

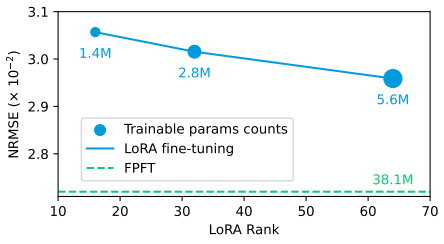

Figure 4: Reconstruction NRMSE of Tadpole-B with different LoRA ranks on `Iso` task. Larger rank shows better performance.

Flexibility in Domain Size and State Variable Count: The proposed crop-based strategy enables Tadpole to seamlessly handle varying domain sizes and state variable counts. In particular, for `Iso`, which features extremely high spatial resolutions, and `MHD`, which has a high state variable count, Tadpole effectively accommodates these variations without any architectural modifications or retraining. Meanwhile, although Tadpole is trained on crops with $H_{X,Y,Z} = 64$, we adopt different inference crop sizes for `TCF` ($H_{X,Y,Z} = 48$) and `TBL` ($H_{X,Y,Z} = 32$) to fully cover the spatial domains in these two cases, and Tadpole can still obtain

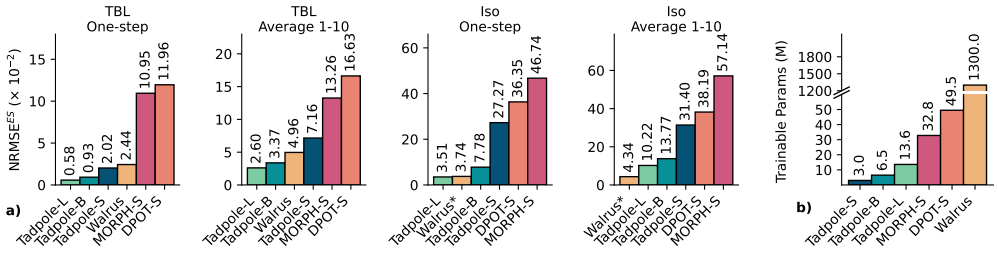

Figure 5: Performance of Tadpole on two distinct downstream dynamics tasks `TBL` and `Iso`. a) Prediction NRMSE^{ES} of different foundation models. Tadpole performs best on `TBL`, and ranks among the top-performing methods on `Iso`. b) The trainable parameters of different foundation models. Thanks to the LoRA introduced in the Tadpole-DFT method, only a very few parameters are fine-tuned in Tadpole, which makes it significantly smaller than the best-performing competitor, Walrus.

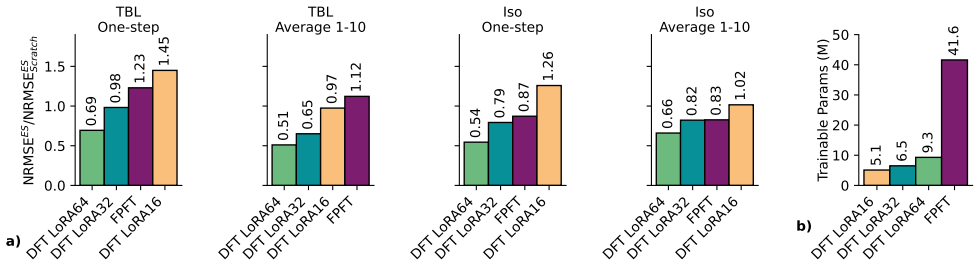

Figure 6: Performance improvements on the dynamics test from pre-training of Tadpole. a) Relative NRMSE^{ES} of Tadpole-B fine-tuned using various methods compared to the from-scratch variant. Increasing the LoRA rank in Tadpole-DFT consistently improves performance. b) Trainable parameters for different fine-tuning methods. The largest Tadpole-DFT variant utilizes only 22.3% of the trainable parameters required by the FPFT/from-scratch variant.

impressive zero-shot performance by leveraging the translation equivariance of convolutional layers. Furthermore, for spatial resolutions such as `TCF` with $96^2 \times 192$, the entire spatial domain can be processed in a single forward pass with similar performance (cf. Appendix A.3).

5.3 Dynamics learning

In this section, we evaluate Tadpole on a challenging dynamics learning task involving 3D cropped turbulent flows, following the setup of prior work [4]. The test contains an isotropic turbulence (`Iso`) and a turbulence boundary layer (`TBL`) simulation, both cropped from a larger domain, with 128^3 points. The cropping removes periodicity and introduces complex boundaries, thereby substantially increasing the difficulty of the dynamics learning task. We compare Tadpole against state-of-the-art 3D-PDE foundation models MORPH [7] and DPOT [9]. We select variants with total parameter counts comparable to those of the corresponding Tadpole models to ensure fair comparisons. Meanwhile, we also include a concurrent foundation model, Walrus [8], with a significantly larger parameter count than all the other models. All PDE foundation model baselines are performed via FPFT from their released pre-trained weights. For Tadpole variants, we employ Tadpole-DFT with LoRA rank $r = 32$ and a sub-network $\mathcal{S}$ with a standard encoder-only transformer architecture. Details of the $\mathcal{S}$ can be found in Appendix C.1. We also additionally compare Tadpole against several state-of-the-art network architectures trained from scratch in Appendix A.6. We utilize an enstrophy-based spectrum metric NRMSE^{ES} to accurately evaluate the rollout performance (cf. [57, 4], and Appendix D.1). Corresponding evaluation in pixel space is also provided in Appendix A.6.

Fig. 5 summarizes model performance for two dynamics tasks. Tadpole models exhibit meaningful scaling: larger models outperform smaller ones. The Tadpole models always perform better than the DPOT and MORPH models, even the smallest S model with 10x fewer trainable parameters. Importantly, in the `TBL` test, both the L and B size models outperform the Walrus model, which has

over two orders of magnitude more trainable parameters: Tadpole-B features a 10-step enstrophy error of 3.37, while the 200x larger Walrus model yields 4.97. It is worth noting that Walrus performs better on the `Iso` test case for longer rollout, which, however, is included in Walrus's training data (marked with subscript * in Fig. 5a)). This provides an advantage that the other models in this comparison do not have.

Fine-Tuning Methodologies: Fig. 6 presents a comparison between the Tadpole-DFT fine-tuning strategy, FPFT, and training from scratch. As the LoRA rank increases, Tadpole-DFT consistently demonstrates improved performance. Notably, the LoRA 32 configuration outperforms the from-scratch model across all evaluated metrics. Furthermore, the LoRA 64 variant achieves a 49% reduction in the 10-step average prediction error on the `TBL` dataset compared to the from-scratch approach, while utilizing 77.7% less trainable parameters. This result highlights the effectiveness of Tadpole-DFT in adapting the pre-trained Tadpole model to dynamics-learning tasks. In contrast, the FPFT variant of Tadpole-B does not consistently surpass the Tadpole-DFT variants, despite requiring substantially more trainable parameters. This behavior can be attributed to the fact that Tadpole is pre-trained as an autoencoder without explicit exposure to temporal dynamics; consequently, directly fine-tuning all parameters may lead to suboptimal local minima and unstable training. In contrast, Tadpole-DFT preserves the pre-trained representation by leveraging frozen weights in LoRA and incrementally incorporates dynamics learning through latent transformations and skip connections, thereby facilitating more effective learning of new dynamics features. Fig. 12 in the appendix presents the validation loss curves for Tadpole-B fine-tuned with Tadpole-DFT LoRA-32 and with FPFT. Tadpole-DFT not only achieves a lower final error but also converges faster and exhibits more stable training behavior than FPFT.

LoRA Rank vs Capacity of $\mathcal{S}$: Fig. 7 presents an ablation study on fine-tuning capacity for the dynamics learning task. We evaluate the Tadpole-DFT strategy under different configurations by varying the LoRA rank (LoRA 16 and LoRA 64) while keeping the sub-network size fixed, and by varying the sub-network size while fixing the LoRA rank. While increasing LoRA rank consistently improves performance, the size of the sub-network has little effect on performance. In Appendix A.6, we provide NRMSE values in physics space, where increasing the sub-network size slightly improves the prediction accuracy, still showing less effect than the LoRA rank. Thus, model performance benefits more strongly from LoRA rank than from the sub-network's capacity.

DFT Components: Our proposed Tadpole-DFT strategy consists of three major ingredients. To evaluate their individual efficacy, we conducted an ablation study with results presented in Fig. 8. We evaluate the performance of Tadpole-B fine-tuned with different Tadpole-DFT variants, including removing the latent transformation sub-network, removing the reintroduced skip connections, and freezing the backbone without LoRA fine-tuning. The results indicate that each component contributes meaningfully to overall performance, and removing any single component results in a noticeable increase in NRMSE^{ES}. Although previous analysis in Fig. 7 suggests that varying the sub-network size has a relatively limited impact on final performance, completely removing the sub-network results in a 4x increase in error. In addition, reintroducing skip connections incurs almost no increase in trainable parameters while improving performance by 68%, highlighting their effectiveness in enhancing information flow across scales during dynamics learning. Meanwhile, we also evaluated

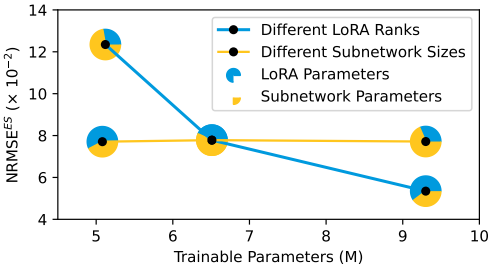

Figure 7: One-step NRMSE^{ES} of Tadpole-B model with different sub-network sizes and LoRA ranks. LoRA ranks positively affect performance.

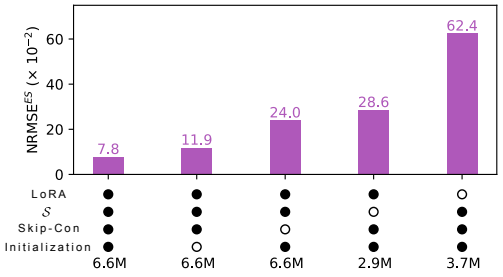

Figure 8: NRMSE^{ES} of Tadpole-B fine-tuned with different Tadpole-DFT components. All components contribute meaningfully to overall performance.

Table 1: Statistical evaluation of generated samples. **Bold** and underlined text shows the best and second-best values, respectively. Tadpole with FPFT fine-tuning performed best across all metrics.

Model	$\chi^2_{\mathrm{PQM}} \downarrow$	$\mathcal{W}_1 \downarrow$ $(\times 10^{-2})$	$\mathrm{MMD}_{\mathrm{RBF}} \downarrow$ $(\times 10^{-2})$	NRMSE $(\times 10^{-2}) \downarrow$		Rel. Time	Trainable Params
				Mean	Std.		
$\mathrm{UNet}_{\mathrm{GenCFD}}$	395.5	0.63	0.74	11.49	26.81	183.72	100.0M
AFNO	373.9	3.37	18.60	5.35	274.19	1.81	64.1M
AViT	557.78	36.53	19.45	12.57	1013.51	10.39	60.0M
Scratch	1775.7	7.08	25.32	40.11	37.30		
FPFT	**181.7**	**0.47**	**0.13**	**0.76**	**20.16**	**1.00**	**12.3M**
LoRA 32	256.5	0.81	0.47	2.28	32.06		

a pure latent dynamics variant, in which a network with the same architecture as $\mathcal{S}$ was trained to predict directly in the latent space encoded by the best-performing FPFT Tadpole encoder. The results are summarized in Tables 3 to 6 in the Appendix. This latent dynamics variant performed significantly worse than all Tadpole-DFT variants, despite having more trainable parameters. These results highlight the limitations of relying exclusively on the latent space for dynamic prediction and emphasize the necessity of utilizing all components of Tadpole-DFT.

5.4 Generative modeling

In this section, we evaluate Tadpole as a backbone for generative modeling of 3D turbulent flows. We focus on the `TCF` dataset, which exhibits complex, anisotropic flow structures. We implement a latent generative model based on flow matching, in which a 12.3M network with the same architecture as $\mathcal{S}$ is trained in the latent space defined by the Tadpole encoder to generate realistic 3D `TCF` fields. During inference, the latent flow matching model generates latent samples, which are subsequently decoded into high-fidelity 3D `TCF` fields using the Tadpole decoder. We consider three Tadpole-B variants: trained from scratch, fine-tuned with FPFT, and fine-tuned with LoRA-32 on the `TCF` autoencoding task, as described in Section 5.2. These Tadpole-based latent generative models are compared with baselines trained to generate samples directly in physical space, without leveraging pre-trained backbones. We introduce several metrics to evaluate the model performance, including the χ^2_{PQM} [58], Wasserstein-1 distance $\mathcal{W}_1$, Maximum Mean Discrepancy with a Radial Basis Function kernel $\mathrm{MMD}_{\mathrm{RBF}}$ [59], and the NRMSE of the mean and standard deviation of the distribution. Details of these metrics can be found in Appendix D.2. Meanwhile, we also evaluate the relative sample generation time (Rel. Time) normalized to the best-performing model. Fine-tuned Tadpole substantially improves generative modeling performance, as shown in Table 1. The latent generative model built upon the FPFT Tadpole achieves the best performance across all metrics. The LoRA 32 variant is the second-best model except for the $\mathcal{W}_1$ and the Std. metrics, where $\mathrm{UNet}_{\mathrm{GenCFD}}$ is better but 183 times slower. These results highlight the advantages and effectiveness of the latent representation learned with Tadpole's pre-training.

6 Conclusions, limitations and outlook

We have introduced Tadpole, a foundation model for 3D PDEs that leverages an efficient crop-based training strategy and a novel online pre-training framework using synthetic data generators. Tadpole is pre-trained as a variational autoencoder on a diverse set of 3D PDEs and can be effectively fine-tuned for various downstream tasks, including autoencoding, dynamics learning, and generative modeling. Extensive experiments demonstrate Tadpole's strong zero-shot reconstruction capabilities and its ability to achieve impressive performance across multiple tasks.

At the same time, several limitations exist: as the approach focuses on regular grids, unstructured grids are a natural extension. While our work, like other approaches, focuses on short-term rollouts, long-term predictions represent an important challenge for all scientific foundation models. Likewise, despite the high accuracy, even larger Tadpole models should be pre-trained and evaluated. In the future, it will be highly interesting to evaluate Tadpole's capacity to predict a broader range of physical systems [60], to couple with differentiable solvers [61–63], and to combine the framework with active learning techniques [64, 65].

Acknowledgements and disclosures

Qiang Liu acknowledges the support from the China Scholarship Council (No.202206120036) for his Ph.D. research at the Technical University of Munich. The authors gratefully acknowledge the scientific support and HPC resources provided by the Erlangen National High Performance Computing Center (NHR@FAU) of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) under the NHR project b278bb. NHR funding is provided by federal and Bavarian state authorities. The authors also gratefully acknowledge the computational and data resources as well as the support provided by the Leibniz Supercomputing Centre. The authors also acknowledge the EuroHPC Joint Undertaking for providing access to the EuroHPC supercomputer LEONARDO, hosted by CINECA (Italy) and the LEONARDO consortium. The authors also acknowledge support from the NVIDIA Academic Grant Program for using four RTX PRO 6000 GPUs.

References

[1] Devon Myers, Rami Mohawesh, Venkata Ishwarya Chellaboina, Anantha Lakshmi Sathvik, Praveen Venkatesh, Yi-Hui Ho, Hanna Henshaw, Muna Alhawawreh, David Berdik, and Yaser Jararweh. Foundation and large language models: fundamentals, challenges, opportunities, and social impacts. *Cluster Computing*, 27(1):1–26, 2024. URL https://link.springer.com/article/10.1007/s10586-023-04203-7.

[2] Muhammad Awais, Muzammal Naseer, Salman Khan, Rao Muhammad Anwer, Hisham Cholakkal, Mubarak Shah, Ming-Hsuan Yang, and Fahad Shahbaz Khan. Foundation Models Defining a New Era in Vision: A Survey and Outlook. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(4):2245–2264, 2025. doi: 10.1109/TPAMI.2024.3506283. URL https://ieeexplore.ieee.org/document/10834497/.

[3] Shashank Subramanian, Peter Harrington, Kurt Keutzer, Wahid Bhimji, Dmitriy Morozov, Michael W. Mahoney, and Amir Gholami. Towards foundation models for scientific machine learning: characterizing scaling and transfer behavior. In *Advances in Neural Information Processing Systems*, NeurIPS, Red Hook, NY, USA, 2023. Curran Associates Inc. URL https://arxiv.org/abs/2306.00258.

[4] Benjamin Holzschuh, Georg Kohl, Florian Redinger, and Nils Thuerey. P3D: Highly Scalable 3D Neural Surrogates for Physics Simulations with Global Context. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2509.10186.

[5] Yunjia Yang, Babak Gholami, Caglar Guerbuez, Mohammad Rashed, and Nils Thuerey. Toward a foundation-model paradigm for aerodynamic prediction in three-dimensional design. *AIAA Journal*, pages 1–21, 2026.

[6] Neil Ashton, Johannes Brandstetter, and Siddhartha Mishra. Fluid Intelligence: A Forward Look on AI Foundation Models in Computational Fluid Dynamics, 2025. URL https://arxiv.org/abs/2511.20455.

[7] Mahindra Singh Rautela, Alexander Most, Siddharth Mansingh, Bradley C. Love, Ayan Biswas, Diane Oyen, and Earl Lawrence. MORPH: PDE Foundation Models with Arbitrary Data Modality, 2025. URL https://arxiv.org/abs/2509.21670.

[8] Michael McCabe, Payel Mukhopadhyay, Tanya Marwah, Bruno Regaldo-Saint Blancard, Francois Rozet, Cristiana Diaconu, Lucas Meyer, Kaze W. K. Wong, Hadi Sotoudeh, Alberto Bietti, Irina Espejo, Rio Fear, Siavash Golkar, Tom Hehir, Keiya Hirashima, Geraud Krawezik, Francois Lanusse, Rudy Morel, Ruben Ohana, Liam Parker, Mariel Pettee, Jeff Shen, Kyunghyun Cho, Miles Cranmer, and Shirley Ho. Walrus: A Cross-Domain Foundation Model for Continuum Dynamics, 2025. URL https://arxiv.org/abs/2511.15684.

[9] Zhongkai Hao, Chang Su, Songming Liu, Julius Berner, Chengyang Ying, Hang Su, Anima Anandkumar, Jian Song, and Jun Zhu. DPOT: auto-regressive denoising operator transformer for large-scale PDE pre-training. In *International Conference on Machine Learning*, ICML'24. JMLR.org, 2024. URL https://dl.acm.org/doi/10.5555/3692070.3692773.

[10] Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, Jing Yi, Weilin Zhao, Xiaozhi Wang, Zhiyuan Liu, Hai-Tao Zheng, Jianfei Chen, Yang Liu, Jie Tang, Juanzi Li, and Maosong Sun. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5 (3):220–235, Mar 2023. ISSN 2522-5839. doi: 10.1038/s42256-023-00626-4. URL https://doi.org/10.1038/s42256-023-00626-4.

[11] Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. Parameter-Efficient Fine-Tuning for Large Models: A Comprehensive Survey. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=lIsCS8b6zj.

[12] Yi Xin, Jianjiang Yang, Siqi Luo, Yuntao Du, Qi Qin, Kangrui Cen, Yangfan He, Zhiwei Zhang, Bin Fu, Xiaokang Yang, Guangtao Zhai, Ming-Hsuan Yang, and Xiaohong Liu. Parameter-Efficient Fine-Tuning for Pre-Trained Vision Models: A Survey and Benchmark, 2025. URL https://arxiv.org/abs/2402.02242.

[13] Dan Zhang, Tao Feng, Lilong Xue, Yuandong Wang, Yuxiao Dong, and Jie Tang. Parameter-Efficient Fine-Tuning for Foundation Models, 2025. URL https://arxiv.org/abs/2501.13787.

[14] Yu Meng, Jiaxin Huang, Yu Zhang, and Jiawei Han. Generating Training Data with Language Models: Towards Zero-Shot Language Understanding. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 462–477. Curran Associates, Inc., 2022. URL https://dl.acm.org/doi/10.5555/3600270.3600304.

[15] Michael McCabe, Bruno Régaldo-Saint Blancard, Liam Parker, Ruben Ohana, Miles Cranmer, Alberto Bietti, Michael Eickenberg, Siavash Golkar, Geraud Krawezik, Francois Lanusse, Mariel Pettee, Tiberiu Tesileanu, Kyunghyun Cho, and Shirley Ho. Multiple physics pretraining for spatiotemporal surrogate models. In *Advances in Neural Information Processing Systems*, NeurIPS, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9798331314385. URL https://neurips.cc/virtual/2024/poster/96095.

[16] Benjamin Holzschuh, Qiang Liu, Georg Kohl, and Nils Thuerey. PDE-transformer: Efficient and versatile transformers for physics simulations. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 23562–23602. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/holzschuh25a.html.

[17] Qiang Liu and Nils Thuerey. Uncertainty-Aware Surrogate Models for Airfoil Flow Simulations with Denoising Diffusion Probabilistic Models. *AIAA Journal*, 62(8):2912–2933, 2024. doi: 10.2514/1.J063440. URL https://doi.org/10.2514/1.J063440.

[18] Salva Rühling Cachay, Bo Zhao, Hailey Joren, and Rose Yu. Dyffusion: A dynamics-informed diffusion model for spatiotemporal forecasting. *Advances in Neural Information Processing Systems*, 36, 2024. URL https://arxiv.org/abs/2306.01984.

[19] Christian Jacobsen, Yilin Zhuang, and Karthik Duraisamy. CoCoGen: Physically Consistent and Conditioned Score-Based Generative Models for Forward and Inverse Problems. *SIAM Journal on Scientific Computing*, 47(2):C399–C425, 2025. doi: 10.1137/24M1636071. URL https://epubs.siam.org/doi/10.1137/24M1636071.

[20] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

[21] Abu Bucker Siddik, Diane Oyen, Alexander Most, Michal Kucer, and Ayan Biswas. SPUS: A Lightweight and Parameter-Efficient Foundation Model for PDEs, 2025. URL https://arxiv.org/abs/2510.01370.

[22] Maximilian Herde, Bogdan Raonić, Tobias Rohner, Roger Käppeli, Roberto Molinaro, Emmanuel de Bézenac, and Siddhartha Mishra. POSEIDON: efficient foundation models for PDEs. In *Advances in Neural Information Processing Systems*, NeurIPS, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9798331314385. URL https://dl.acm.org/doi/10.5555/3737916.3740227.

[23] Zhanhong Ye, Xiang Huang, Leheng Chen, Hongsheng Liu, Zidong Wang, and Bin Dong. PDEformer: Towards a foundation model for one-dimensional partial differential equations. In *ICLR 2024 Workshop on AI4DifferentialEquations In Science*, 2024. URL https://openreview.net/forum?id=GLDMCwdhTK.

[24] Zhanhong Ye, Zining Liu, Bingyang Wu, Hongjie Jiang, Leheng Chen, Minyan Zhang, Xiang Huang, Qinghe Meng, Jingyuan Zou, Hongsheng Liu, and Bin Dong. PDEformer-2: A Versatile Foundation Model for Two-Dimensional Partial Differential Equations, 2025. URL https://arxiv.org/abs/2507.15409.

[25] Hang Zhou, Yuezhou Ma, Haixu Wu, Haowen Wang, and Mingsheng Long. Unisolver: PDE-conditional transformers are universal neural PDE solvers. In *ICLR 2025 Workshop on Foundation Models in the Wild*, 2025. URL https://openreview.net/forum?id=6HlgUqkjmW.

[26] Derek Jollie, Jingmin Sun, Zecheng Zhang, and Hayden Schaeffer. Time-Series Forecasting, Knowledge Distillation, and Refinement within a Multimodal PDE Foundation Model, 2024. URL https://arxiv.org/abs/2409.11609.

[27] Jingmin Sun, Yuxuan Liu, Zecheng Zhang, and Hayden Schaeffer. Towards a foundation model for partial differential equations: Multioperator learning and extrapolation. *Phys. Rev. E*, 111: 035304, Mar 2025. doi: 10.1103/PhysRevE.111.035304. URL https://link.aps.org/doi/10.1103/PhysRevE.111.035304.

[28] Junhong Shen, Tanya Marwah, and Ameet Talwalkar. UPS: efficiently building foundation models for PDE solving via cross-modal adaptation. *Trans. Mach. Learn. Res.*, 2024, 2024. URL https://openreview.net/forum?id=0r9mhjRv1E.

[29] Florian Wiesner, Matthias Wessling, and Stephen Baek. Towards a Physics Foundation Model, 2025. URL https://arxiv.org/abs/2509.13805.

[30] Elisa Negrini, Yuxuan Liu, Liu Yang, Stanley J. Osher, and Hayden Schaeffer. A Multimodal PDE Foundation Model for Prediction and Scientific Text Descriptions, 2025. URL https://arxiv.org/abs/2502.06026.

[31] Liu Yang, Siting Liu, Tingwei Meng, and Stanley J. Osher. In-context operator learning with data prompts for differential equation problems. *Proceedings of the National Academy of Sciences*, 120(39):e2310142120, 2023. doi: 10.1073/pnas.2310142120. URL https://www.pnas.org/doi/abs/10.1073/pnas.2310142120.

[32] Yadi Cao, Yuxuan Liu, Liu Yang, Rose Yu, Hayden Schaeffer, and Stanley Osher. VICON: Vision In-Context Operator Networks for Multi-Physics Fluid Dynamics Prediction, 2025. URL https://arxiv.org/abs/2411.16063.

[33] Zezheng Song, Jiaxin Yuan, and Haizhao Yang. FMint: Bridging Human Designed and Data Pretrained Models for Differential Equation Foundation Model, 2024. URL https://arxiv.org/abs/2404.14688.

[34] Louis Serrano, Armand Kassaï Koupaï, Thomas X Wang, Pierre Erbacher, and Patrick Gallinari. Zebra: In-Context Generative Pretraining for Solving Parametric PDEs. In *International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=22kNOkkokU.

[35] Yuxuan Liu, Jingmin Sun, and Hayden Schaeffer. BCAT: A Block Causal Transformer for PDE Foundation Models for Fluid Dynamics, 2025. URL https://arxiv.org/abs/2501.18972.

[36] Tung Nguyen, Arsh Koneru, Shufan Li, and Aditya Grover. PhysiX: A Foundation Model for Physics Simulations, 2025. URL https://arxiv.org/abs/2506.17774.

[37] AmirPouya Hemmasian and Amir Barati Farimani. Pretraining a neural operator in lower dimensions. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=ZewaRoZehI.

[38] Tianyu Chen, Haoyi Zhou, Ying Li, Hao Wang, Chonghan Gao, Rongye Shi, Shanghang Zhang, and Jianxin Li. OmniArch: Building Foundation Model for Scientific Computing. In *International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=UlprLwWYKP.

[39] Hangwei Zhang, Chun Kang, Yan Wang, and Difan Zou. F-Adapter: Frequency-Adaptive Parameter-Efficient Fine-Tuning in Scientific Machine Learning. *Advances in Neural Information Processing Systems*, 2025. URL https://neurips.cc/virtual/2025/poster/119790.

[40] Amin Totounferoush, Serge Kotchourko, Michael W. Mahoney, and Steffen Staab. Paving the way for scientific foundation models: enhancing generalization and robustness in PDEs with constraint-aware pre-training, 2025. URL https://arxiv.org/abs/2503.19081.

[41] Congcong Zhu, Xiaoyan Xu, Jiayue Han, and Jingrun Chen. Physics-informed Temporal Alignment for Auto-regressive PDE foundation models. In *International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=OKDN1Hg3im.

[42] Anthony Zhou and Amir Barati Farimani. Masked Autoencoders are PDE learners. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=rZNuiFwXVs.

[43] S. Wiewel, M. Becher, and N. Thuerey. Latent Space Physics: Towards Learning the Temporal Evolution of Fluid Flow. *Computer Graphics Forum*, 38(2):71–82, 2019. doi: https://doi.org/10.1111/cgf.13620. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13620.

[44] Francesco Regazzoni, Stefano Pagani, Matteo Salvador, Luca Dede', and Alfio Quarteroni. Learning the intrinsic dynamics of spatio-temporal processes through Latent Dynamics Networks. *Nature Communications*, 15(1):1834, Feb 2024. ISSN 2041-1723. doi: 10.1038/s41467-024-45323-x. URL https://doi.org/10.1038/s41467-024-45323-x.

[45] Md Ashiqur Rahman, Robert Joseph George, Mogab Elleithy, Daniel Leibovici, Zongyi Li, Boris Bonev, Colin White, Julius Berner, Raymond A Yeh, Jean Kossaifi, et al. Pretraining codomain attention neural operators for solving multiphysics pdes. *Advances in Neural Information Processing Systems*, 37:104035–104064, 2024. URL https://neurips.cc/virtual/2024/poster/93155.

[46] Rudy Morel, Jiequn Han, and Edouard Oyallon. DISCO: learning to DISCover an evolution operator for multi-physics-agnostic prediction. In *International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=6EZ3MDDf6p.

[47] Siddharth Mansingh, James Amarel, Ragib Arnab, Arvind Mohan, Kamaljeet Singh, Gerd J. Kunde, Nicolas Hengartner, Benjamin Migliori, Emily Casleton, Nathan A. Debardeleben, Ayan Biswas, Diane Oyen, and Earl Lawrence. Towards Reasoning for PDE Foundation Models: A Reward-Model-Driven Inference-Time-Scaling Algorithm, 2025. URL https://arxiv.org/abs/2509.02846.

[48] Patrick Esser, Robin Rombach, and Björn Ommer. Taming Transformers for High-Resolution Image Synthesis. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12868–12878, 2021. doi: 10.1109/CVPR46437.2021.01268. URL https://openaccess.thecvf.com/content/CVPR2021/html/Esser_Taming_Transformers_for_High-Resolution_Image_Synthesis_CVPR_2021_paper.html.

[49] 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. URL https://www.computer.org/csdl/proceedings-article/cvpr/2022/694600k0674/1H1iFsO7Zuw.

[50] Steven C.H. Hoi, Doyen Sahoo, Jing Lu, and Peilin Zhao. Online learning: A comprehensive survey. *Neurocomputing*, 459:249–289, 2021. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2021.04.112. URL https://www.sciencedirect.com/science/article/pii/S0925231221006706.

[51] Lucas Meyer, Marc Schouler, Robert A. Caulk, Alejandro Ribes, and Bruno Raffin. Training deep surrogate models with large scale online learning. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023. URL https://dl.acm.org/doi/10.5555/3618408.3619432.

[52] Théophile Terraz, Alejandro Ribes, Yvan Fournier, Bertrand Iooss, and Bruno Raffin. Melissa: large scale in transit sensitivity analysis avoiding intermediate files. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '17, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450351140. doi: 10.1145/3126908.3126922. URL https://doi.org/10.1145/3126908.3126922.

[53] Qiang Liu, Felix Koehler, and Nils Thuerey. TorchFSM: Fourier Spectral Method with PyTorch, May 2025. URL https://qiauil.github.io/torchfsm/.

[54] S.M. Cox and P.C. Matthews. Exponential Time Differencing for Stiff Systems. *Journal of Computational Physics*, 176(2):430–455, 2002. ISSN 0021-9991. doi: https://doi.org/10.1006/jcph.2002.6995. URL https://www.sciencedirect.com/science/article/pii/S0021999102969950.

[55] Felix Koehler, Simon Niedermayr, Rüdiger Westermann, and Nils Thuerey. APEBench: A Benchmark for Autoregressive Neural Emulators of PDEs. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 120252–120310. Curran Associates, Inc., 2024. doi: 10.52202/079017-3822. URL https://tum-pbs.github.io/apebench-paper/.

[56] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow Matching for Generative Modeling. In *The Eleventh International Conference on Learning Representations* , 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

[57] Yifan Chen, Mark Goldstein, Mengjian Hua, Michael Samuel Albergo, Nicholas Matthew Boffi, and Eric Vanden-Eijnden. Probabilistic Forecasting with Stochastic Interpolants and Föllmer Processes. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 6728–6756. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/chen24n.html.

[58] Pablo Lemos, Sammy Nasser Sharief, Nikolay Malkin, Salma Salhi, Connor Stone, Laurence Perreault Levasseur, and Yashar Hezaveh. PQMass: Probabilistic Assessment of the Quality of Generative Models using Probability Mass Estimation. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=n7qGCmluZr.

[59] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander J. Smola. A Kernel Two-Sample Test. *J. Mach. Learn. Res.*, 13:723–773, 2012. doi: 10.5555/2503308.2188410. URL https://dl.acm.org/doi/10.5555/2503308.2188410.

[60] Ruben Ohana, Michael McCabe, Lucas Meyer, Rudy Morel, Fruzsina Agocs, Miguel Beneitez, Marsha Berger, Blakesly Burkhart, Stuart Dalziel, Drummond Fielding, et al. The well: a large-scale collection of diverse physics simulations for machine learning. *Advances in Neural Information Processing Systems*, 37:44989–45037, 2024. URL https://neurips.cc/virtual/2024/poster/97882.

[61] Kiwon Um, Robert Brand, Philipp Holl, Raymond Fei, and Nils Thuerey. Solver-in-the-loop: Learning from differentiable physics to interact with iterative pde-solvers. 2020. URL https://ge.in.tum.de/publications/2020-um-solver-in-the-loop/.

[62] N. Thuerey, B. Holzschuh, P. Holl, G. Kohl, M. Lino, Q. Liu, P. Schnell, and F. Trost. *Physics-based Deep Learning*. WWW, 2021. URL https://physicsbaseddeeplearning.org.

[63] Giacomo Baldan, Qiang Liu, Alberto Guardone, and Nils Thuerey. Physics vs distributions: Pareto optimal flow matching with physics constraints. *International Conference on Learning Representations (ICLR)*, 2026.

[64] Daniel Musekamp, Marimuthu Kalimuthu, David Holzmüller, Makoto Takamoto, and Mathias Niepert. Active Learning for Neural PDE solvers. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=x4ZmQaumRg.

[65] Raphaël Pestourie, Youssef Mroueh, Thanh V Nguyen, Payel Das, and Steven G Johnson. Active learning of deep surrogates for PDEs: application to metasurface design. *npj Computational Materials*, 6(1):164, 2020. URL https://www.nature.com/articles/s41524-020-00431-2.

[66] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, Furu Wei, and Baining Guo. Swin Transformer V2: Scaling Up Capacity and Resolution. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11999–12009, 2022. doi: 10.1109/CVPR52688.2022.01170. URL https://ieeexplore.ieee.org/document/9879380.

[67] John Guibas, Morteza Mardani, Zongyi Li, Andrew Tao, Anima Anandkumar, and Bryan Catanzaro. Efficient Token Mixing for Transformers via Adaptive Fourier Neural Operators. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2111.13587.

[68] Aly-Khan Kassam and Lloyd N. Trefethen. Fourth-Order Time-Stepping for Stiff PDEs. *SIAM Journal on Scientific Computing*, 26(4):1214–1233, 2005. doi: 10.1137/S1064827502410633. URL https://doi.org/10.1137/S1064827502410633.

[69] 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.

[70] Aleksandra Franz, Hao Wei, Luca Guastoni, and Nils Thuerey. PICT–A differentiable, GPU-accelerated multi-block PISO solver for simulation-coupled learning tasks in fluid dynamics. *Journal of Computational Physics*, 544:114433, 2026. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2025.114433. URL https://www.sciencedirect.com/science/article/pii/S0021999125007156.

[71] Jae Hyun Lim and Jong Chul Ye. Geometric GAN, 2017. URL https://arxiv.org/abs/1705.02894.

[72] Dustin Tran, Rajesh Ranganath, and David M. Blei. Hierarchical implicit models and likelihood-free variational inference. In *Advances in Neural Information Processing Systems*, NeurIPS, page 5529–5539, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

[73] Takeru Miyato and Masanori Koyama. cGANs with projection discriminator. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=ByS1VpgRZ.

[74] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Technical appendices and supplementary material

This appendix provides details and background information for Tadpole on the following topics:

- Appendix A: Additional Analysis, Results, and Visualizations.
- Appendix B: Dataset and online training setups
- Appendix C: Training details and network architectures
- Appendix D: Evaluation metrics
- Appendix E: Nomenclature and abbreviations
- Appendix F: Changelog

A Additional analysis, results, and visualizations

A.1 On the geometric complexity of reconstruction and dynamics learning

Below, we explain why the reconstruction objective is easier to learn compared to the dynamics objective from a manifold point of view.

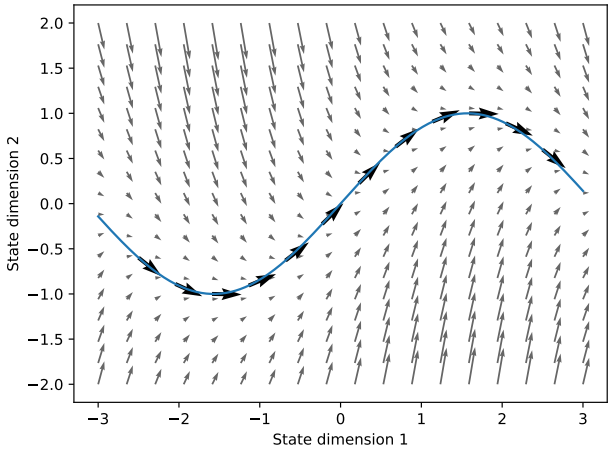

Figure 9: Illustration of the distinction between learning the solution manifold and learning the induced dynamics. The curve represents a low-dimensional solution manifold $\mathcal{M}$ embedded in a higher-dimensional space. Tangent vectors along the manifold correspond to the intrinsic dynamics $F(u) \in T_u\mathcal{M}$, while the surrounding vector field illustrates the additional requirement of maintaining consistency with the manifold by attracting perturbed states back toward M. Reconstruction-based learning only needs to learn the geometry of $\mathcal{M}$, whereas dynamics learning also requires learning both tangent and surrounding vectors.

Let $\mathcal{H}$ and $\mathcal{M} \subset \mathcal{H}$ denote a high-dimensional function space and a set of admissible PDE solution states u, respectively. $\mathcal{M}$ is typically concentrated near a low-dimensional, smooth manifold due to spatial coupling induced by differential operators. Training with the reconstruction target is equivalent to learning a coordinate chart and its inverse on $\mathcal{M}$:

$$\mathcal{E} : \mathcal{M} \to \mathbb{R}^k, \ \mathcal{D} : \mathbb{R}^k \to \mathcal{M}, \ \mathcal{D} \circ \mathcal{E} \approx \mathrm{Id}_{\mathcal{M}}, \tag{1}$$

where $\mathbb{R}^k$ denotes a k-dimensional Euclidean latent space. The above learning target is primarily governed by the geometric features, e.g., dimensionality and regularity, of the solution manifold $\mathcal{M}$ rather than the high-dimensional function space $\mathcal{H}$. In particular, when $\mathcal{M}$ is a smooth manifold, the associated coordinate maps can also be chosen to be smooth, and their neural approximations, i.e., $\mathcal{E}$ and $\mathcal{D}$, can be small and simple.

In contrast, the PDE evolution can be treated as a dynamical system on $\mathcal{M}$:

$$\frac{du}{dt} = F(u), \tag{2}$$

where $F(u) \in T_u\mathcal{M}$ is a tangent vector assigned to each state $u \in \mathcal{M}$. Learning a one-step prediction (i.e., $\mathbf{u}_t \to \mathbf{u}_{t+\Delta t}$) corresponds to learning a flow map generated through integrating F, which is more challenging for many reasons:

- While reconstruction requires approximating the geometry of $\mathcal{M}$, dynamical prediction also requires learning additional vector fields defined on $\mathcal{M}$ besides the manifold's geometry.
- In addition to approximating the tangent vector F on $\mathcal{M}$, a learned dynamics model must remain stable under perturbations, e.g., it should also learn the surrounding vectors, as shown in Fig. 9, to map states in a neighborhood of M back toward M, thereby preserving the manifold's invariance.
- Although $\mathcal{M}$ can be smooth and low-dimensional, the vector field F and corresponding surrounding vector fields could still exhibit strong variability, particularly when PDEs have strong nonlinear interactions or multiscale effects.

Consequently, reconstruction-based objectives primarily learn the geometric structure of the solution manifold, whereas dynamical models must additionally resolve the induced flow on this manifold. This distinction helps explain why reconstruction-based training is typically easier and exhibits better generalization properties than directly learning PDEs time-stepping.

A.2 Methodological Summary of the Tadpole approach

The discussion of the advantages of reconstruction-based objectives for general *representation learning* highlights key advantages of the proposed approach.

To summarize, Tadpoles distinguishes itself from existing approaches for scientific foundation models in the following ways:

- It focuses on *autoencoding* as generalizable, central objective for represetnation learning.
- Tadpole employs *online data-generation* with a fast, semi-spectral, GPU-based solver, circumventing storage and I/O bottlenecks.
- It comes with a highly *flexible architecture*, e.g., supporting arbitrary numbers of channels, and temporal dynamics (via learned skip-connections) for downstream tasks.
- Tadpole's capabilities are demonstrated for a wide range of downstream applications, from reconstruction, over generative modeling to temporal dynamics.

These aspects also provide important distinctions from existing approaches for foundation models, in particular from large language models: the representations of LLMs typically emerge implicitly from the next-token prediction task using a pre-determined tokenization codebook. Tadpole instead explicitly optimizes for representation learning via autoencoding. Specifically, we induce meaningful representations by combining the reconstruction task with large streams of synthetic PDE data, such that the resulting latent space captures the low-dimensional solution manifold and allows for generalizing transfers to new downstream tasks. This is more closely aligned with the learned latent spaces of imaging and video FMs, which, in contrast, typically focus on perceptually-driven latent spaces.

A.3 Comparison between crop-based inference and whole-domain inference

We compare the zero-shot reconstruction RMSE of the Tadpole B-size model across `TCF` datasets for crop-based and whole-domain inference. For the S-size model, the reconstruction error slightly improved by 2.8% with whole-domain inference. For the B-size model, the performance degrades slightly by 0.8%. Fig. 10 shows the corresponding visualizations of the reconstructions with whole-domain inference and crop-based inference, and Fig. 11 shows the corresponding absolute error. We can see that the whole-domain approach helps to remove the inconsistency near crop boundaries in the S-model zero-shot. But in general, the zero-shot reconstruction RMSE on the `TCF` dataset is not significantly affected by the inference strategy, highlighting Tadpole's strong generalization across different resolutions.

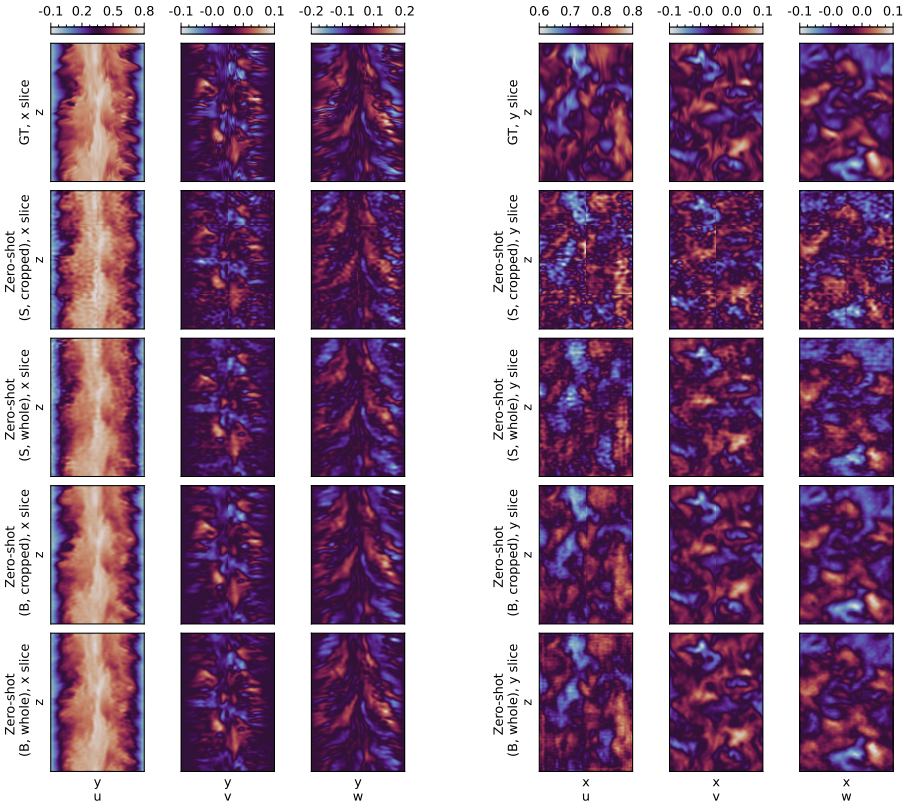

Figure 10: Visualization of the reconstruction of `TCF` with crop-based and whole-domain inference for slices $x = X/2$ (left), and $y = Y/2$ (right).

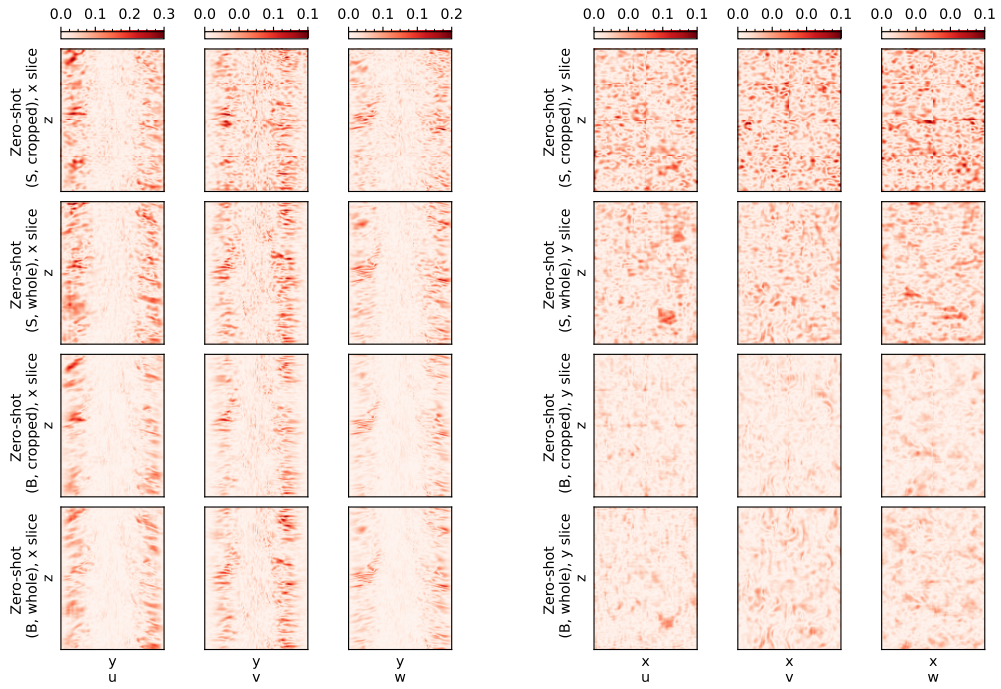

Figure 11: Visualization of the absolute error for the reconstruction of TCF with crop-based and whole-domain inference for slices $x = X/2$ (left), and $y = Y/2$ (right).

A.4 Convergence curve of FPFT and LoRA fine-tuning on dynamics learning

Fig. 12 shows the validation RMSE during FPFT and LoRA fine-tuning. FPFT fine-tuning exhibits many oscillations during training, especially at the start, whereas LoRA fine-tuning remains stable. This highlights LoRA fine-tuning's ability to preserve pretrained knowledge, thereby avoiding training instabilities.

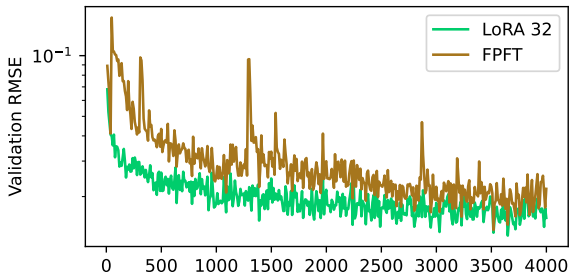

Figure 12: Validation RMSE of Tadpole fine-tuned with LoRA-32 and FPFT. LoRA yields stable training with reduced errors.

A.5 Ablation study of dynamics learning in pixel space

In Fig. 7 and Fig. 8, we perform ablation studies in spectrum space, showing that the sub-network has little effect on model performance. Here, we provide additional evaluations in the pixel space. Fig. 13 and Fig. 14 compare the model performance with different sub-network sizes, LoRA ranks, and DFT components. Compared to the results in spectrum space, the effect of the sub-network is more evident in physics space, where improving the sub-network size clearly decreases the NRMSE of the prediction, although increasing the LoRA rank is more efficient. Dropping the sub-network results in the largest increase in pixel-space error compared to removing other components.

The above results suggest that LoRA has a greater impact on performance in the spectrum space, while the sub-network has a greater impact in the pixel space. This is because LoRA works with the model backbone, where convolutional and vision transformer layers are applied to learn spatial correlations. Improving backbone performance yields a better-learned spatial pattern of PDE solutions. While the sub-network learns in the latent space, spatial correlations are more difficult to learn because of the backbone's downsampling layers. It focuses more on improving model performance by combining information from state variables. Thus, although it can improve performance in pixel space, the spatial patterns of PDE solutions are only slightly affected.

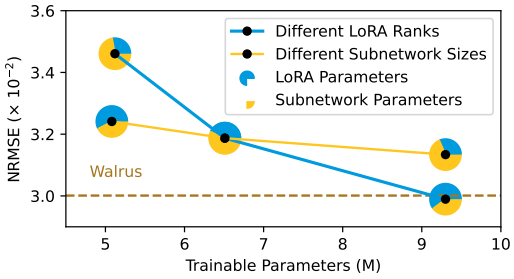

Figure 13: One-step NRMSE of the Tadpole-B model with different sub-network sizes and LoRA ranks. Especially the latter positively affects performance.

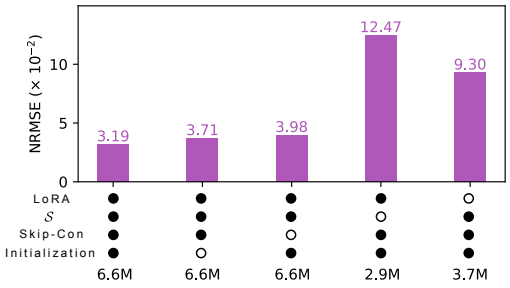

Figure 14: One-step NRMSE of Tadpole-B fine-tuned with different Tadpole-DFT components.

A.6 Detailed metric values of the main experiments

In this section, we summarize the detailed metric values from the previous experiments. Table 2 shows the reconstruction NRMSE of Tadpole models on the autoencoding task. Corresponding results are illustrated in Fig. 2. Table 3-Table 6 summarize the NRMSE^{ES} and NRMSE of different methods in dynamics tasks. Corresponding results are illustrated in Fig. 5. Here, we also compare the model's performance with Swin3D [66], AViT [15], and AFNO [67], all trained from scratch, where Tadpole and other foundation models outperform them in $NRMSE^{ES}$.

Table 2: Tadpole's reconstruction NRMSE ($\times 10^{-2}$) on downstream autoencoding tasks. The zero-shot model surpasses a from-scratch model across three datasets, highlighting the efficacy of pre-training. Fine-tuning further enhances the performance.

			Iso	TCF	TBL	MHD	Trainable Params
Zero Shot	B	S	6.11	9.82	5.93	7.22	\
		B	3.23	7.87	1.27	3.42	
		L	2.52	7.27	1.17	<u>2.93</u>	
		1 PDE	112.36	14.57	18.88	6.48	
		3 PDEs	3.35	8.50	1.86	3.57	
		Initial	3.75	8.20	1.63	4.04	
		Local	4.14	7.96	1.79	4.21	
Scratch	B		7.17	4.40	1.67	8.22	<u>38.1M</u>
FPFT			**2.73**	**2.94**	**0.50**	**2.50**	<u>38.1M</u>
LoRA-32			<u>3.01</u>	<u>3.82</u>	<u>0.78</u>	3.18	**2.8M**

Table 3: Rollout NRMSE^{ES} ($\times 10^{-2}$) of different models on the dynamics downstream task for TBL dataset. Tadpole with DFT fine-tuning strategy outperforms all other specialized and foundational models.

Rollout steps	1	2	3	4	5	6	7	8	9	10	Average	Trainable Params (M)
Swin3D	1.21	1.96	2.49	2.87	3.16	3.36	3.64	3.95	4.28	4.67	3.16	50.3M
AViT	7.99	8.47	9.21	10.00	10.77	11.56	12.36	12.89	13.64	14.21	11.11	60.0M
AFNO	3.62	4.95	6.53	8.19	9.82	11.19	12.24	13.17	14.02	14.78	9.85	64.1M
MorphNet-S	10.95	11.89	12.36	12.98	13.22	13.35	13.83	14.29	14.69	15.04	13.26	32.8M
DPOT-S	11.96	13.60	14.80	15.92	16.64	17.20	18.01	18.70	19.41	20.05	16.63	49.5M
Walrus	2.44	2.92	2.97	3.05	3.68	4.45	5.55	6.76	8.20	9.58	4.96	1300.0M
Tadpole-B-Scratch	0.94	1.83	2.71	3.58	4.48	5.37	6.42	7.62	8.80	10.05	5.18	41.6M
Tadpole-B-FPFT	1.16	2.24	3.31	4.35	5.40	6.36	7.36	8.37	9.31	10.27	5.81	41.6M
Latent Dynamics	51.84	65.74	69.72	71.16	71.69	71.94	72.00	72.39	72.26	72.51	69.12	7.4M
Tadpole-S-LoRA-32	2.02	3.47	4.69	5.85	6.92	7.86	8.85	9.80	10.65	11.44	7.16	**3.0M**
Tadpole-B-LoRA-16	1.37	2.38	3.27	4.14	4.91	5.57	6.28	6.93	7.53	8.09	5.05	5.1M
Tadpole-B-LoRA-32	0.93	1.64	2.28	2.89	3.39	3.76	4.19	4.56	4.89	5.18	3.37	6.5M
Tadpole-B-LoRA-64	0.65	1.19	1.68	2.17	2.59	**2.94**	3.32	3.67	3.97	4.25	2.64	9.3M
Tadpole-L-LoRA-32	**0.58**	**1.10**	**1.62**	**2.13**	**2.57**	2.94	**3.31**	**3.65**	**3.93**	**4.21**	**2.60**	13.6M

Table 4: Rollout NRMSE ($\times 10^{-2}$) of different models on the dynamics downstream task for TBL dataset. Tadpole with DFT fine-tuning strategy outperforms Walrus in one-step prediction with much fewer trainable parameters.

Rollout steps	1	2	3	4	5	6	7	8	9	10	Average	Trainable Params (M)
Swin3D	0.40	1.16	**1.73**	**2.44**	**3.11**	**3.74**	**4.30**	**4.82**	**5.36**	**5.86**	**3.29**	50.3M
AViT	0.87	1.58	2.39	3.16	3.90	4.61	5.24	5.83	6.41	6.94	4.09	60.0M
AFNO	0.52	1.20	2.01	2.76	3.50	4.20	4.83	5.42	6.00	6.54	3.70	64.1M
MorphNet-S	0.95	1.55	2.26	2.96	3.66	4.33	4.94	5.48	6.02	6.52	3.87	32.8M
DPOT-S	1.01	1.60	2.31	3.00	3.70	4.38	5.00	5.58	6.17	6.71	3.95	49.5M
Walrus	0.49	1.09	1.80	2.49	3.17	3.84	4.43	4.98	5.57	6.05	3.39	1300.0M
Tadpole-B-Scratch	0.49	1.30	2.41	3.62	4.91	6.20	7.44	8.67	9.87	11.04	5.59	41.6M
Tadpole-B-FPFT	0.57	1.42	2.61	3.78	4.91	5.97	6.93	7.83	8.70	9.54	5.22	41.6M
Latent Dynamics	11.30	15.48	18.54	20.17	21.64	22.83	23.64	23.89	24.52	24.79	20.68	7.4M
Tadpole-S-LoRA-32	0.55	1.35	2.27	3.14	3.99	4.81	5.58	6.32	7.06	7.79	4.29	**3.0M**
Tadpole-B-LoRA-16	0.46	1.17	2.07	2.93	3.79	4.62	5.39	6.13	6.86	7.58	4.10	5.1M
Tadpole-B-LoRA-32	0.41	1.11	2.05	2.95	3.81	4.61	5.32	5.99	6.65	7.27	4.02	6.5M
Tadpole-B-LoRA-64	0.38	1.05	1.90	2.70	3.46	4.18	4.85	5.47	6.09	6.69	3.68	9.3M
Tadpole-L-LoRA-32	**0.34**	**1.01**	1.90	2.78	3.62	4.43	5.18	5.87	6.54	7.19	3.89	13.6M

Table 5: Rollout NRMSE^{ES} ($\times 10^{-2}$) of different models on the dynamics downstream task for `Iso` dataset. Besides the Walrus with two orders of magnitude more parameters, Tadpole with DFT fine-tuning strategy outperforms all other specialized and foundational models.

Rollout steps	1	2	3	4	5	6	7	8	9	10	Average	Trainable Params (M)
Swin3D	30.23	37.46	40.14	41.65	42.69	43.45	44.13	45.01	45.40	46.19	41.64	50.3M
AViT	36.57	49.76	73.28	102.54	135.52	171.18	208.46	243.67	281.19	309.62	161.18	60.0M
AFNO	25.45	23.27	24.87	27.62	30.80	33.93	36.84	38.69	41.24	42.17	32.49	64.1M
MorphNet-S	46.74	50.61	52.99	55.04	56.91	58.64	60.31	62.04	63.38	64.74	57.14	32.8M
DPOT-S	36.35	36.96	37.27	37.69	38.06	38.39	38.71	39.22	39.40	39.86	38.19	49.5M
Walrus	3.74	**4.04**	**4.04**	**4.02**	**4.09**	**4.32**	**4.29**	**4.61**	**5.01**	**5.28**	**4.34**	1300.0M
Tadpole-B-Scratch	9.81	12.32	13.94	15.52	16.93	18.14	19.18	20.10	20.63	20.94	16.75	41.6M
Tadpole-B-FPFT	8.55	10.26	11.47	12.80	14.02	15.01	15.87	16.54	16.95	16.93	13.84	41.6M
Latent Dynamics	61.15	63.74	65.81	67.77	69.56	71.15	72.69	74.23	75.44	76.69	69.82	7.4M
Tadpole-S-LoRA-32	27.27	30.25	31.07	31.48	31.70	31.88	32.14	32.61	32.65	32.91	31.40	**3.0M**
Tadpole-B-LoRA-16	12.35	14.44	15.36	16.23	16.97	17.65	18.27	19.05	19.53	20.26	17.01	5.1M
Tadpole-B-LoRA-32	7.78	9.60	10.94	12.35	13.65	14.83	15.89	16.90	17.59	18.18	13.77	6.5M
Tadpole-B-LoRA-64	5.35	6.82	8.07	9.40	10.62	11.80	12.98	14.21	15.08	16.22	11.05	9.3M
Tadpole-L-LoRA-32	**3.51**	5.14	6.77	8.40	9.93	11.32	12.57	13.80	14.84	15.92	10.22	13.6M

Table 6: Rollout NRMSE ($\times 10^{-2}$) of different models on the dynamics downstream task for `Iso` dataset. Tadpole outperforms the Walrus in one-step prediction

Rollout steps	1	2	3	4	5	6	7	8	9	10	Average	Trainable Params (M)
Swin3D	5.53	9.26	11.97	14.07	15.76	17.21	18.50	19.70	**20.64**	**21.70**	15.43	50.3M
AViT	51.54	52.70	54.33	56.22	58.39	60.95	63.73	66.91	70.44	74.13	60.93	60.0M
AFNO	6.06	10.19	13.41	16.13	18.55	20.80	22.95	25.01	26.89	28.93	18.89	64.1M
MorphNet-S	9.37	13.00	16.63	20.25	23.85	27.43	30.83	34.15	37.35	40.46	25.33	32.8M
DPOT-S	6.14	8.36	10.85	13.41	16.05	18.74	21.41	24.06	26.67	29.36	17.50	49.5M
Walrus	3.00	**4.67**	**6.68**	**8.92**	**11.41**	**13.99**	**16.65**	**19.59**	22.46	25.42	**13.28**	1300.0M
Tadpole-B-Scratch	4.19	7.98	11.92	15.87	19.89	24.06	28.31	32.80	37.43	42.53	22.50	41.6M
Tadpole-B-FPFT	3.66	6.94	10.47	14.11	17.90	21.89	26.04	30.47	35.05	40.01	20.65	41.6M
Latent Dynamics	10.30	13.88	17.72	21.47	25.14	28.75	32.19	35.59	38.93	42.32	26.63	7.4M
Tadpole-S-LoRA-32	4.41	7.28	10.64	14.36	18.48	22.99	27.88	33.31	39.09	45.66	22.41	**3.0M**
Tadpole-B-LoRA-16	3.46	6.26	9.53	13.17	17.11	21.29	25.62	30.10	34.69	39.59	20.08	5.1M
Tadpole-B-LoRA-32	3.19	5.84	8.94	12.35	16.04	19.98	24.07	28.34	32.71	37.41	18.89	6.5M
Tadpole-B-LoRA-64	2.99	5.54	8.54	11.85	15.47	19.38	23.50	27.88	32.36	37.14	18.47	9.3M
Tadpole-L-LoRA-32	**2.85**	5.29	8.07	11.06	14.23	17.59	21.04	24.58	28.19	32.08	16.50	13.6M

A.7 Pre-training gains of other baselines

The current from-scratch version of Tadpole already demonstrates strong advantages over pre-trained DPOT and Morph models, indicating that the P3D network itself serves as a strong baseline. To distinguish the performance gains attributable to pre-training and network architecture, Tables 7 and 8 compare the NRMSE^{ES} and NRMSE for DPOT and Morph trained from scratch with their respective fine-tuned models in the `Iso` case. DPOT achieves average improvements of 2% and 14% in frequency and physics space, respectively, while Morph achieves improvements of 5% and 15%. For Tadpole, the corresponding improvements are 18% in frequency space and 16% in physics space for a LoRA 32 configuration (Fig. 6 and Tables 5 and 6). Increasing the LoRA rank to 64 further increases the improvement in frequency space to 34%. These gains are achieved with 6.5 million (LoRA 32) or 9.3 million (LoRA 64) trainable parameters, compared to the full network's 41.6 million. Overall, pre-training and fine-tuning the Tadpole models result in substantially greater improvements, particularly in the frequency domain, even when starting from the stronger P3D baseline.

Table 7: Rollout NRMSE$^{ES}(\times 10^{-2})$ of DPOT and Morph on the `Iso` dynamics downstream task.

Rollout Step	1	2	5	10	Average 1–10
DPOT (Scratch)	36.58	36.97	38.86	41.53	39.03
DPOT (Finetuned)	36.35	36.96	38.06	39.86	38.19 (-2%)
Morph (Scratch)	55.52	58.22	60.69	63.68	60.57
Morph (Finetuned)	46.74	50.61	56.91	64.74	57.14 (-5%)

Table 8: Rollout direct NRMSE$(\times 10^{-2})$ of DPOT and Morph on the `Iso` dynamics downstream task.

Rollout Step	1	2	5	10	Average 1–10
DPOT (Scratch)	7.13	9.71	18.56	34.22	20.32
DPOT (Finetuned)	6.14	8.36	16.05	29.36	17.50 (-14%)
Morph (Scratch)	10.96	15.31	27.44	47.74	29.92
Morph (Finetuned)	9.37	13.00	23.85	40.46	25.33 (-15%)

The results for DPOT and MORPH trained from scratch yield important insights into their respective architectures. When considered alongside the previous comparison between fine-tuned Tadpole and from-scratch Tadpole (Fig. 6 and Tables 3 to 6), these findings clarify the contributions of each component in Tadpole. Specifically, a portion of the absolute performance can be attributed to the backbone, while the controlled same-architecture comparison isolates the additional benefits provided by pretrained representations and the fine-tuning method, which account for up to 49% improvement in LoRA-64 DFT fine-tuning in the TBL case.

A.8 Computation Scaling Relative to Data Size

In this section, we evaluate the computation scaling of Tadpole models with data size. All the experiments are performed on an NVIDIA RTX PRO 6000 Blackwell GPU.

Table 9 summarizes the changes in inference computational cost and peak VRAM consumption of Tadpole-B as the dimension of the 3D PDE state $D = C \times X \times Y \times Z$ increases. In this experiment, the process crop size is fixed at 64^3, and Tadpole processes the data in a single batch, where the batch size equals $D/64^3$. The results demonstrate that inference cost increases linearly with the PDE state D.

However, since Tadpole incorporates the channel and crop dimensions into the batch dimension, crops can be processed in internal mini-batches. Table 10 presents the changes in calculation time and peak memory usage as a function of internal minibatch size when $D/64^3 = 256$. The internal minibatch operation does not change the total number of floating-point operations, but it significantly reduces peak memory usage, with only a slight increase in computation time.

Table 9: Computational cost and peak allocated memory for different PDE state dimensions D.

$D/64^3$	GFLOPs	Peak allocated memory (GB)
1	10.336	0.339
2	20.672	0.483
4	41.345	0.772
8	82.689	1.349
16	165.379	2.503
32	330.758	4.811
64	661.516	9.429
128	1,323.032	18.664
256	2,646.063	37.131

Table 10: Trade-off between computation time and memory usage for internal minibatch inference when $D/64^3 = 256$. An internal batch size of 1 indicates that Tadpole processes a single crop at a time, requiring 256 batches to process the entire dataset. In contrast, an internal batch size of 256 indicates that Tadpole processes all data in a single internal batch.

Internal batch size	Median time (s)	Peak allocated memory (GB)
1	4.169	0.945
2	3.521	0.980
4	3.491	1.263
8	3.159	1.834
16	4.142	2.971
32	3.288	4.799
64	3.036	8.048
128	3.027	12.854
256	2.891	37.131

A.9 Visualizations of autoencoding reconstructions

Figures 15–37 present qualitative reconstruction results for different datasets under various training strategies. Due to the high spatial resolution of `Iso` and `MHD`, differences in the reconstructions are difficult to discern at lower visualization resolutions. We therefore additionally provide visualizations of 128^3 crops for clearer comparison. Overall, models trained from scratch exhibit higher reconstruction errors than the other variants. The zero-shot model occasionally shows inconsistencies at crop boundaries; however, these artifacts disappear after fine-tuning on the corresponding dataset.

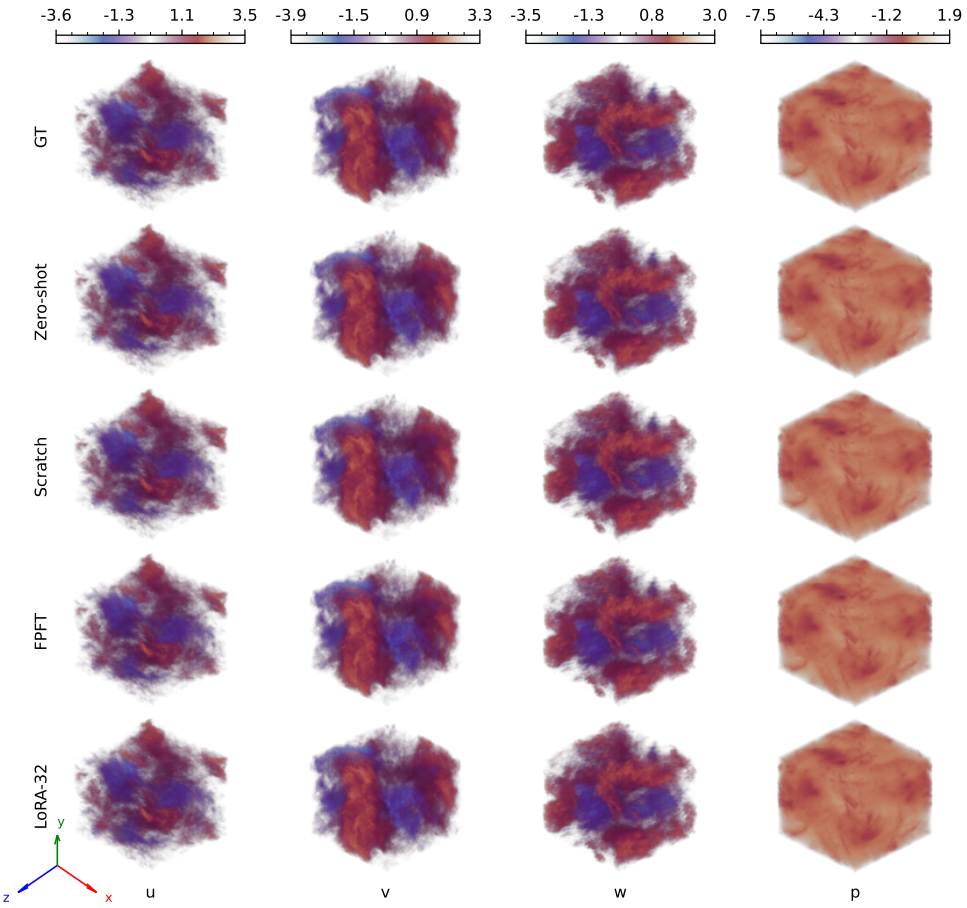

Figure 15: Volume rendering of the reconstructed 1024^3 `Iso` fields generated by different Tadpole training methods.

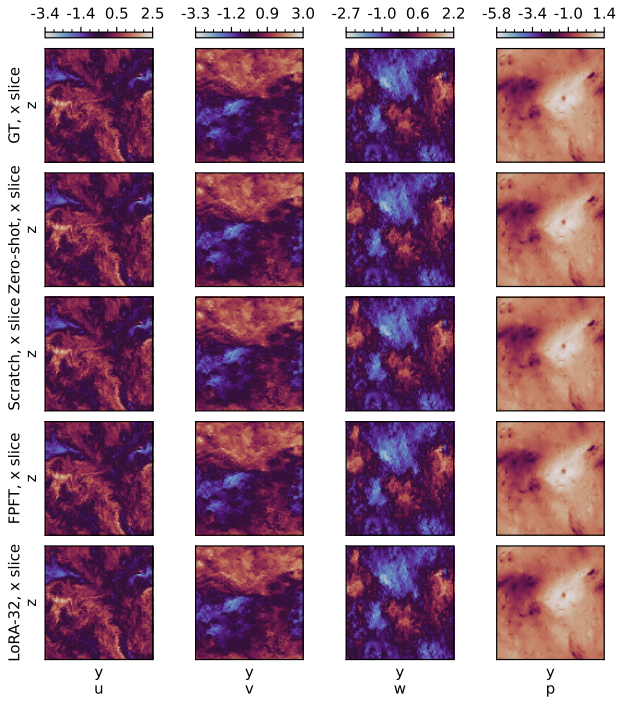

Figure 16: Visualization of the reconstruction of `Iso` at the slice where $x = X/2$.

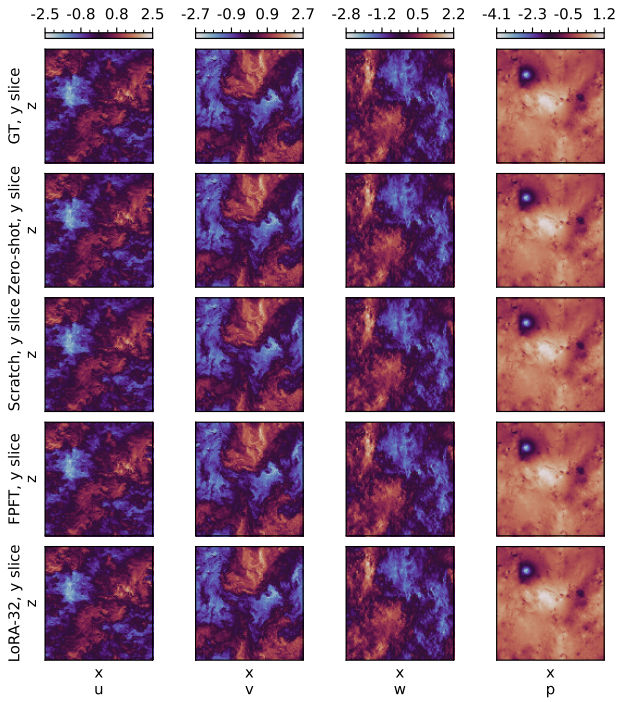

Figure 17: Visualization of the reconstruction of `Iso` at the slice where $y = Y/2$.

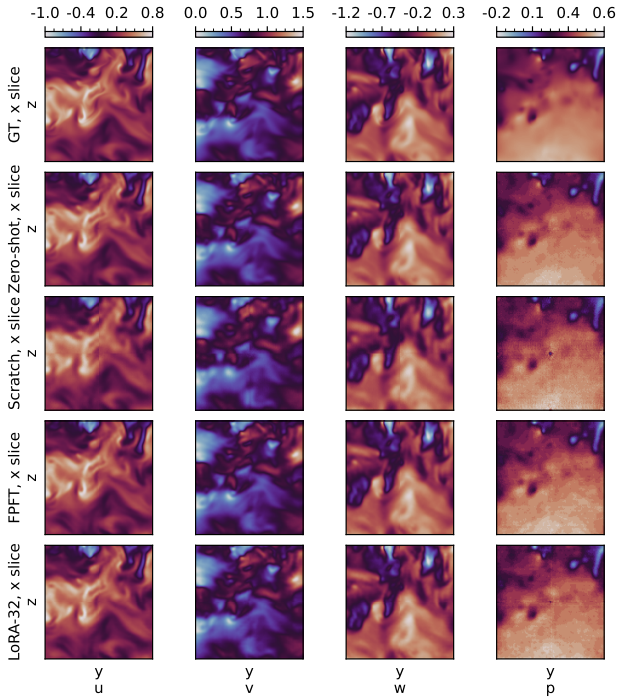

Figure 18: Visualization of the reconstruction of `Iso` 128^3 crops at the slice where $x = X/2$.

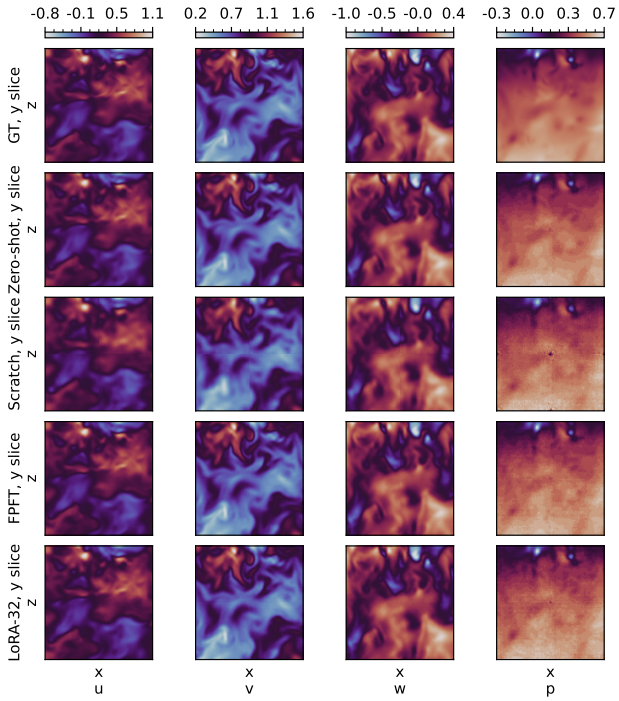

Figure 19: Visualization of the reconstruction of `Iso` 128^3 crops at the slice where $y = Y/2$.

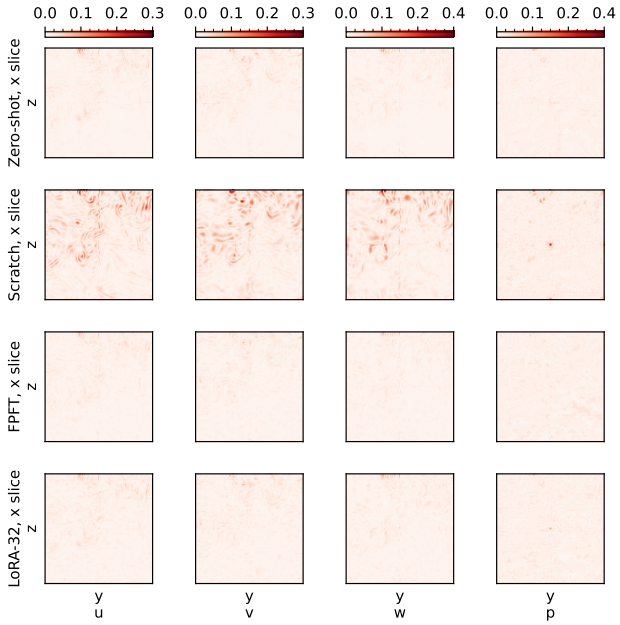

Figure 20: Visualization of the absolute error for the reconstruction of `Iso` 128^3 crops at the slice where $x = X/2$.

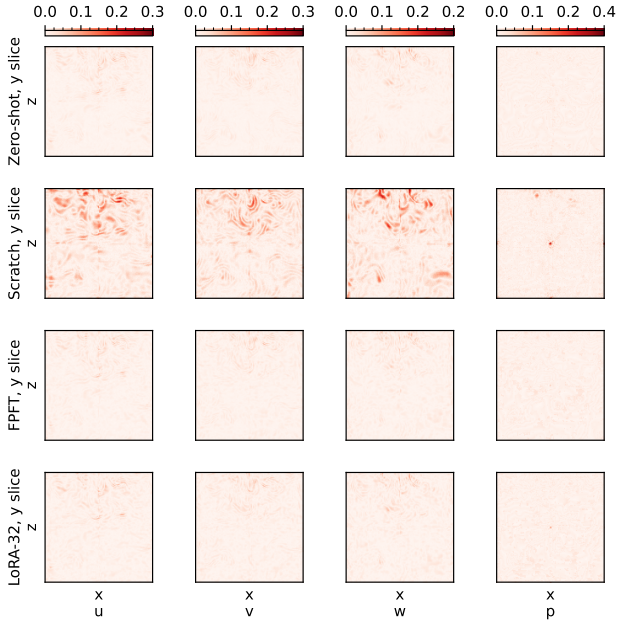

Figure 21: Visualization of the absolute error for the reconstruction of `Iso` 128^3 crops at the slice where $y = Y/2$.

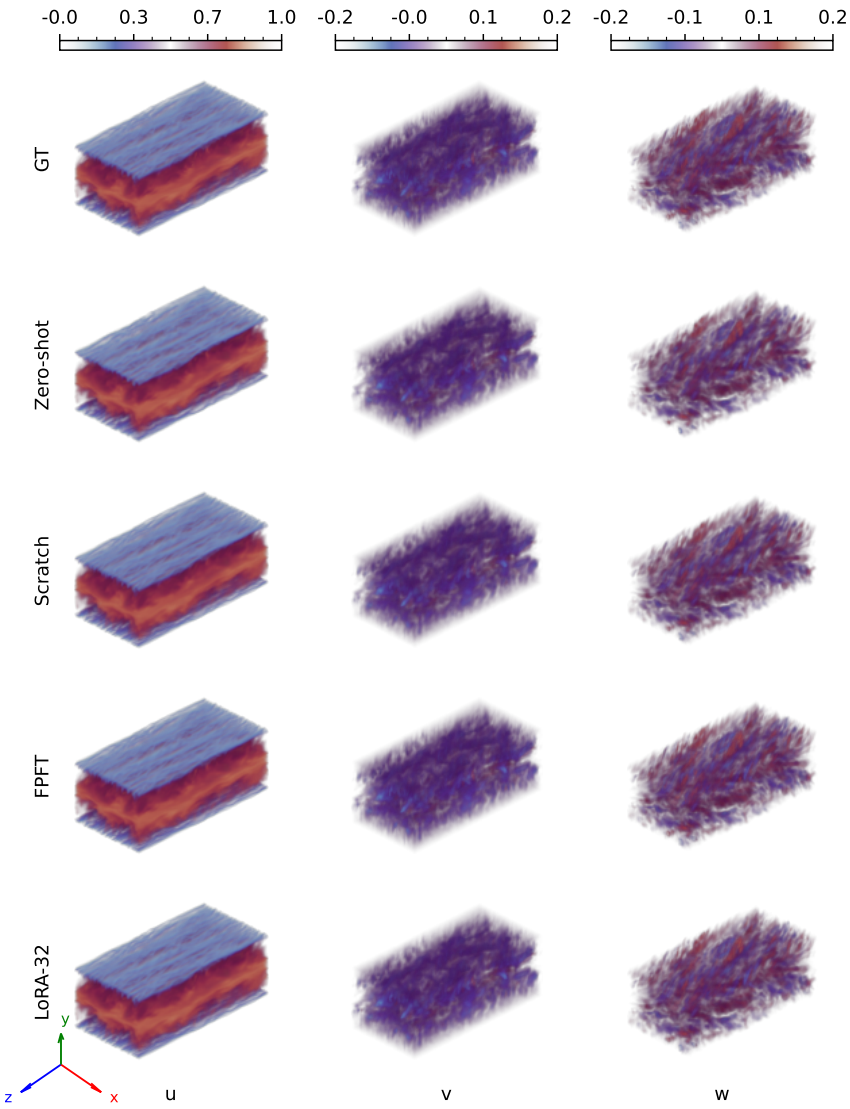

Figure 22: Volume rendering of the reconstructed $96^2 \times 192$ TCF fields generated by different Tadpole training methods. This visualization confirms that all methods have successfully learned the large-scale structures of the data. Differences become apparent in the following visualizations of individual slices through the volume.

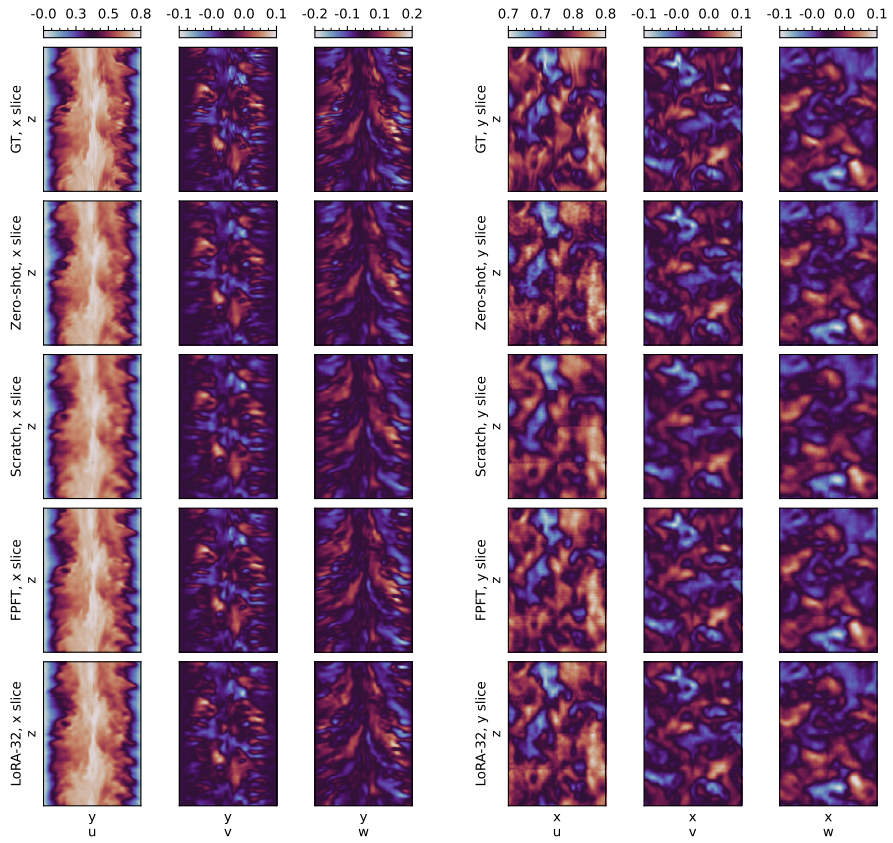

Figure 23: Visualization of the reconstruction of TCF at the slice where $x = X/2$ and $y = Y/2$.

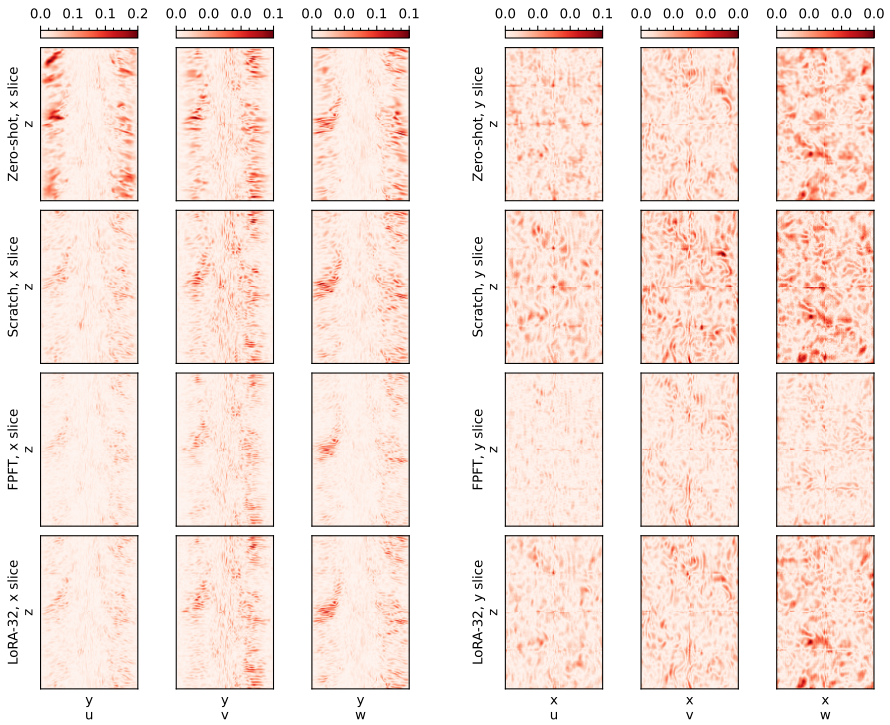

Figure 24: Visualization of the absolute error for the reconstruction of TCF at the slice where $x = X/2$ and $y = Y/2$.

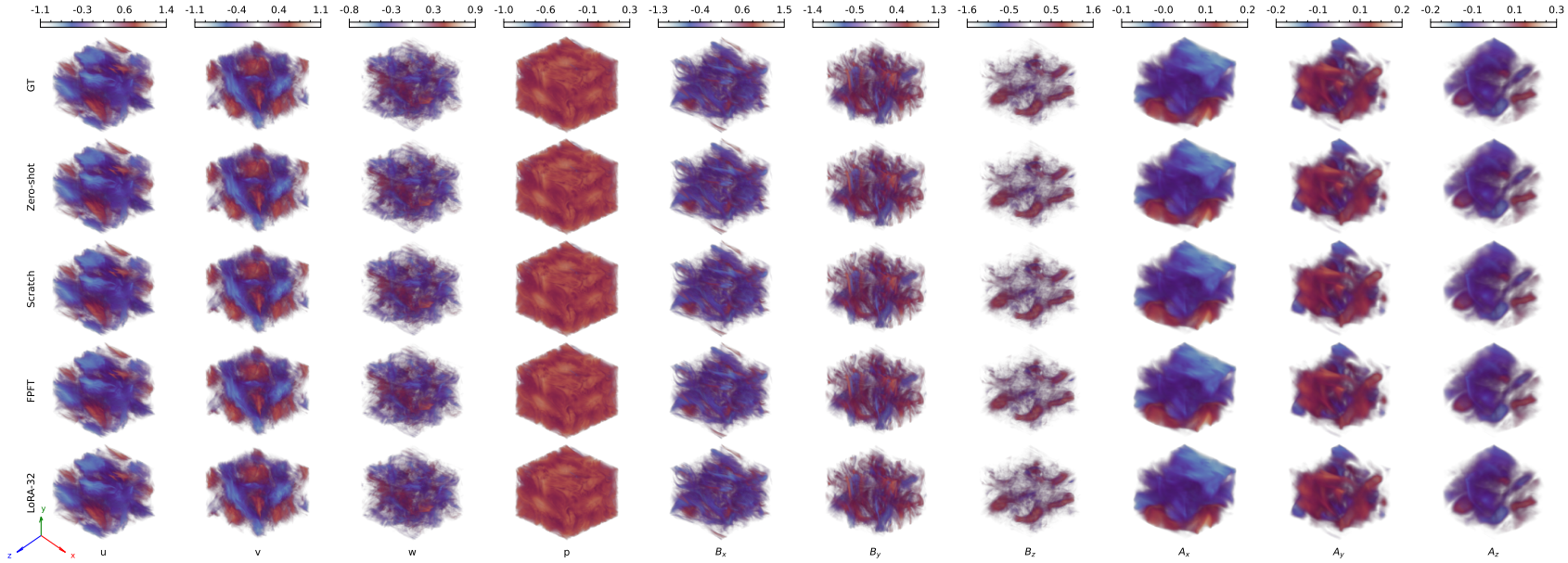

Figure 25: Volume rendering of the reconstructed 512^3 MHD fields generated by different Tadpole training methods.

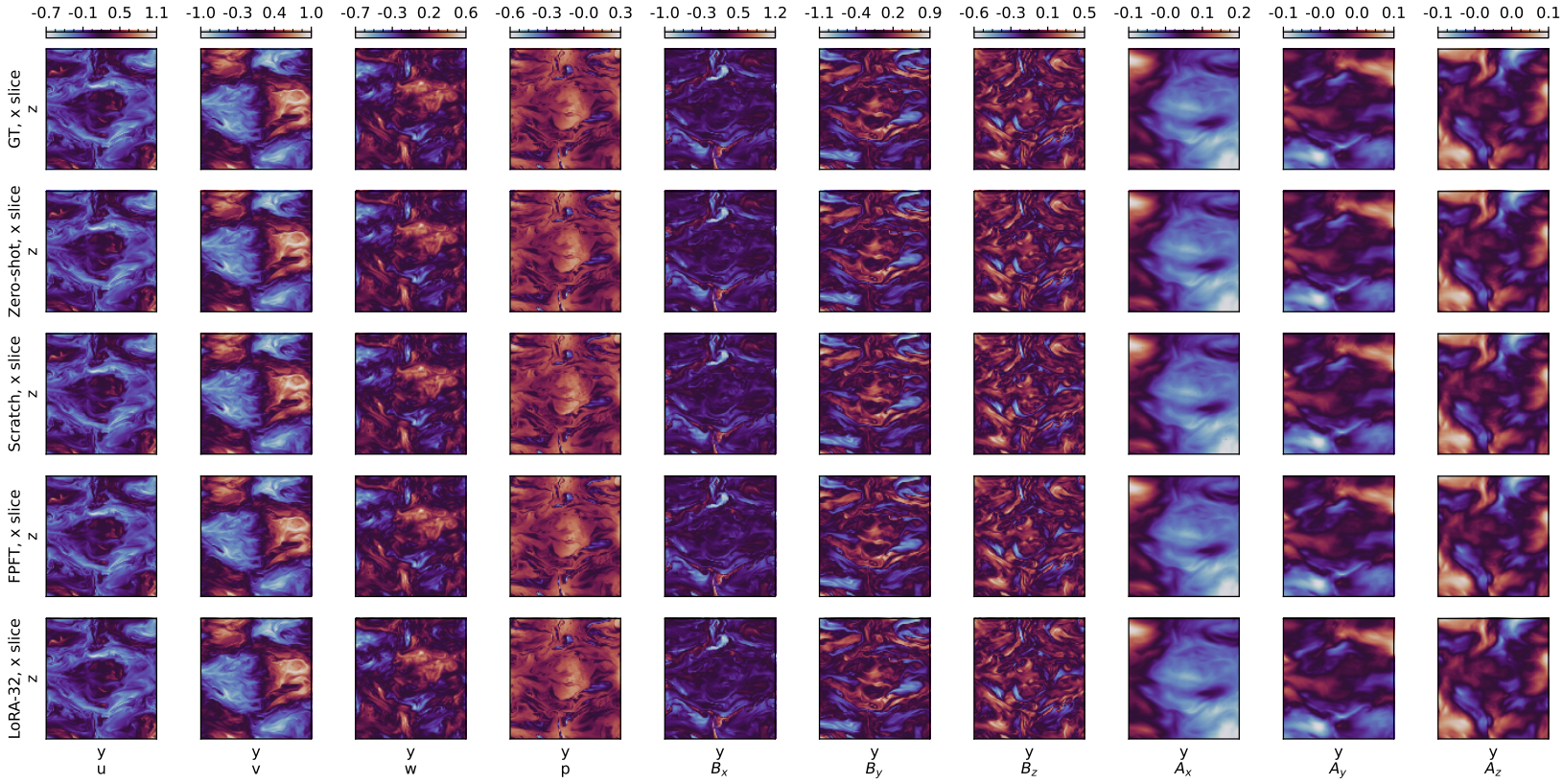

Figure 26: Visualization of the reconstruction of MHD at the slice where $x = X/2$.

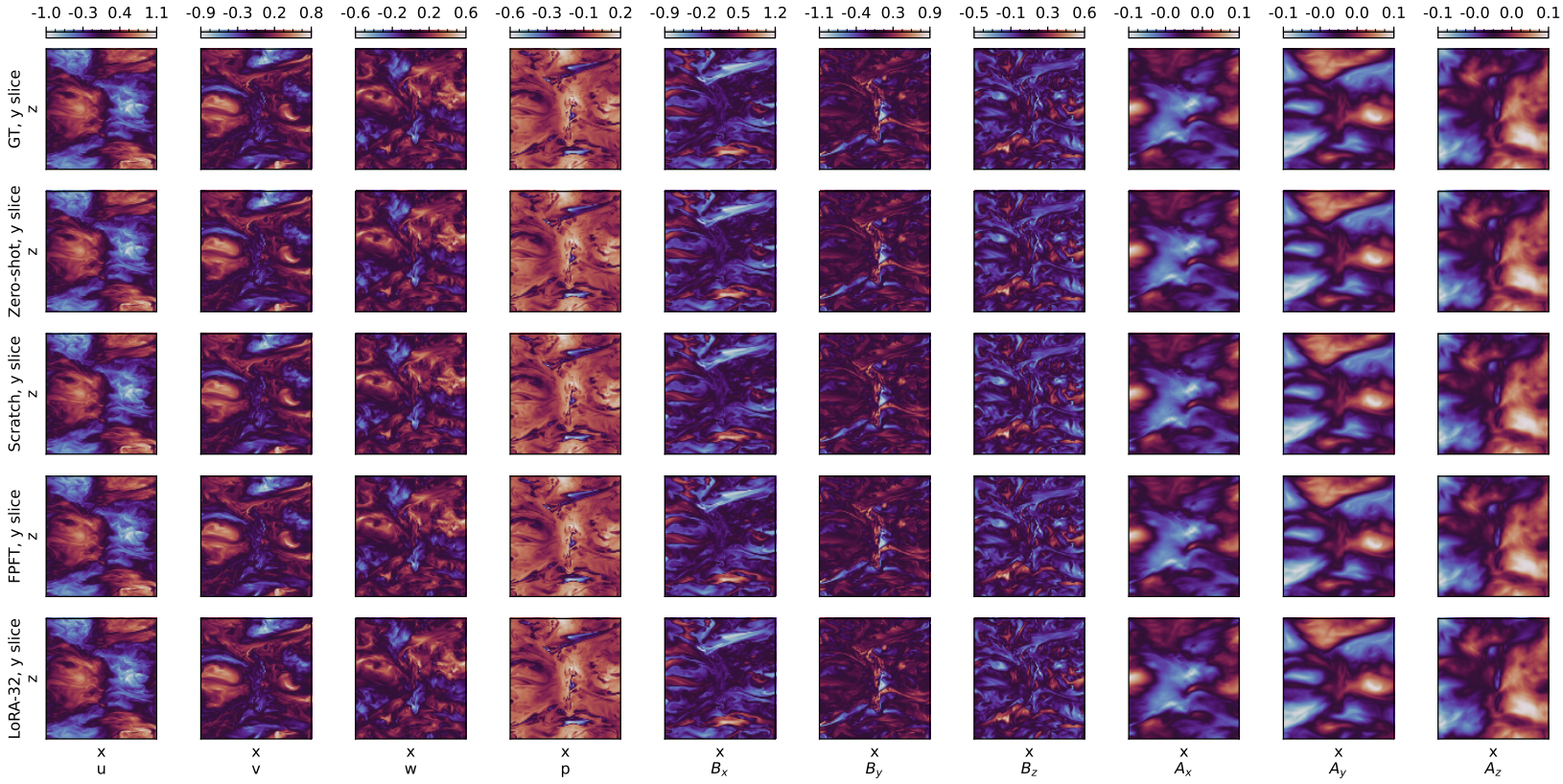

Figure 27: Visualization of the reconstruction of MHD at the slice where $y = Y/2$.

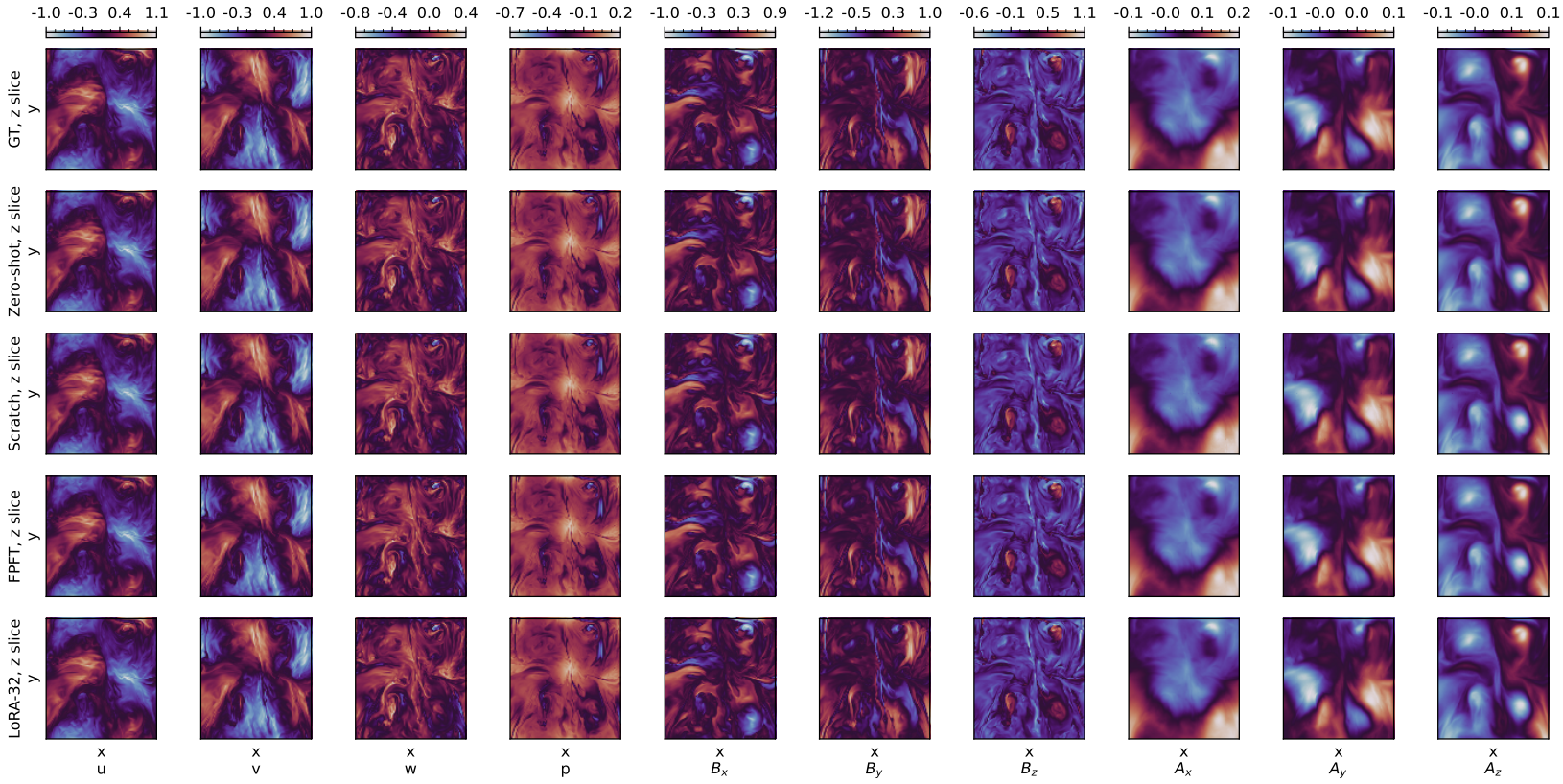

Figure 28: Visualization of the reconstruction of MHD at the slice where $z = Z/2$.

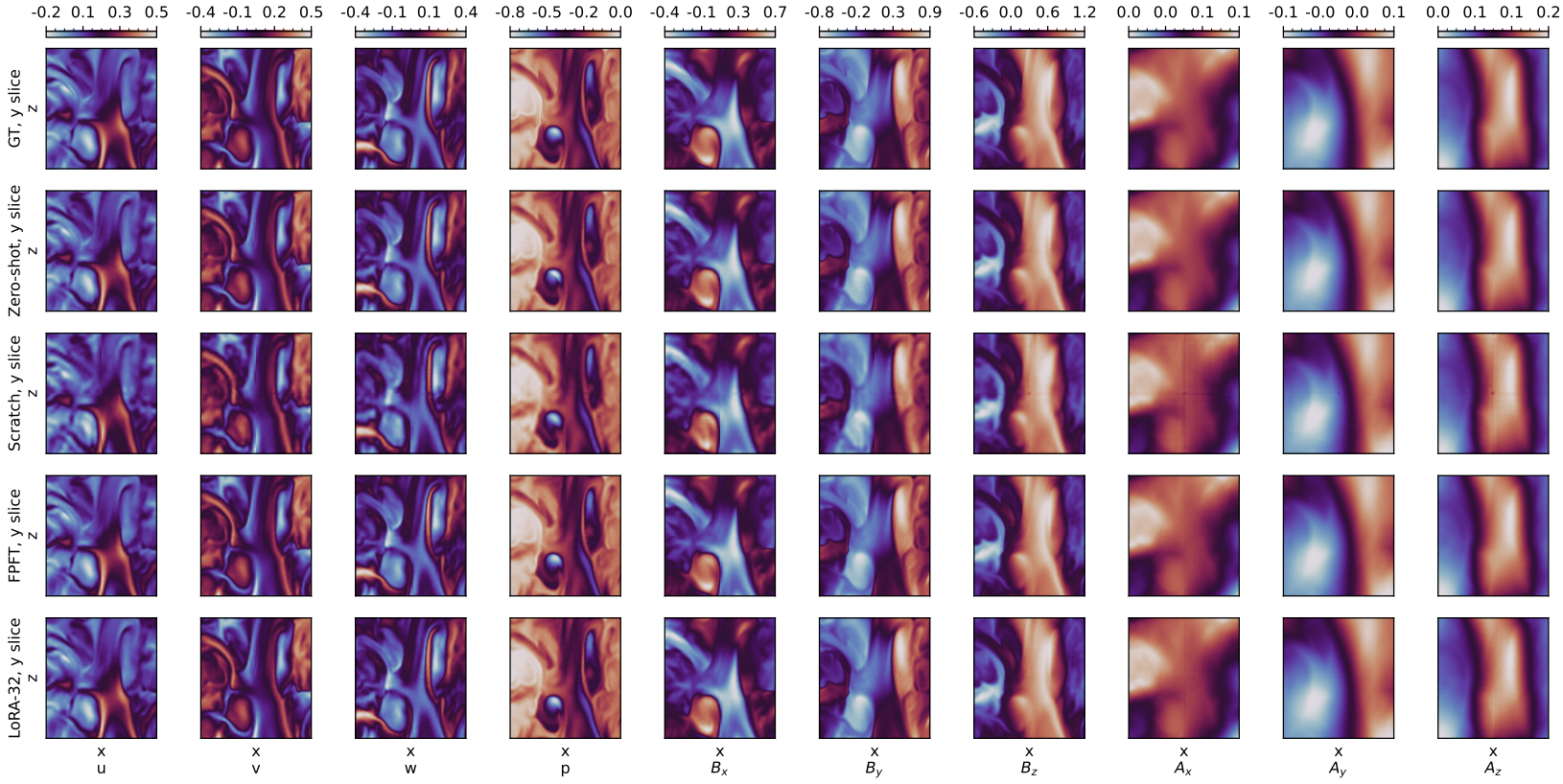

Figure 29: Visualization of the reconstruction of MHD 128^3 crops at the slice where $y = Y/2$.

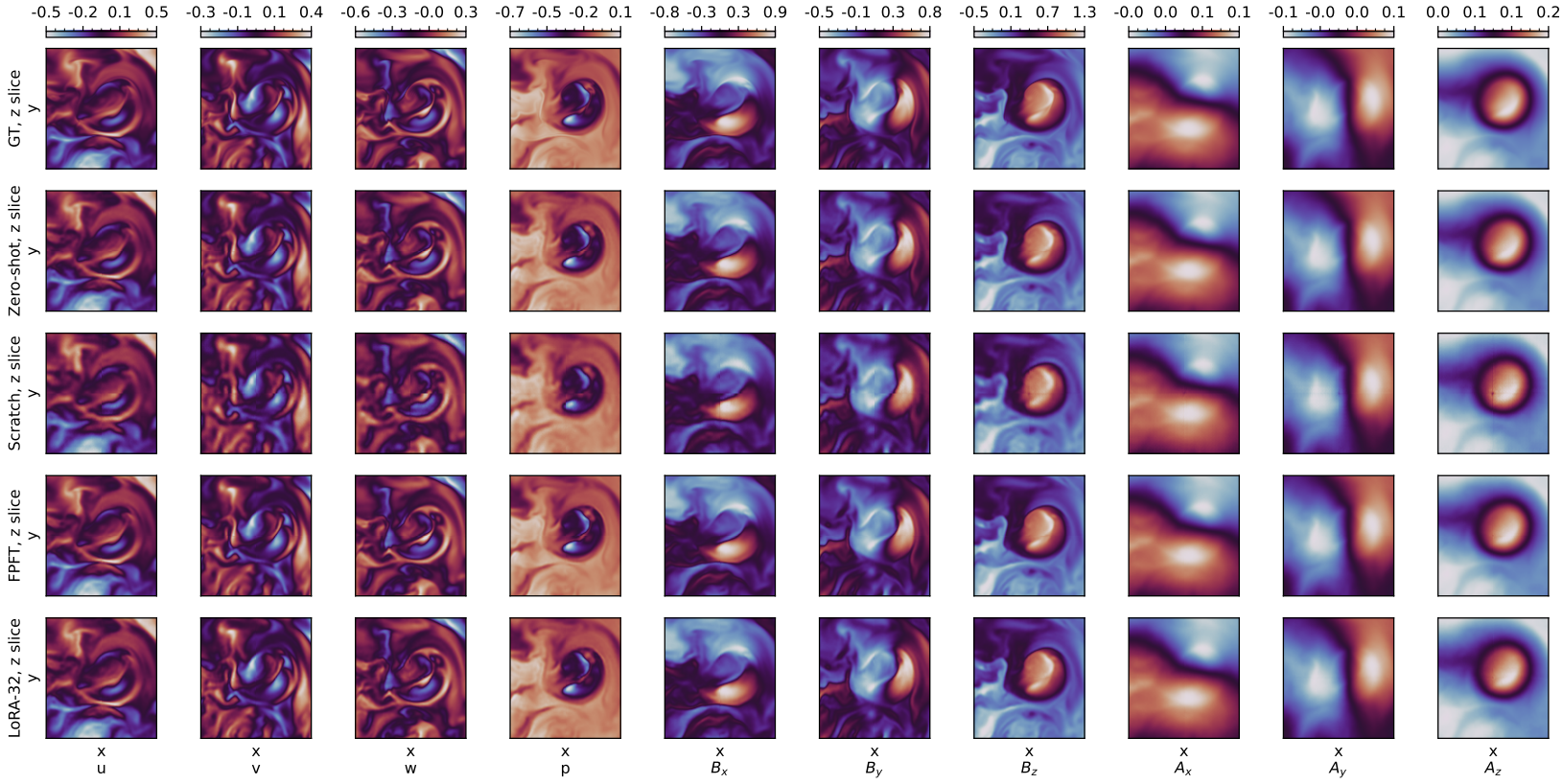

Figure 30: Visualization of the reconstruction of MHD 128^3 crops at the slice where $z = Z/2$.

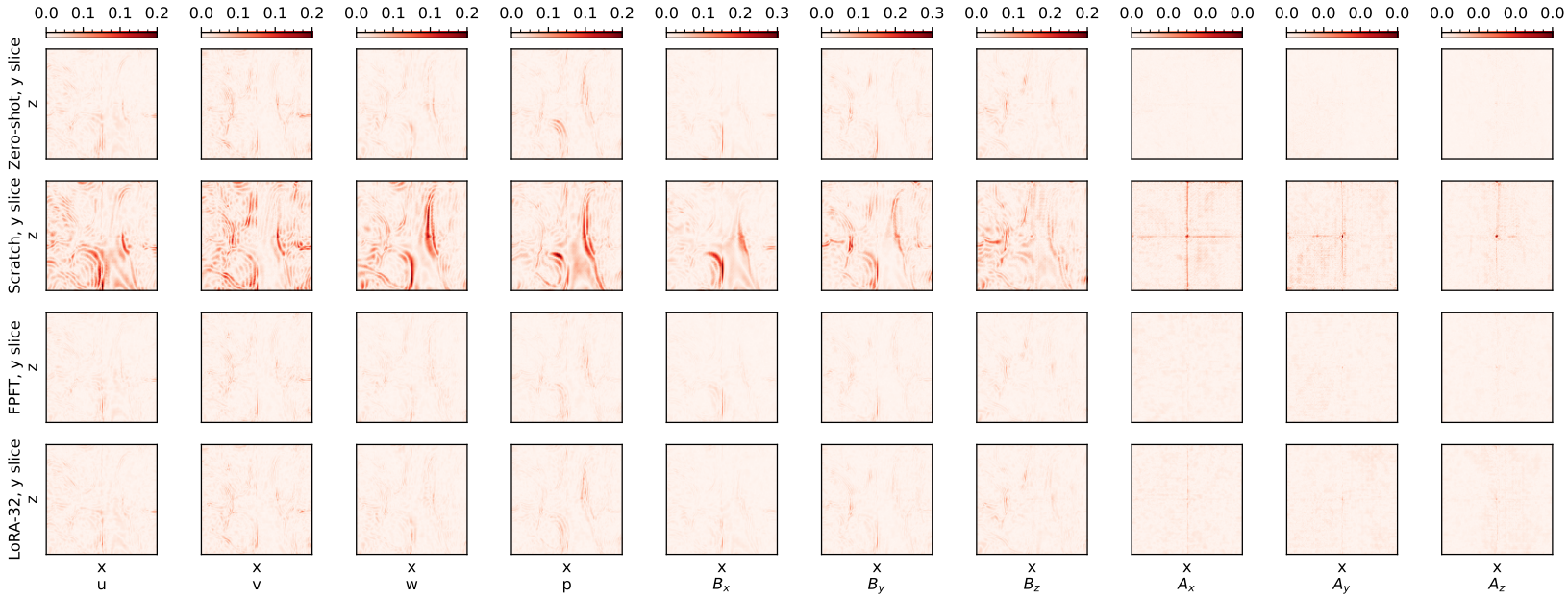

Figure 31: Visualization of the absolute error for the reconstruction of MHD 128^3 crops at the slice where $y = Y/2$.

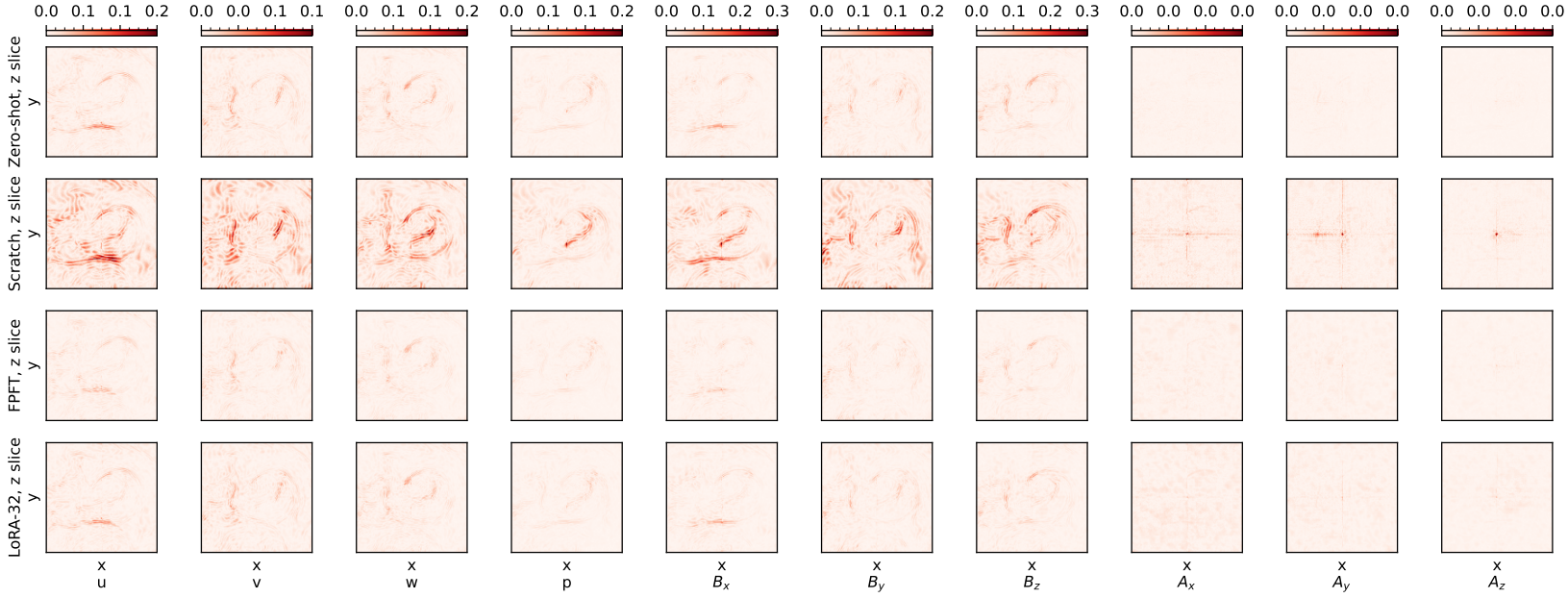

Figure 32: Visualization of the absolute error for the reconstruction of MHD 128^3 crops at the slice where $z = Z/2$.

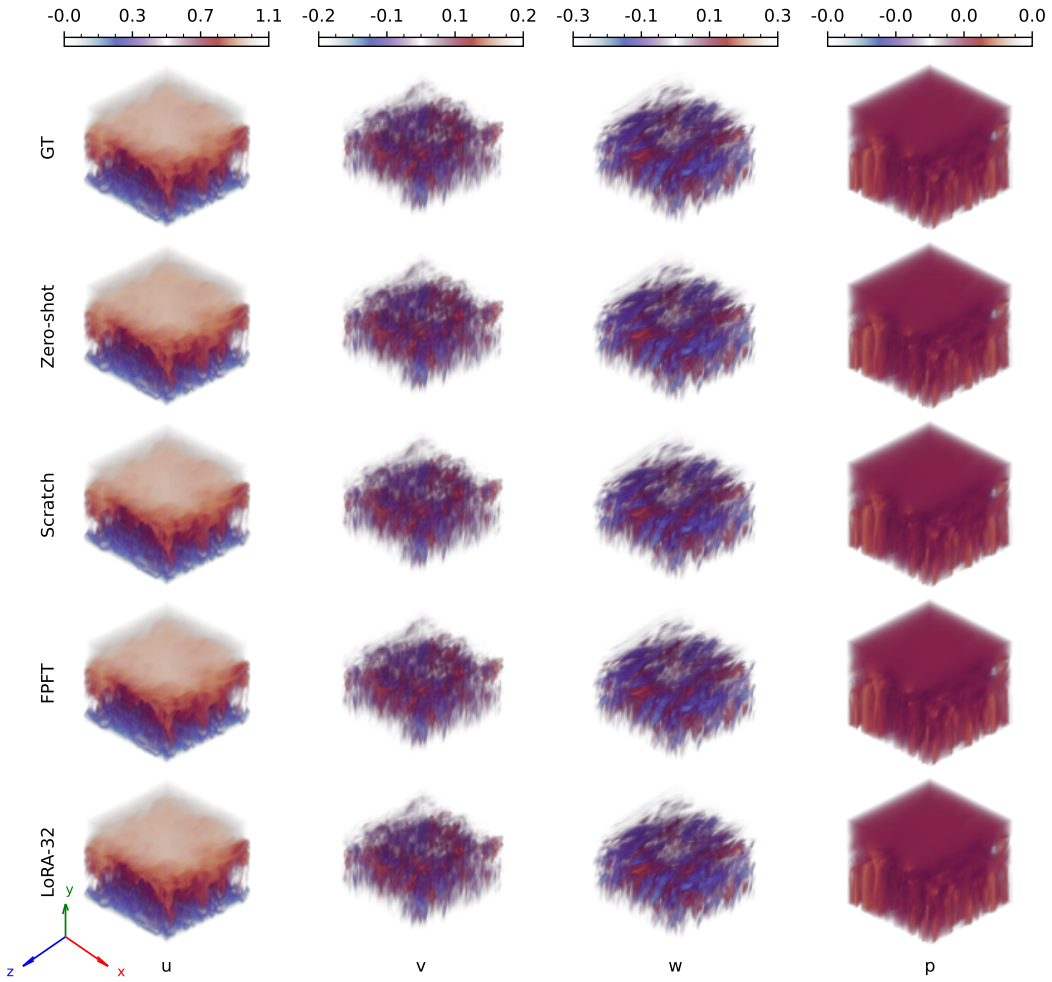

Figure 33: Volume rendering of the reconstructed 224^3 TBL fields generated by different Tadpole training methods.

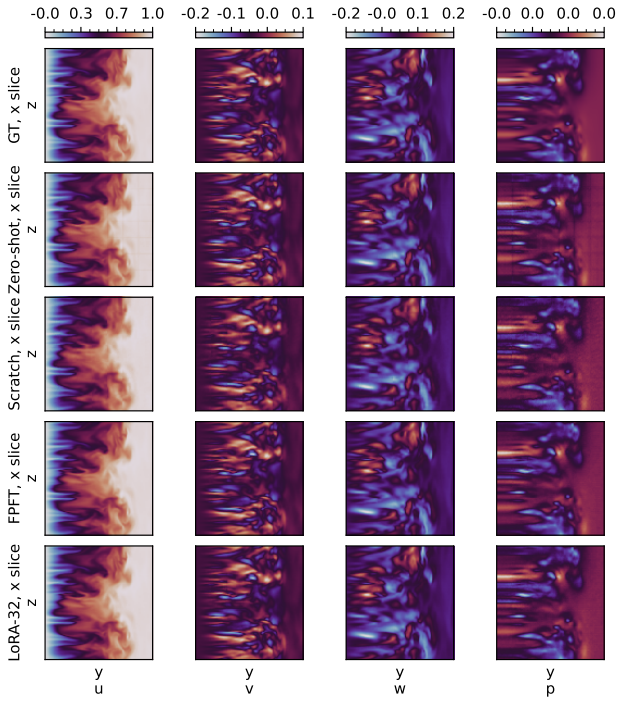

Figure 34: Visualization of the reconstruction of TBL at the slice where $x = X/2$.

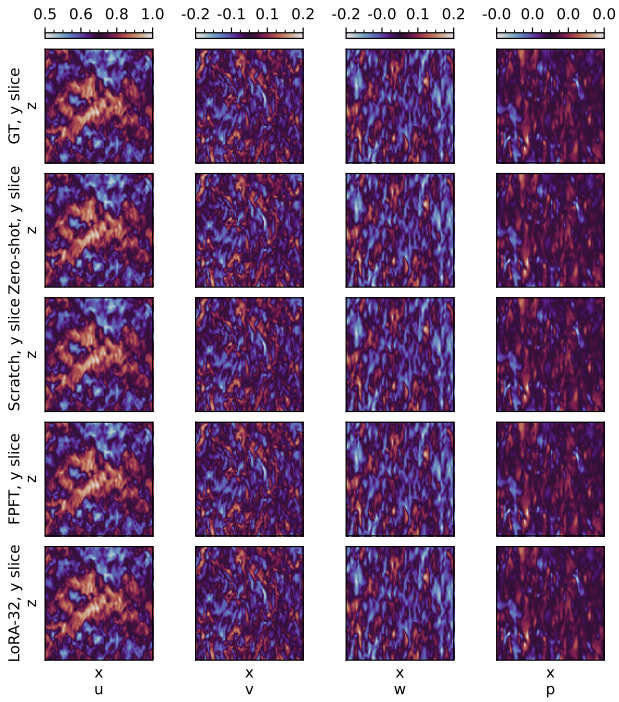

Figure 35: Visualization of the reconstruction of TBL at the slice where $y = Y/2$.

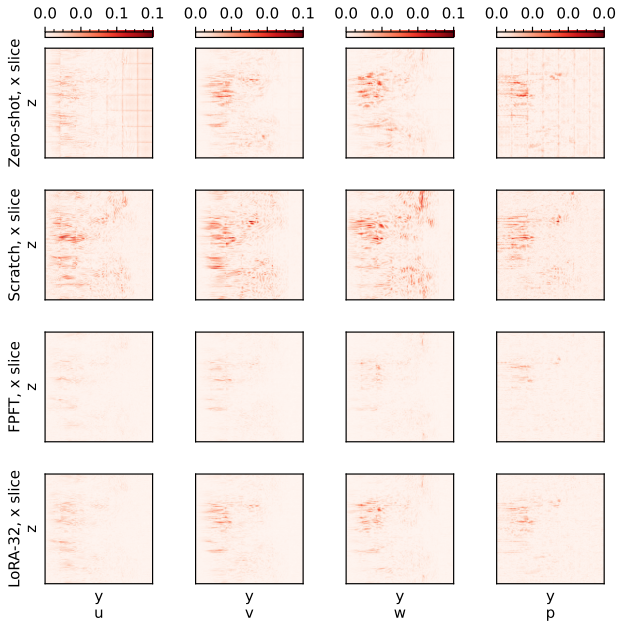

Figure 36: Visualization of absolute error for the reconstruction of TBL at the slice where $x = X/2$.

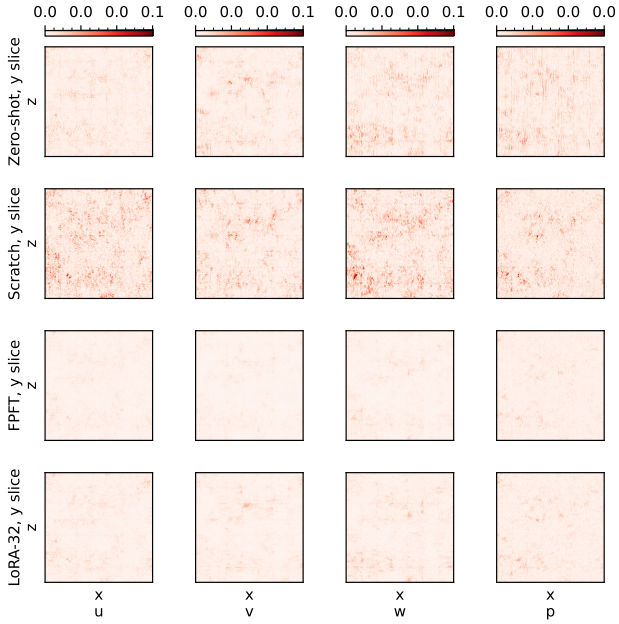

Figure 37: Visualization of absolute error for the reconstruction of TBL at the slice where $y = Y/2$.

A.10 Visualizations on dynamics learning

The following images provide qualitative samples for the dynamics rollout task over the course of three time steps. Each image compares the ground truth result (at the top) with the three baseline models (next three rows, top to bottom: Walrus, DPOT, and MORPH), with the Tadpole-DFT result shown in the bottom row.

As indicated by the MSE measurements in the main text, both DPOT and MORPH exhibit a worse performance. The outputs are relatively smooth, and both methods show clear patch boundaries induced by the underlying architectures. The MORPH model induces a substantial amount of smoothing in the first step, while DPOT loses detail more gradually. In contrast, both Walrus and Tadpole produce outputs that are hard to distinguish from the ground truth. Importantly, Tadpole achieves this with a more than $100\times$ smaller number of trainable weights.

In addition, the Walrus results feature noticeable noise in the pressure channel (e.g., fourth column in Fig. 40). This is most likely caused by the Walrus model having difficulties in adjusting to the different scales of velocity and pressure channels. The outputs of the Tadpole model are significantly smoother without losing detail.

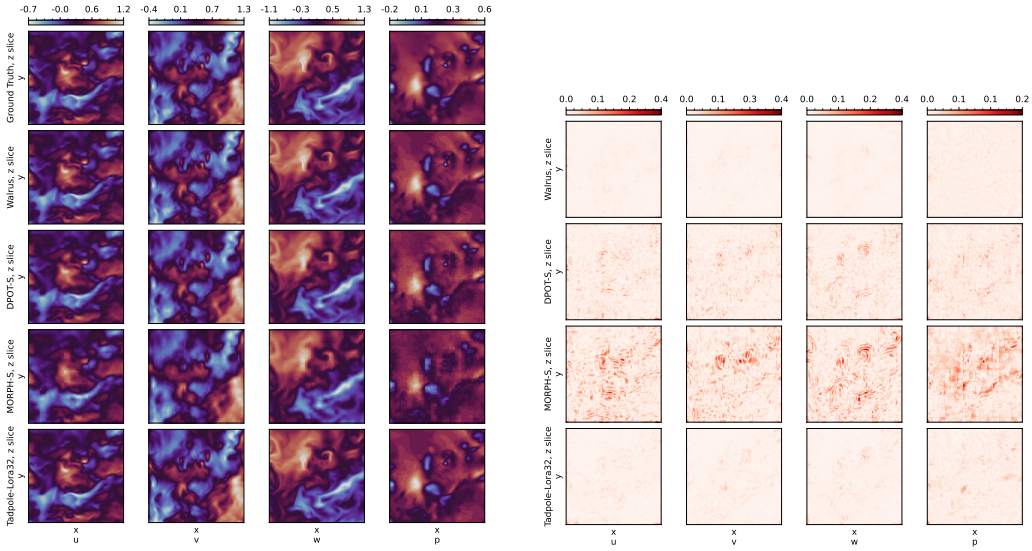

Figure 38: Visualization of the prediction (left) of `Iso` and the corresponding absolute error (right) at the **first rollout step** and the slice where $z = Z/2$.

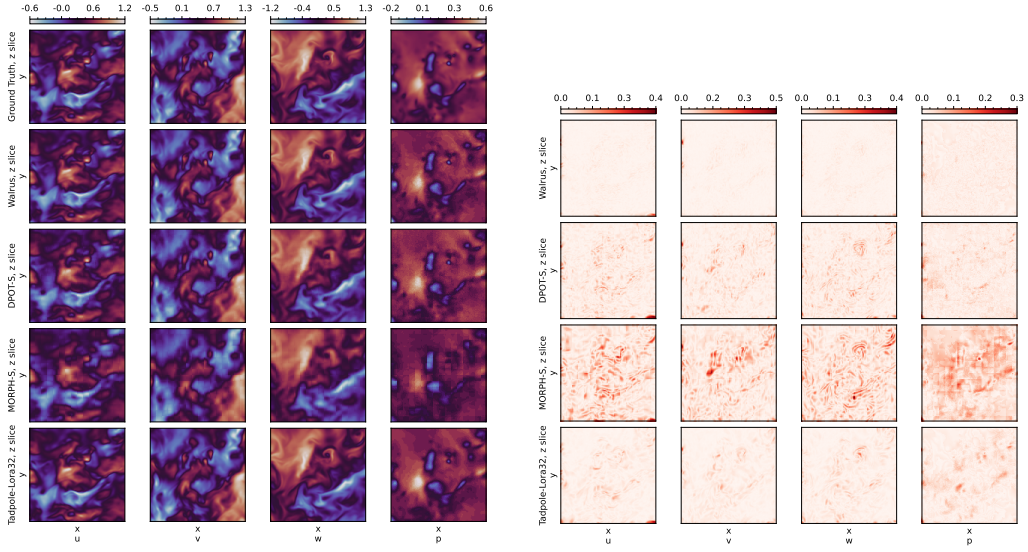

Figure 39: Visualization of the prediction (left) of `Iso` and the corresponding absolute error (right) at the **second rollout step** and the slice where $z = Z/2$.

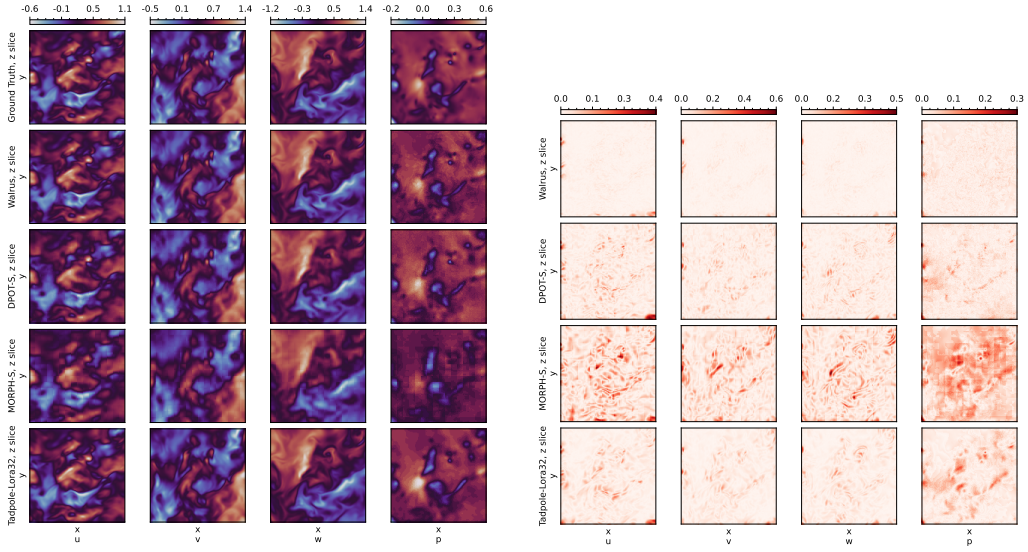

Figure 40: Visualization of the prediction (left) of `Iso` and the corresponding absolute error (right) at the **third rollout step** and the slice where $z = Z/2$.

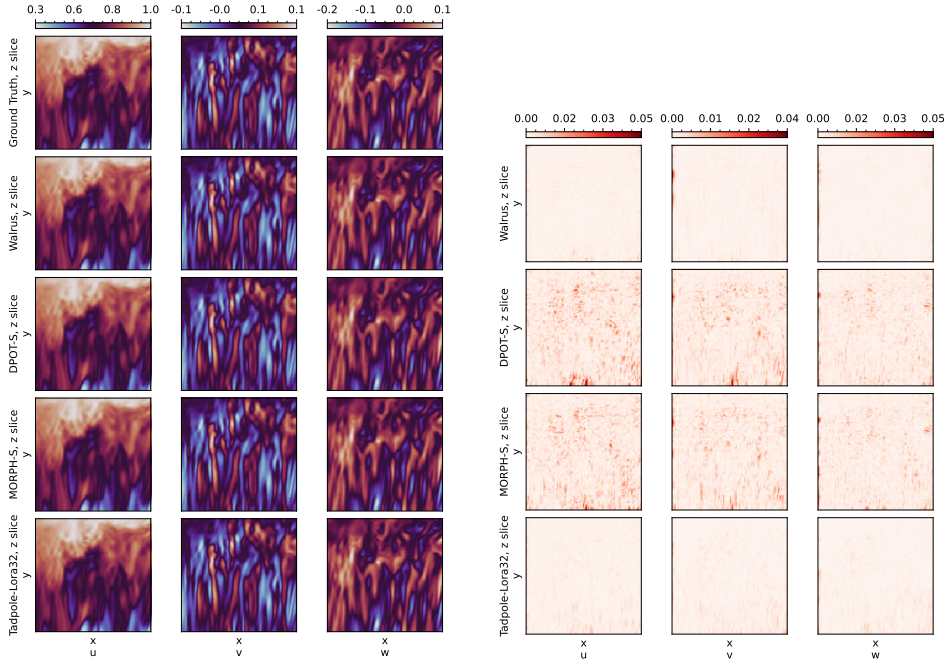

Figure 41: Visualization of the prediction (left) of TBL and the corresponding absolute error (right) at the **first rollout step** and the slice where $z = Z/2$.

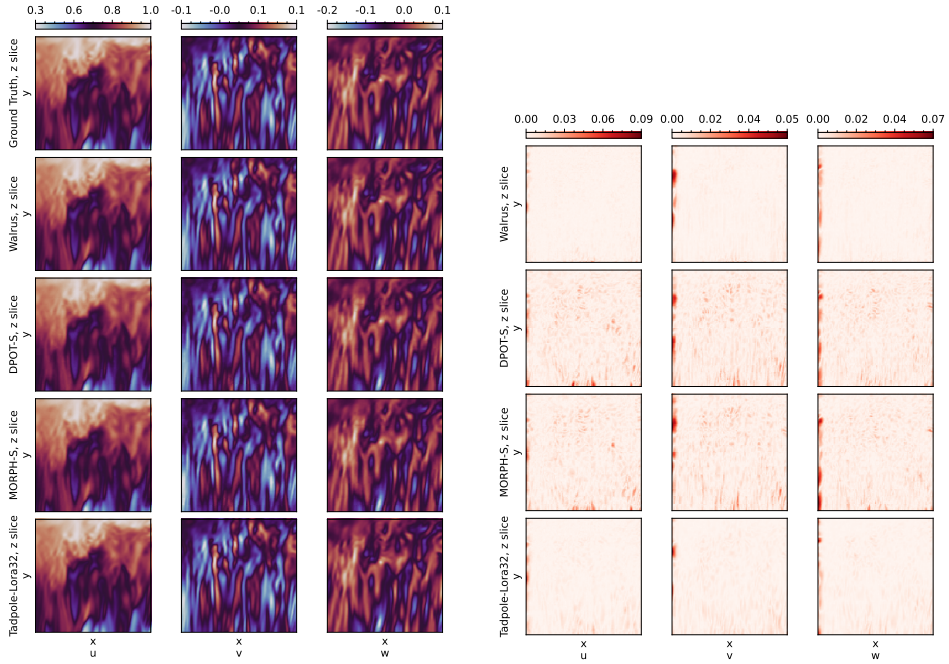

Figure 42: Visualization of the prediction (left) of TBL and the corresponding absolute error (right) at the **second rollout step** and the slice where $z = Z/2$.

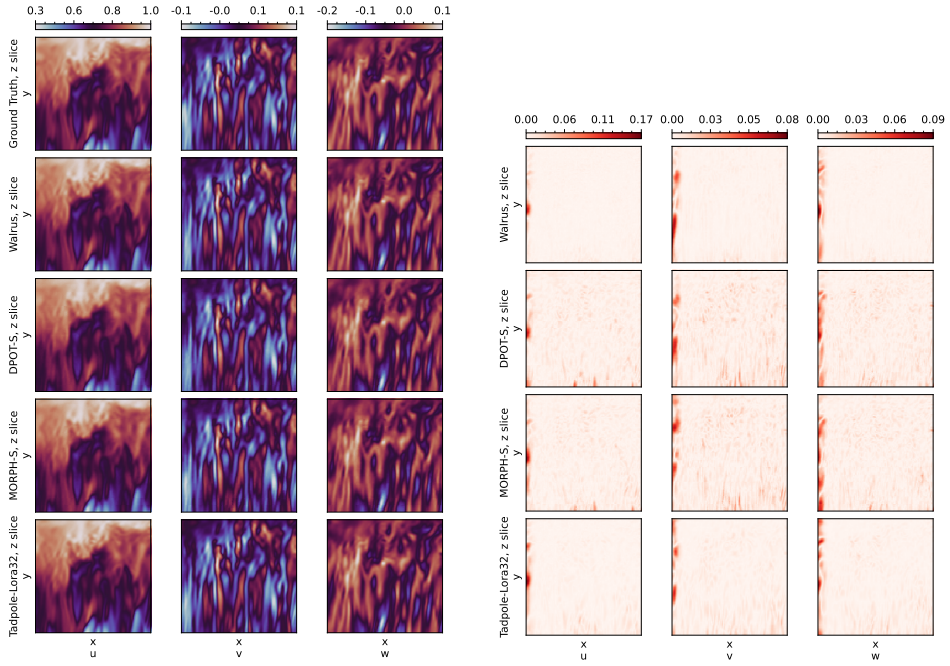

Figure 43: Visualization of the prediction (left) of TBL and the corresponding absolute error (right) at the **third rollout step** and the slice where $z = Z/2$.

B Dataset and online learning setups

B.1 Pre-training dataset

A significant challenge in training 3D physics foundation models is the production, storage, and efficient loading of large-scale spatiotemporal datasets. To illustrate the magnitude of this bottleneck, consider a single trajectory of 100 frames on a 384^3 grid for a vector field with three components. In half-precision floating-point format, a single trajectory requires approximately 30 GB of storage. Scaling this to a dataset of just 100 trajectories per phenomenon across 10 different physical phenomena results in a storage requirement between 10 TB and 100 TB.

Beyond storage, the I/O overhead required to shuffle and stream this data to GPUs often exceeds the computational cost of the training step itself. While many 3D phenomena are computationally expensive to simulate (requiring at least minutes per frame), there exists a class of semi-linear Partial Differential Equations (PDEs) that admit fast, stable solutions via spectral methods. By leveraging efficient spectral solvers on modern GPUs, we decouple the training process from disk I/O. This procedural, online data generation strategy enables access to a theoretically infinite dataset with great spectral diversity and significantly lower engineering overhead than traditional offline storage.

This appendix details the mathematical formulation of the governing equations, the spectral solver implementation, and the communication strategy used to bridge the simulation and training environments. Corresponding numerical solvers have been independently released as TorchFSM [53] and can be used in other physics-based deep learning research.

B.1.1 Overview of equations and their configurations

Consider a scaled three-dimensional unit cube $\Omega = (0, L)^3 \subset \mathbb{R}^3$ in which L describes the extent along each dimension. We assume periodic boundary conditions in each direction. On this domain, we want to solve equations of the form

$$\partial_t u = \mathcal{L}u + \mathcal{N}(u), \tag{3}$$

Table 11: Mathematical formulation of the considered PDEs. We denote scalar fields by u and vector fields by $\mathbf{u}$. Parameters drawn from distributions are sampled per simulator instance, see Appendix B.2.1.

Equation	**Linear Term $\mathcal{L}$**	**Nonlinear Term $\mathcal{N}$**	**Parameters**
Diffusion	$\nu\Delta u$	0	$\nu \sim \mathcal{U}(\text{5e-4}, \text{5e-3})$
Hyper-Diffusion	$-\zeta\Delta^2 u$	0	$\zeta \sim \mathcal{U}(\text{5e-4}, \text{5e-3})$
Burgers	$\nu\Delta\mathbf{u}$	$-\mathbf{u}\cdot\nabla\mathbf{u}$	$\nu \sim \mathcal{U}(\text{1e-3}, \text{5e-3})$
Korteweg-de Vries (KdV)	$\xi(\mathbf{1}\cdot\nabla)\Delta\mathbf{u}$	$-\mathbf{u}\cdot\nabla\mathbf{u}$	$\xi = -6$
Kuramoto-Sivashinsky (KS)	$-(\Delta+\Delta^2)u$	$-\frac{1}{2}\|\nabla u\|^2$	None
Fisher-KPP	$\nu\Delta u + ru$	$-ru^2$	$\nu \sim \text{Log}\,\mathcal{U}(\text{1e-4}, 0.02)$ $r \sim \mathcal{U}(5, 15)$
Swift-Hohenberg	$ru - (1+\Delta)^2 u$	$u^2 - u^3$	$r = 0.1$

where $\mathcal{L}$ describes a linear differential operator and $\mathcal{N}(\cdot)$ a nonlinear differential operator. Specifically, we are interested in equations in which the order of derivatives in $\mathcal{L}$ is higher than in $\mathcal{N}(\cdot)$. PDEs of this form are also called *semi-linear*. We select a diverse set of equations to cover a wide range of spectral characteristics, including diffusive, dispersive, shock-forming, chaotic, and pattern-forming dynamics. Table 11 summarizes the mathematical formulation of the selected equations. The specific sampling parameters (time-steps Δt, grid resolutions N, and warmup periods) are detailed in the subsequent implementation section (see Table 12 and Table 13).

Diffusion and Hyper-Diffusion Equation

The diffusion equation describes an isotropic smoothing process where the initial energy dissipates over time. The spectrum decays radially as $|\hat{u}(\mathbf{k})| \propto \exp(-\nu\|\mathbf{k}\|_2^2 t)$. To introduce higher-order damping effects often seen in turbulence modeling, we also include the hyper-diffusion equation, which applies a fourth-order Laplacian, resulting in a steeper quartic spectral decay $|\hat{u}(\mathbf{k})| \propto \exp(-\zeta\|\mathbf{k}\|_2^4 t)$.

Burgers Equation

We utilize the viscous 3D Burgers' equation to model shock formation and the transfer of energy from large to small scales. The solution is a three-dimensional vector field. This nonlinear PDE generates sharp gradients, which are essential for training the model to resolve high-frequency features.

Korteweg-de-Vries (KdV) Equation

While classically a 1D equation describing soliton waves, we extend the KdV dynamics to 3D by employing a similar nonlinearity as the Burgers equation and by using an extension of the dispersion effect to higher dimensions. This system balances nonlinearity with dispersion rather than diffusion.

Kuramoto-Sivashinsky (KS) Equation

The KS equation is characterized by a negative diffusion term (instability at low wavenumbers) stabilized by hyper-diffusion at high wavenumbers. This balance between energy production, energy movement based on the nonlinearity and high-frequency dissipation is known for exhibiting spatiotemporal chaos, yielding rich spectra.

Fisher-KPP Equation

The Fisher-Kolmogorov-Petrovsky-Piscounov (Fisher-KPP) equation combines diffusion with a logistic reaction term. The polynomial nonlinearity contributes interesting spectral components.

Swift-Hohenberg (SH) Equation

This equation is a canonical model for pattern formation leading to the emergence of complex spatial structures. We select parameters to ensure the system remains in the pattern-forming regime. These patterns are characterized by spectral richness.

Table 12: Discretization configurations for each considered PDE system. Note that values differ depending on the resolution N on which the system is simulated. The effective Δt realized when recording the trajectory is the one given here multiplied by the **Save Frequency**, allowing substepping on stiffer configurations. After data recording, each frame is voxel-wise normalized to be within **Value Range**, which was precomputed based on reasonable limits seen in the data distribution.

Equation	Extent L	Time-Step Size Δt	Save Frequency	Value Range
Diffusion	1	5e-4 $N = 64$ 5e-5 $N \in \{128, 256, 384\}$	1	$[-1, 1]$
Hyper-Diffusion	1	5e-4 $N = 64$ 5e-5 $N \in \{128, 256, 384\}$	1	$[-1, 1]$
Burgers	1	5e-3 $N = 64$ 2e-3 $N = 128$ 1e-3 $N \in \{256, 384\}$	1 $N = 64$ 2 $N = 128$ 5 $N \in \{256, 384\}$	$[-1, 1]$
Korteweg-de-Vries	1	2e-6	2	$[-1.25, 1.25]$
Kuramoto-Sivashinsky	64	0.1	1	$[-25, 25]$
Fisher-KPP	1	1e-3 $N \in \{64, 128\}$ 5e-4 $N \in \{256, 384\}$	1 $N \in \{64, 128\}$ 2 $N \in \{256, 384\}$	$[0, 1]$
Swift-Hohenberg	20	0.1 $N = 64$ 0.02 $N \in \{128, 256, 384\}$	1 $N = 64$ 2 $N = 128$ 5 $N \in \{256, 384\}$	$[-2, 3]$

B.1.2 Fast semi-linear PDE solvers in PyTorch

We solve the semi-linear PDEs $\partial_t u = \mathcal{L}u + \mathcal{N}(u)$ using Exponential Time Differencing Runge-Kutta (ETDRK) methods. These schemes are particularly well-suited for stiff PDEs where the linear term $\mathcal{L}$ contains high-order derivatives (e.g., Δ^2). By treating the linear part exactly via an integrating factor, we avoid the severe time-step restrictions typical of explicit schemes.

Table 13: Configurations for recording trajectories. **Warmup** is the number of frames starting from the initial condition that are being discarded *before* recording starts. This ensures we are within the physics state space we deem most interesting. The number of frames recorded per trajectory (after warmup) is given by **Trajectory Length**. Before moving on to the next simulator configuration, we repeat the recording process **Num Runs** times. As such, each setup always contributes 30 frames. We also conducted an ablation with an offline local training dataset of approximately 500GB. The **Offline Train Num Trajectory** settings denote how many trajectories are produced for each split of the data in the offline setup.

Equation	Warmup	Traj. Length	Num Runs	Offline Train Num Traj.
Diffusion	0	2	15	75
Hyper-Diffusion	0	2	15	75
Burgers	30 $N = 64$ 60 $N = 128$ 150 $N \in \{256, 384\}$	30	1	5
Korteweg-de-Vries	40	10	3	15
Kuramoto-Sivashinsky	500	30	1	5
Fisher-KPP	20 $N \in \{64, 128\}$ 40 $N \in \{256, 384\}$	10	3	15
Swift-Hohenberg	0	30	1	5

We discretize the scaled unit cube $\Omega = (0, L)^3$ into N intervals of size Δx per dimension (yielding N^3 total number of voxels). Then we consider the left end of each interval a nodal degree of freedom, i.e., in one dimension the grid points are located at positions $[0, \Delta x, 2\Delta x, \ldots, (N-1)\Delta x]^T \in \mathbb{R}^N$. This also means that the left end of the domain is considered a degree of freedom, while the right end is not, which naturally encodes periodic boundary conditions and is a prerequisite for most Fast Fourier Transform implementations. The three-dimensional grid is given by the tensor product of the one-dimensional coordinates.

We denote by $\mathbf{u}_t \in \mathbb{R}^{C \times N \times N \times N}$ a state array at time point t with C channels and a value for each valid degree of freedom. Applying a three-dimensional discrete Fourier $\mathcal{F}$ transformation yields $\hat{\mathbf{u}}_t \in \mathbb{C}^{C \times N \times N \times N}$ which is an array of the same shape but with complex-valued entries [1].

In state space, time integration with a fixed time step size Δt can be represented by the time-stepping operator $\mathcal{P}$ which yields

$$\mathbf{u}_{t+\Delta t} = \mathcal{P}(\mathbf{u}_t). \tag{4}$$

Due to the diagonalization of derivative operators in Fourier space, we choose to perform time integration in the spectral domain via

$$\mathbf{u}_{t+\Delta t} = \mathcal{F}^{-1}\left(\hat{\mathcal{P}}\left(\mathcal{F}\left(\mathbf{u}_t\right)\right)\right), \tag{5}$$

in which the spectral time-stepping operator $\hat{\mathcal{P}}(\cdot)$ is implemented via a two-stage process

$$\hat{\mathbf{u}}_* = \exp(\hat{\mathcal{L}}\Delta t) \odot \hat{\mathbf{u}}_t + \frac{\exp(\hat{\mathcal{L}}\Delta t) - 1}{\hat{\mathcal{L}}} \odot \hat{\mathcal{N}}(\hat{\mathbf{u}}_t), \tag{6}$$

$$\hat{\mathbf{u}}_{t+\Delta t} = \hat{\mathbf{u}}_* + \frac{\exp(\hat{\mathcal{L}}\Delta t) - 1 - \hat{\mathcal{L}}\Delta t}{\hat{\mathcal{L}}^2 \Delta t} \left(\hat{\mathcal{N}}(\hat{\mathbf{u}}_*) - \hat{\mathcal{N}}(\hat{\mathbf{u}}_t)\right). \tag{7}$$

Here, $\hat{\mathcal{L}}$ and $\hat{\mathcal{N}}(\cdot)$ are the linear and nonlinear differential operators in Fourier space, respectively. Both can be built using spectral derivative operators. As a property of spectral differentiation, these operators diagonalize, which allows all the operations in Eqs. (6) and (7) to be evaluated pointwise. This includes an elementwise exponentiation that is at the core of the ETDRK methods. For the nonlinear operator in Fourier space $\hat{\mathcal{N}}(\cdot)$, a pseudo-spectral evaluation strategy using transformation to the state space, evaluating the nonlinearity, and transforming back into Fourier space is used. We supplement this with appropriate anti-aliasing. The time integration procedure of Eqs. (6) and (7) is of second order consistency. We use this ETDRK2 method for most equations because it offers the best compromise between speed, accuracy, and stability in single precision across our tested systems. Only for the KdV equation, we use the fourth-order ETDRK version [54].

Since these methods are based on array computations, they operate efficiently within modern acceleration frameworks like PyTorch and can therefore be easily ported to GPUs. For more details on Fourier pseudo-spectral ETDRK methods, their implementation, and a discussion of their limitations, we refer to [54, 68, 55].

B.1.3 Overview of initializers

For many equations, their long-term behavior throughout the trajectory is influenced by the distribution of the initial state. For Tadpole, we used five initialization routines chosen to ensure a wide range of spectral representations. The primary difference between them is how the spectrum behaves as a function of wavenumber $\mathbf{k}$. Some of the initialization routines additionally sample hyperparameters according to Table 14.

For most initializers, we apply a post-processing step that clamps the state to the range $(c_{\text{min}}, c_{\text{max}})$. These limits c_{min} and c_{max} can either be fixed or randomly drawn. We normalize the initial conditions to ensure that their order of magnitude is around 1. Note that this normalization is different from the normalization of each recorded frame according to Table 12.

Each PDE system uses a distinct set of initializers and their respective hyperparameters. This is to enable sufficiently wide spectral representation in the created frames while ensuring stable time integration. We note down the match in Table 15.

[1] Note that given all our PDEs are real-valued, one could have used a real-valued FFT that typically halves the number of entries in the last array axis and halves the compute cost. However, for simplicity, we opted for the general FFT since we did not observe significant performance degradations in our online learning setup.

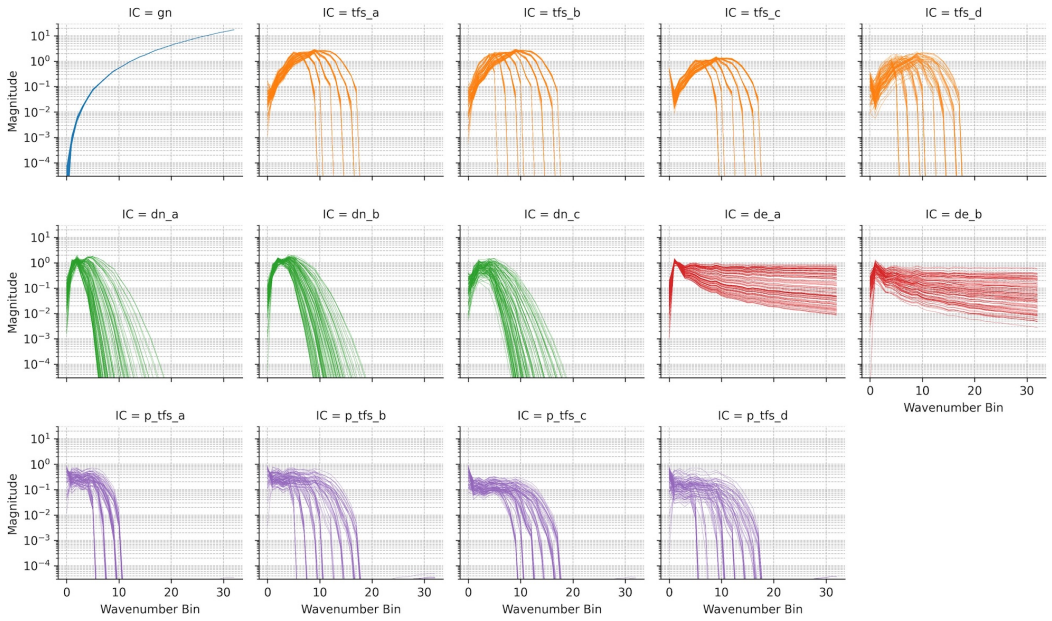

Figure 44: Radially *shell-aggregated* magnitude spectra (100 samples per distribution). **Row 1 (GN & TFS):** Gaussian (white) noise (GN) exhibits quadratic growth due to the volume of spherical shells in 3D Fourier space. The Truncated Fourier Series (TFS) initializers show distinct spectral cutoffs determined by k_{limit}. Note that TFS-D exhibits vertical variation due to its randomized normalization bounds. **Row 2 (DN & DE):** Diffused Noise (DN) follows an exponential quadratic decay ($\sim \exp(-\nu\|\mathbf{k}\|_2^2)$), with DN-A appearing to have more spectrally compact samples due to higher diffusivity sampling. Decayed Energy (DE) follows a rough power-law distribution with significant high-frequency contributions. **Row 3 (P-TFS):** The Poisson (P-TFS) initializers retain the cutoff frequency of the source TFS but exhibit smoother maxima and a steeper decay within the active modes, consistent with the smoothing properties of the inverse Laplacian.

In Fig. 44, we display 100 samples for each initial condition distribution.

Pixel-Wise Gaussian Noise (GN)

The state array for a single channel $\mathbf{u}_0 \in \mathbb{R}^{1\times N\times N\times N}$ is built by drawing each degree of freedom value identically and independently from a standard normal distribution

$$u_j \sim \mathcal{N}(0,1). \tag{8}$$

This corresponds to white noise. We use this initializer for the KS equation whose long-term behavior is independent of the initial state. For the other PDE systems, this initializer contains too much high-frequency content. We only use it as a starting point to apply spectral modifications.

Truncated Fourier Series (TFS)

We generate a spectrally compact IC by filtering white noise in the Fourier domain. Let $\hat{\mathbf{u}} = \mathcal{F}(\mathbf{u}_{\text{noise}})$. We apply a three-dimensional binary mask $M(\mathbf{k})$ which is 1 if $\mathbf{k} \in [0, k_{\text{limit}}]^3$ and 0 otherwise:

$$\mathbf{u}_{\text{TFS}} = \mathcal{F}^{-1}(M \odot \hat{\mathbf{u}}). \tag{9}$$

This ensures energy is distributed only among specific low-to-mid frequency modes. Since the mask does not alter the spectrum within its active region, energy is equally distributed among all active modes. We draw the limit k_{limit} from a discrete uniform distributions with extents $k_{\min}$ and $k_{\max}$.

Diffused Noise (DN)

To generate smooth fields with physically natural decay, we initialize white noise and integrate the linear diffusion operator $\partial_t u = \nu\Delta u$ for a single time step of size $\Delta t = 1$. The resulting magnitude spectrum follows an exponential quadratic decay:

$$|\hat{u}(\mathbf{k})| \propto \exp(-\nu\|\mathbf{k}\|_2^2). \tag{10}$$

Table 14: These hyperparameter configurations for the initialization schemes are used to instantiate the initial condition distribution. Normalization bounds are used to clamp the initial condition and to keep their order of magnitude consistent.

Config Name	Parameters	Normalization Bounds
TFS-A	$k_{\text{min}}, k_{\text{max}} = \{3, 5\}$	Fixed: $[-1, 1]$
TFS-B	$k_{\text{min}}, k_{\text{max}} = \{3, 9\}$	Fixed: $[-1, 1]$
TFS-C	$k_{\text{min}}, k_{\text{max}} = \{5, 9\}$	Fixed: $[0, 1]$
TFS-D	$k_{\text{min}}, k_{\text{max}} = \{3, 9\}$	Random: $\{c_{\text{min}}, c_{\text{max}}\} \sim \{\mathcal{U}(-1, -0.1), \mathcal{U}(0.1, 1)\}$
DN-A	$\nu \sim \mathcal{U}(0.001, 0.01)$	Fixed: $[-1, 1]$
DN-B	$\nu \sim \mathcal{U}(0.001, 0.005)$	Fixed: $[-1, 1]$
DN-C	$\nu \sim \mathcal{U}(0.001, 0.005)$	Random: $\{c_{\text{min}}, c_{\text{max}}\} \sim \{\mathcal{U}(-1, -0.1), \mathcal{U}(0.1, 1)\}$
DE-A	$\alpha \sim \mathcal{U}(-5, -2)$	Fixed: $[-1, 1]$
DE-B	$\alpha \sim \mathcal{U}(-5, -2)$	Random: $\{c_{\text{min}}, c_{\text{max}}\} \sim \{\mathcal{U}(-1, -0.1), \mathcal{U}(0.1, 1)\}$

Table 15: To achieve maximal spectral diversity, we pair specific initializers and certain PDE systems. See Table 14 for the specifics of each initialization configuration.

Equation	Eligible Initializers
Diffusion	TFS-D, DN-C, DE-B, P-TFS-D
Hyper-Diffusion	TFS-D, DN-C, DE-B, P-TFS-D
Burgers	TFS-A, DN-A, DE-A, P-TFS-A
KdV	TFS-A, DN-A, DE-A, P-TFS-A
KS	GN (Gaussian Noise)
KPP-Fisher	TFS-C
Swift-Hohenberg	TFS-B, DN-B, DE-A, P-TFS-B

We vary the smoothness of the IC by sampling the diffusivity parameter ν.

Decayed Energy (DE)

This initializer creates Gaussian Random Fields (GRF) with a specific power-law spectral density. We explicitly enforce the amplitude of the Fourier modes to follow $|\hat{u}(\mathbf{k})| \propto \|\mathbf{k}\|_2^{-\alpha}$, where the exponent α is drawn uniformly from the range $(-5, -2)$. This directly controls the field's roughness.

Poisson (P)

We solve a Poisson equation $\Delta u = -f$, where the source term f is generated via the Truncated Fourier Series (TFS) method described above. In the spectral domain, inversion of the Laplacian acts as a low-pass filter, scaling the spectrum by $\|\mathbf{k}\|_2^{-2}$. This results in smoother initial conditions than the raw TFS source. Since Poisson inversion on periodic boundaries does not move energy between the modes, we retain the characteristics of a compact spectrum (i.e., non-zero energy only in the modes leftover by the mask). However, within this patch, the magnitude follows an isotropic polynomial decay. We denote these configurations as P-TFS.

B.1.4 Spectral distributions and the beneficial effect of cropping

The combinations of partial differential equations, initialization distributions, and integration horizons were specifically chosen to expose the foundation model to a wide range of plausible physics states. To confirm this, we scraped 100,000 samples from a simulation server and analyzed their spectra. This equals $\approx 3.5\,\text{TB}$ of full-resolution simulation data, and this is approximately a tenth of what the Tadpole B-size model was exposed to during its pre-training.

The analysis is based on the random 64^3 crops of data used to train the model. Since those crops are no longer ensured to be periodic, we first apply a Hann window per dimension

$$w(n) = 0.5 - 0.5\cos\left(\frac{2\pi n}{N-1}\right) \qquad 0 \le n \le N-1 \tag{11}$$

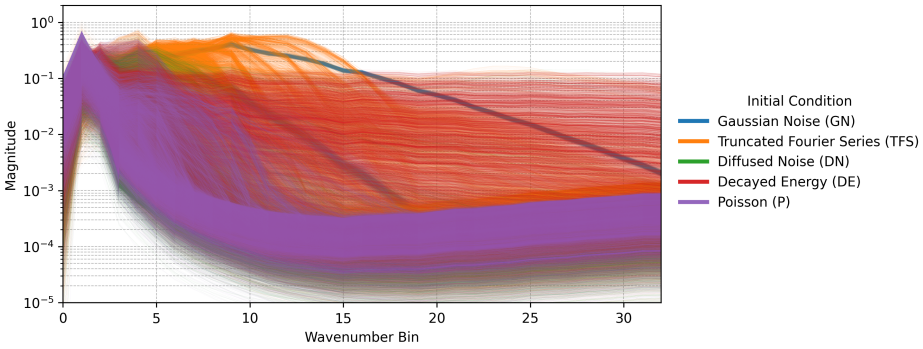

Figure 45: This collection of all *shell-aggregated* spectra across 100,000 samples (about one tenth of the pre-training amount) from the simulation server highlights the diversity of states exposed to the foundational pre-training of Tadpole.

before applying the Fourier transform. While this smoothes the cropped field, it ensures alias-free spectral analysis. We radially aggregate the magnitude of the Fourier coefficients. For example, bin 3 contains the sum of all magnitudes of Fourier coefficients such that $\|\mathbf{k}\|_2 \in [2.5, 3.5)$. The results of this, separated by PDE, resolution, and initializer, are presented in Fig. 46. We also show an aggregated version across all PDEs and simulation resolutions in Fig. 45.

We see that the linear equations (Diffusion & Hyper-Diffusion) retain the characteristic spectra of their respective initializers, but also further diffuse them. The Burgers equation develops noticeable shocks, as evidenced by a richer spectral content than in the initialization. Also, the pattern-forming KPP-Fisher and the Swift-Hohenberg equation display rich spectral content characteristic of their polynomial nonlinearities. As expected, the Kuramoto-Sivashinsky in its chaotic state attains a characteristic spectrum that depends on the domain extent L.

Interestingly, we see that using different simulation resolutions with consistently sized crops exposes the model to the same phenomena at different scales. This is most noticeable for the KS equation, which develops the same spectrum, independent of the resolution. However, because the crops in the high-resolution simulation occupy a small physical space, the model sees a smoother version of that same pattern. We hypothesize that this relationship of similar patterns across different resolutions enables the model to fundamentally understand and be applied at varying resolutions.

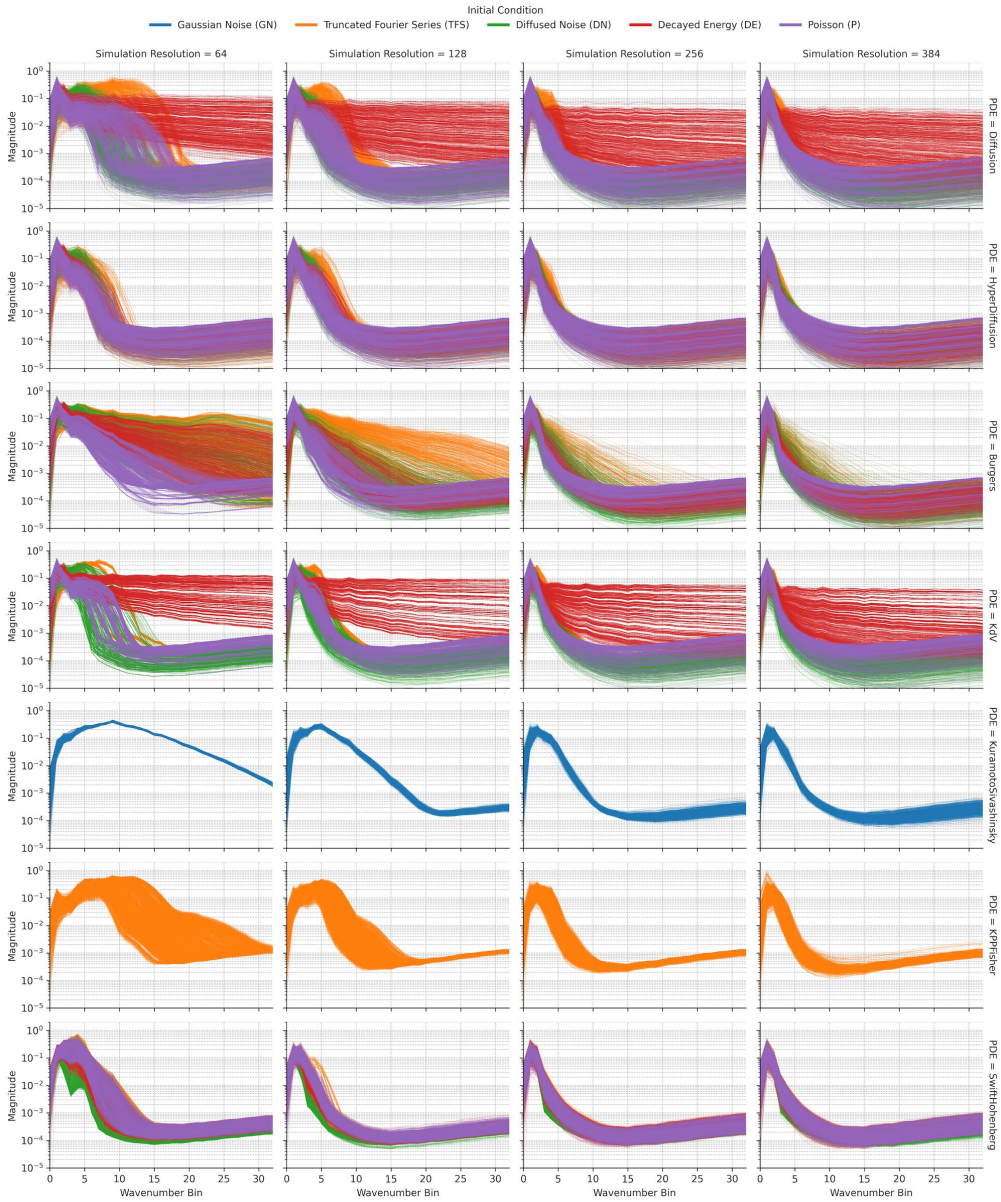

Figure 46: Distribution of Fourier coefficient magnitude across radially aggregated bins for a spectral analysis of the 64^3 training crops based on 100,000 samples from the simulation server (about 10% of the amount of data used for Tadpole B-size pre-taining). Each row represents a different PDE (according to Table 11) and each column a different simulation resolution $\{64, 128, 256, 384\}$. Colors indicate different initial condition distributions, see Table 14 and Table 15. The states produced cover a large range of plausible physics spectra.

B.2 Online learning framework

B.2.1 Sampling strategies and data pipelines

Since the pre-training objective relies on reconstructing physical fields across a vast manifold of plausible physics states, we devised a procedural sampling strategy that maximizes the diversity of states $\mathbf{u}_t$ while maintaining a balanced distribution of physical phenomena. We employ a decoupled client-server architecture: a dedicated simulation server continuously synthesizes physical trajectories and pushes individual frames to an asynchronous First-In-First-Out (FIFO) queue. The training

clients consume data from this queue, ensuring that the computationally expensive simulation steps do not bottleneck the GPU training throughput. For a detailed breakdown of the communication protocol, see Appendix B.2.2. The procedural generation logic is formalized in Algorithm 1 and detailed below.

Continuous Generation Cycle The simulation server operates in an infinite loop, cycling through the set of available PDE systems (e.g., Burgers, Swift-Hohenberg) and the set of resolutions $\mathcal{N} = \{64, 128, 256, 384\}$ in random order. By systematically varying the simulation resolution N, we ensure that the training crops expose the model to spectral features at different scales, mimicking the multi-scale nature of downstream tasks.

Throughput Standardization and Channel Batching To maintain consistent tensor shapes and maximize computational efficiency, we standardize the generation process around three-channel inputs.

- **Vector Fields:** Systems such as the 3D Burgers equation naturally possess three channels ($C = 3$). These are integrated as a single dependent system.
- **Scalar Fields:** For single-component systems ($C = 1$, e.g., Diffusion, Fisher-KPP), we instantiate three independent initial conditions in parallel. These are batched along the channel dimension to form a pseudo-3-channel tensor, $\mathbf{u}_t \in \mathbb{R}^{3\times N\times N\times N}$.

Upon completion of a time-step, the channels are decoupled and pushed to the queue individually. *Exception:* The Kuramoto-Sivashinsky (KS) equation is integrated as a single channel without parallel batching. Consequently, the KS equation contributes proportionally less data volume (approx. 1/3) compared to other phenomena.

Transient Dynamics (Physics Warmup) For some of the PDE systems, we only produce frames within the physically most meaningful regime. In the case of the Burgers equation, this would be in the shock formation and propagation phase. For the KS equation, this is within the chaotic attractor. To ensure we produce samples in these parts of the physics state space, we implement a *physics warmup* phase. For every trajectory, the simulator integrates for W steps (see Table 13), which are strictly discarded. Data recording commences only after this period.

Queue Pre-filling (Server Warmup) Distinct from the physics warmup, we employ a *server warmup* strategy to ensure the initial data distribution is sufficiently diverse. We define a server round counter r and a target threshold R. During the startup phase ($r < R$), we artificially truncate the recorded trajectory length by an early-stop ratio $\xi = \min(r/R, 1)$. This increases the turnover rate of simulators, rapidly filling the buffer with diverse physics states.

Trajectory Balancing and Re-initialization Regardless of the underlying physics, each simulator runs for a specific number of repetitions (**Num Runs**, see Table 13) such that it contributes exactly 30 distinct time steps to the dataset before being discarded. Once this quota is met, the simulator is torn down and a new system is instantiated with fresh constitutive parameters and initial conditions.

Spatial Subsampling and Normalization To optimize bandwidth, we extract random spatial crops $\mathbf{C}$ of size $H' \times H' \times H'$ (where $H' = 96$). If the native simulation resolution is $N = 64$, the full frame is transmitted. Finally, to stabilize the input distribution, each crop is clamped to the precomputed value ranges defined in Table 12.

Fault Tolerance and Checkpointing To support long-running training jobs on preemptible clusters, the simulation server is designed to be fully checkpointable. We serialize the active simulator configurations. This ensures that training can be paused and resumed deterministically without altering the data distribution or repeating sequences.

B.2.2 Buffer and communication strategy

To bridge the gap between high-speed numerical solvers and deep learning frameworks, we implemented a custom asynchronous data loading pipeline. This system abstracts the continuous simulation stream into a PyTorch-compatible dataset, allowing the training loop (managed via PyTorch Lightning) to interface with the procedural generators as if they were a standard static dataset.

Architecture and Communication The pipeline operates on a Producer-Consumer model implemented via `torch.multiprocessing`. A dedicated subset of GPU resources is assigned to the

Algorithm 1 Procedural Data Generation Loop

Require: Set of Equations $\mathcal{E}$, Set of Resolutions $\mathcal{N} = \{64, 128, 256, 384\}$
Require: Server Warmup Rounds $R = 10$, Crop size $H' = 96$, Server Queue $\mathcal{Q}$
Set simulation round counter $r = 0$
while Server Running **do**
  $r \leftarrow r + 1$
  for $\{E, N\}$ in all combinations of $\mathcal{E} \times \mathcal{N}$ **do**
    1. Simulator Configuration
    Sample constitutive parameters (e.g., ν, ζ) per Table 11
    Retrieve discretization settings (Δt, save-freq, value-range) per Table 12
    Retrieve trajectory settings (physics-warmup W, length T, num-runs) per Table 13
    Instantiate ETDRK time-stepper $\mathcal{P}$

    for 1 : num-runs **do**
      // Initialize States
      for $b = 1 : 3$ **do**
        {For KS equation, only $b = 1$ is executed}
        Sample IC type $\mathcal{I}$ and hyper-parameters per Table 15
        Generate $\mathbf{u}_0^{(b)} \sim \mathcal{I}$ and apply IC normalization
      end for
      Stack initial states: $\mathbf{u}_0 \leftarrow \text{Concat}(\mathbf{u}_0^{(1)}, \ldots, \mathbf{u}_0^{(3)})$

      // Time Integration
      Reset Simulator with $\mathbf{u}_0$
      Compute server early-stop ratio $\xi = \min(r/R, 1)$
      $T_{total} \leftarrow (T \cdot \text{SaveFreq} \cdot \xi) + W$
      for $1 : T_{total}$ **do**
        $\mathbf{u}_{t+\Delta t} \leftarrow \mathcal{P}(\mathbf{u}_t)$ {Step via ETDRK (Eqs. (6) and (7))}
        if $t > W$ **and** $(t - W) \pmod{\text{SaveFreq}} == 0$ **then**
          for $c = 1 : 3$ **do**
            // Post-processing & Transport
            $\mathbf{C} \leftarrow (N > H')$? RandCrop$(\mathbf{u}_{t+\Delta t}[c], H')$: $\mathbf{u}_{t+\Delta t}[c]$
            Clamp $\mathbf{C}$ to limits per Table 12
            Push $\mathbf{C}$ to $\mathcal{Q}$
          end for
        end if
      end for
    end for
  end for
end while

Producer role (simulation), while the remaining GPUs function as *Consumers* (training). Communication between these processes is handled via asynchronous thread-safe queues. Similar to the active learning framework Melissa [52], our approach bypasses the file system entirely. Data is passed directly through shared memory or TCP sockets (depending on the node topology), eliminating disk I/O latency. To further reduce the communication overhead, we employ a "transport crop" strategy: simulation frames are cropped to an intermediate size of $H'_{X,Y,Z} = 96^3$ before transmission, as outlined in Appendix B.2.1. This significantly reduces payload size compared to full-resolution grids while remaining larger than the final training crop, enabling further data augmentation/cropping on the consumer side.

Multi-Stage Buffering and Latency Hiding Although spectral solvers are computationally efficient, network latency and synchronization overheads can cause pipeline stalls. To mitigate this, we employ a hierarchical buffering strategy:

1. **Transmission Queue (FIFO):** The simulation server pushes completed transport samples into a finite-sized First-In-First-Out buffer. If this buffer fills up, the simulator pauses,

preventing memory overflows. From this queue, data is sent to all participating training GPUs in a round-robin fashion.

2. **Local Staging Buffer (FIFO):** Each training GPU maintains an incoming "mailbox" queue. New frames are received here before being processed for the training cache.
3. **Consumer Cache (MFU):** On the training side, frames are moved from the staging buffer into a larger local cache governed by a Most-Frequently-Used (MFU) replacement policy. Background threads continuously replenish this cache.

The training loop samples batches from the Consumer Cache rather than the stream directly. This decouples the training step time from the simulation step time. Consequently, even if the simulator throughput fluctuates (e.g., due to varying solver times based on resolution or other overhead), the trainer always has immediate access to data.

Epoch Definition in Infinite Streams In this procedural paradigm, the concept of an "epoch", traditionally seen as one full pass over a static dataset, becomes ill-defined. We redefine an epoch as a fixed number of samples seen during training, which we set to $13'200$.

Numerical Stability Guardrails Given the stochastic initialization of parameters, numerical instabilities (divergences) are rare but possible. To prevent invalid gradients from propagating into the model weights, we implement a strict fail-safe mechanism. Before any frame enters the transmission queue, it is scanned for `NaN` or `Inf` values. If a numerical anomaly is detected:

1. The corrupted trajectory is immediately discarded.
2. The specific simulator instance responsible is reset with a new random seed and parameters.
3. A global error counter is incremented.

If the error counter exceeds a tolerance threshold (set to 10 events per training run), the entire training process is halted to allow for debugging. This ensures that the model is never exposed to corrupted gradients.

Multi-Node and Distributed Training Our default configuration utilizes a single node with four GPUs (1 Producer, 3 Consumers). Under PyTorch Lightning's data-parallel strategy, microbatches are distributed among the consumers, and gradients are synchronized via AllReduce. We also experimented with multi-node configurations by replicating the topology: each node instantiates its own local Producer GPU alongside its local Consumers. This *Local-Producer Strategy* offers a significant bandwidth advantage. By confining the transmission of high-dimensional simulation tensors to the intra-node PCIe bus, we ensure that inter-node interconnects (e.g., InfiniBand) are reserved exclusively for gradient synchronization. This demonstrates the practical scalability of procedural online training, effectively bypassing both disk I/O and inter-node bandwidth bottlenecks.

B.3 Downstream datasets

For the downstream tasks, we include 4 challenging datasets. Details of these datasets are summarized as follows:

`Iso` contains a direct numerical simulation of homogeneous isotropic turbulence from Johns Hopkins Turbulence Database (JHTDB, Open Data Commons Attribution License (ODC-By)) [69], in which the statistical properties are invariant under translations and rotations of the coordinate system. We sample 500 frames from the original dataset. In the autoencoding task, random 64^3 crops are generated from the first 420 frames for training. And we select 3 complete 1024^3-resolution frames from the remaining 80 for testing. In the dynamics learning task, random 128^3 crops are still generated from the first 420 frames for training, and the testing crops are generated from the remaining 80 frames. Below is a brief summary of the key characteristics of the dataset:

- Spatial resolution: $X = 1024, Y = 1024, Z = 1024$
- Spatial size: $[0, 2\pi] \times [0, 2\pi] \times [0, 2\pi]$
- Reynolds number: $Re = 433$
- State variables: x/y/z components of velocity and pressure.
- Time step between stored data: 0.002

- Boundary conditions: periodic

`TCF` contains 21 simulations with Reynolds numbers ranging from $Re = 400$ to $Re = 800$ simulated with PICT [70]. Each simulation contains 200 snapshots. In the autoencoding task, random 48^3 crops are generated from the first 20 simulations for training. We select 200 full-resolution frames from the remaining 1 simulation for testing. In the generative modeling task, the latent flow matching models are trained in the latent space of the first 20 simulations. Below is a brief summary of the key characteristics of the dataset:

- Spatial resolution: $X = 96, Y = 96, Z = 192$
- Spatial size: $[-1, 1] \times [-1, 1] \times [-\pi, \pi]$
- Reynolds number: $Re \in [400, 800]$
- State variables: x/y/z components of velocity.
- Time step between stored data: 0.1
- Boundary conditions: periodic (x), wall (y,z)

`MHD` contains 100 frames sampled from the magnetohydrodynamics turbulence simulation of the JHTDB [69]. We generate crops of size 512^3 from the original 1024^3 simulations. In the autoencoding task, random 64^3 crops are generated from the first 80 frames for training. And we select complete 512^3-resolution frames from the remaining 20 for testing. Below is a brief summary of the key characteristics of the dataset:

- Spatial resolution: $X = 512, Y = 512, Z = 512$ (cropped from $1024 \times 1024 \times 1024$)
- Spatial size: $[0, \pi] \times [0, \pi] \times [0, \pi]$ (cropped from $[0, 2\pi] \times [0, 2\pi] \times [0, 2\pi]$)
- Reynolds number: $Re = 186$
- State variables: x/y/z components of velocity, pressure, x/y/z components of magnetic field, x/y/z components of vector potential
- Time step between stored data: 0.025
- Boundary conditions: crop

`TBL` contains 940 frames sampled from the transitional boundary layer simulations of the JHTDB [69]. We generate crops of size 224^3 from the original $10240 \times 1536 \times 2048$ simulations. In the autoencoding task, random 32^3 crops are generated from the first 840 frames for training. And we select complete 224^3-resolution frames from the remaining 100 for testing. In the dynamics learning task, random 128^3 crops are still generated from the first 840 frames for training, and the testing crops are generated from the remaining 100 frames. The pressure channel is not included in the dynamics channel because the original data is not well normalized for it. Below is a brief summary of the key characteristics of the dataset:

- Spatial resolution: $X = 224, Y = 224, Z = 224$ (cropped from $10240 \times 1536 \times 2048$)
- Spatial size: $[293.2, 314.3] \times [0, 3.9] \times [0, 26.25]$ (cropped from $[0, 969.8] \times [0, 26.5] \times [0, 240]$)
- Reynolds number: $Re = 800$
- State variables: x/y/z components of velocity, pressure
- Time step between stored data: 1.25
- Boundary conditions: crop

C Training details and network architectures

C.1 Network architectures

We build the backbone of Tadpole based on P3D [4]. Fig. 47 illustrates the architecture of the network, and Table 16 summarizes the hyperparameters used for Tadpole with different sizes. Compared to the original P3D architecture, the embedding layers for PDE parameters and skip connections are removed, and an additional convolutional layer is appended to the encoder to project its output to the mean and log-variance of the latent distribution. The discriminator network $\mathcal{A}$ adopts the same architecture as the encoder, but removes the final convolutional projection layer. This makes $\mathcal{A}$ a patch-based discriminator, and we utilize the mean of its output as the final belief.

Table 16: Architecture hyperparameters of Tadpole with different sizes. The definition of each hyperparameter can be found in Fig. 47

	S	B	L
FED	[32, 32, 64]	[64, 128, 128]	[128, 256, 256]
n	2	2	2
r	2	2	2
g	16	32	32
depth	[2, 2, 2, 2, 2]	[2, 2, 2, 2, 2]	[2, 2, 2, 2, 2]
heads	[4, 4, 4, 4, 4]	[4, 4, 4, 4, 4]	[8, 8, 8, 8, 8]
window	[4, 4, 4, 4, 4]	[4, 4, 4, 4, 4]	[4, 4, 4, 4, 4]
mlp ratio		4	

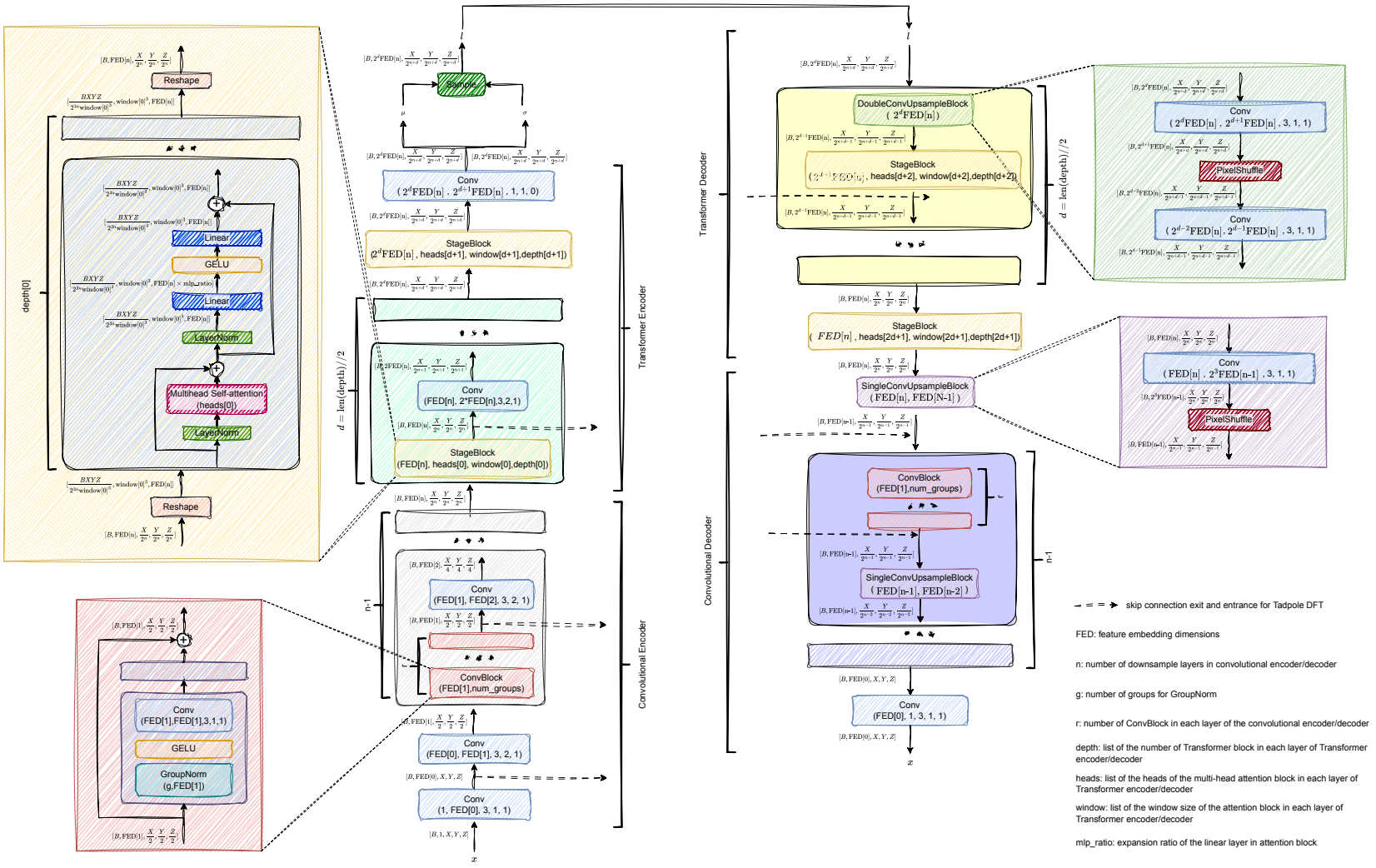

Figure 47: Network architecture of Tadpole based on P3D [4].

For the sub-network $\mathcal{S}$ used in the current experiments, we use a standard encoder-only transformer architecture. Fig. 48 illustrates the architecture of the network, and Table 17 summarizes the hyperparameters with different sizes.

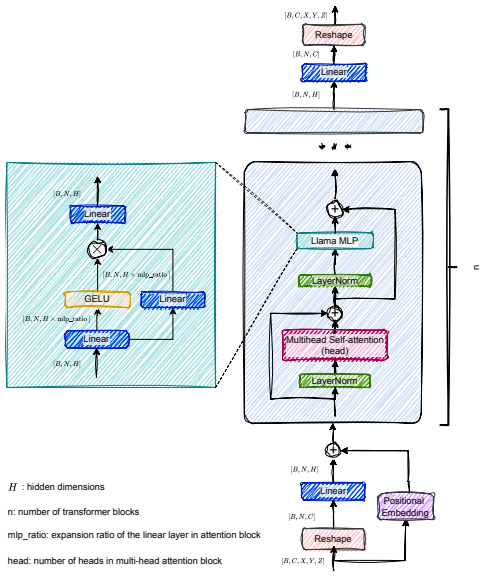

Figure 48: Network architecture of the sub-network $\mathcal{S}$.

Table 17: Architecture hyperparameters of $\mathcal{S}$ with different sizes. The definition of each hyperparameter can be found in Fig. 48

	S	B	L
H	144	176	224
n	4	6	8
head		8	
mlp ratio		4	

C.2 Training objective

The loss function for the VAE consists of three terms: a reconstruction loss, a KL-divergence regularization term, and an adversarial loss term weighted by $\lambda_{\mathcal{A}}$. The discriminator loss function encourages correct classification of real and reconstructed samples. For the adversarial loss, the discriminator $\mathcal{A}$ outputs a scalar score $\mathcal{A}(\mathbf{u}_t)$ indicating the authenticity of the current state $\mathbf{u}_t$. The discriminator is trained using a hinge loss [71–73], and the overall objectives for the VAE and the discriminator are defined as follows:

$$\begin{aligned}\mathcal{L}_{VAE} =& \mathbb{E}_{p_{\mathcal{E}}(\mathbf{z}_t|\mathbf{u}_t)}\left[-\log p_{\mathcal{D}}(\mathbf{u}_t|\mathbf{z}_t)\right] \\ &+ \lambda_{\text{KL}}\text{KL}(p_{\mathcal{E}}(\mathbf{z}_t|\mathbf{u}_t)||q(\mathbf{z}_t)) \\ &- \lambda_{\mathcal{A}}\mathbb{E}_{p_{\mathcal{E}}(\mathbf{z}_t|\mathbf{u}_t)}[\mathcal{A}(\mathcal{D}(\mathbf{z}_t))]\end{aligned} \tag{12}$$

$$\mathcal{L}_{Dis} = \mathbb{E}_{p(\mathbf{u}_t)}[\max(0, 1 - \mathcal{A}(\mathbf{u}_t))] + \mathbb{E}_{p_{\mathcal{E}}(\mathbf{z}_t|\mathbf{u}_t)}[\max(0, 1 + \mathcal{A}(\mathcal{D}(\mathbf{z}_t)))] \tag{13}$$

For the KL-divergence term in Eq. (12), we set $\lambda_{\text{KL}} = 10^{-6}$. For the adversarial loss term, we use a gradient-based scale strategy [48] for λ_{adv} with a maximum scale value of 10^{-4}. The discriminator will only be trained when the L2 reconstruction loss is below a threshold of 0.001 to stabilize training. After the start of training, the feedback from the discriminator will not be added to Tadpole's training

directly until a 1000-iteration learning rate warm-up stage, followed by another 1000 iterations of warm-up for λ_{adv}.

C.3 Training hyperparameters

Table 18: Training hyperparameters of Tadpole

Training		Batch Size	Learning Rate	Iterations
Pre-training	S	48	5×10^{-5}	550000
	B			825000
	L		5×10^{-6} for $\mathcal{E}$ 5×10^{-5} for $\mathcal{D}$ and $\mathcal{A}$	1700000
Downstream task	Autoencoding	32	5×10^{-5}	14000
	Dynamics Learning	32	2×10^{-4}	56000
	Generative Modeling	256	1×10^{-4}	26400

This section summarizes the training hyperparameters for Tadpole. The primary values are presented in Table 18.

Pre-training: Pre-training is conducted in bf16-mixed precision using the AdamW optimizer [74] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and a weight decay of 10^{-15}. A loss-adaptive learning rate scheduler reduces the learning rate by a factor of 0.5 when the training loss decreases by an order of magnitude below the previous threshold. The initial and minimal learning rates are set to 5×10^{-5} and 5×10^{-6}, respectively. A linear learning rate warm-up is applied during the first 1000 iterations for both Tadpole and the discriminator. The KL-divergence term in Eq. (12) is optimized solely by the encoder and becomes increasingly unstable as network size increases. Therefore, the initial learning rate of the Tadpole-L encoder is reduced to 5×10^{-6}. Different-sized Tadpoles are pre-trained with varying numbers of training iterations, as larger models require more iterations to converge. The training iterations for S, B, and L-size models are 5.5×10^5, 8.25×10^5, and 1.7×10^6, respectively. The batch size for pre-training is 48, with gradient accumulation employed to reduce VRAM consumption.

Downstream autoencoding: The downstream autoencoding uses the same hyperparameters as pre-training, except the batch size is reduced to 32 and the number of training iterations is set to 1.4×10^4.

Downstream dynamics: The same hyperparameter configuration is applied to both `Iso` and `TCF` datasets. Training is performed in bf16-mixed precision using the AdamW optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and a weight decay of 10^{-15}. The learning rate is fixed at 2×10^{-4}. The number of training iterations is 5.6×10^3. The batch size is 32, with gradient accumulation used to reduce VRAM consumption.

Downstream generative modeling: Generative modeling training is conducted in bf16-mixed precision using the AdamW optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and a weight decay of 10^{-15}. The learning rate is fixed at 1×10^{-4}. The number of training iterations is 2.64×10^3. The batch size is set to 256, as the latent generative model requires less training memory. For models trained in pixel space, gradient accumulation is used to reduce memory consumption.

C.4 Training cost

Training foundation models incurs a substantial computational cost. Current Tadpole training involves multiple systems with different hardware configurations. Below, we summarize the training costs for each model in terms of GPU hours. Note that if the model is trained with multiple GPUs in parallel, the total GPU hours are estimated by multiplying the actual training hours by the number of GPUs. Meanwhile, the training costs across different downstream datasets are similar as we typically use a fixed number of training iterations. Below, we show the average training cost estimates across datasets and runs. Central cost factor for pre-training is the model size:

- Tadpole S Pre-training: 372 GPU hours with L40S GPUs.
- Tadpole B Pre-training: 620 GPU hours with A100 GPUs.
- Tadpole L Pre-training: 2300 GPU hours with A100 GPUs.

Training via fine-tuning is substantially faster, but likewise directly scales with model size:

- Tadpole B fine-tuning for autoencoding task (FPFT/Scr.): 40 GPU hours with L40S GPUs
- Tadpole B fine-tuning for autoencoding task (LoRA 32): 36 GPU hours with L40S GPUs
- Tadpole S for dynamics task (Tadpole-DFT): 64 GPU hours with L40S GPUs
- Tadpole B for dynamics task (Tadpole-DFT): 120 GPU hours with L40S GPUs
- Tadpole B for dynamics task (FPFT/Scratch): 110 GPU hours with L40S GPUs
- Tadpole L for dynamics task (Tadpole-DFT): 280 GPU hours with A100 GPUs
- Tadpole latent generative models: 8 GPU hours with L40S GPUs

Nonetheless, even dynamic tasks using the L-size model converged in $8\times$ fewer GPU hours than pre-training. Above, the NVIDIA L40S GPUs were equipped with 48GB RAM, while the A100 GPUs had 40GB RAM.

D Evaluation metrics

D.1 Enstrophy-based evaluation for dynamics learning

Neural networks are known to exhibit spectral bias, favoring the learning of low-frequency components while under-resolving high-frequency structures. This limitation becomes particularly pronounced in autoregressive rollout settings, where prediction errors accumulate over time and manifest as progressive attenuation of high-frequency modes, resulting in overly smooth or blurred solutions. Conventional pixel-space metrics, such as mean squared error, are dominated by large-scale features and may therefore underestimate the degradation of fine-scale structures. To more faithfully assess model performance, especially in long-horizon predictions, we incorporate spectrum-based evaluation metrics that quantify errors across frequency bands. The spectrum-based metrics provide a scale-resolved characterization of model accuracy and explicitly capture the loss of high-frequency content, which is critical in many PDE systems. That's why we emphasize the spectral metrics in the current manuscript.

The spectrum-based evaluation for the dynamic rollout test case of the main text is performed as follows: The enstrophy spectrum at wavenumber $k \in \mathbb{R}_+$ is given by

$$S(k) = \sum_{k<|m|\leq k+1} \frac{1}{2} \sum (|\widehat{\omega_x}(m)|^2 + |\widehat{\omega_y}(m)|^2 + |\widehat{\omega_z}(m)|^2), \tag{14}$$

where $|\widehat{\boldsymbol{\omega}_{x,y,z}}(m)|$, with $m \in \mathbb{Z}^3$, denotes the Fourier coefficients of the vorticity component. To quantify discrepancies between spectra, we compute the NRMSE between the averaged reference spectrum and the averaged spectrum of generated vorticity fields,

$$\text{NRMSE}^{ES} = \sqrt{\frac{\text{mean}_k\big((S_{\text{pred}}(k) - S_{\text{ref}}(k))^2\big)}{\text{mean}_k\big(S_{\text{ref}}(k)^2\big)}} \tag{15}$$

Since the cropped regions are not periodic, discontinuities at the domain boundaries introduce artifacts in the Fourier transform. To mitigate these effects, we apply a Hann window to smoothly attenuate $\boldsymbol{\omega}$ toward the boundaries prior to computing the Fourier coefficients.

D.2 Statistical evaluation for generative modeling

Properly assessing the quality of generative models for scientific data is an open problem. Two central difficulties are that (1) the number of samples in the reference dataset is often small and (2) the dimensionality of the data is very high. The `TCF` dataset comprises three velocity channels at a spatial discretization of $96 \times 96 \times 192$ in 3D. When flattened, this corresponds to a ca. $5M$-dimensional vector. Taking samples from the reference simulation is futher complicated, since snapshots that are close in time are highly correlated, which can have implications for the statistical evaluation, which often assumes that samples are independent. To avoid a high auto-correlation of samples from the reference simulations, we take every 10th sample from the `TCF` dataset for Reynolds numbers in $[400, 500, 600, 700, 800]$, which corresponds to a step size $\Delta t = 1$. This yields 100 reference samples in total. We generate the same number of samples for each generative model.

To simplify the evaluation process, we split a single high-resolution sample into multiple low-dimensional samples. While this means that information on long-range correlation and structure is lost, we consider the distributional metrics on the set of low-dimensional derived samples as a lower bound on the distributional metrics for the high-resolution data.

There are many strategies to reduce the high-resolution samples. We choose a crop-based strategy, which partitions the full $3 \times 96 \times 96 \times 192$-sized data into chunks of size $3 \times 16 \times 16 \times 16$. This transforms a single high-resolution sample into 432 smaller samples, which have dimensionality $12\,288$. The smaller samples are no longer independent, however, we believe that this has a negligible effect on the evaluation. In total, there are $43\,200$ samples with reduced dimensionality.

Besides the NRMSE of the mean and std., we were not able to run the computation on the full set of samples or the full dimensionality due to computation and stability constraints. We denote the maximum number of samples used for computation with n_{points} and the maximum dimensionality with $d_{\max}$, and select the maximum values acceptable for the computation budget. If n_{points} is smaller

Table 19: n_{points} and $d_{\max}$ for different distributional metrics.

	χ^2_{PQM}	$\mathcal{W}_1$	MMD_{RBF}
n_{points}	1000	50000	10000
$d_{\max}$	inf	10000	100

than the dataset size, we randomly sample a subset whose size matches n_{points} without replacement. If $d_{\max}$ is smaller than the dimensionality of the data, we only use the first $d_{\max}$ dimensions. Table 19 shows the exact values of n_{points} and $d_{\max}$ used for different distributional metrics.

E Nomenclature and abbreviations

E.1 Nomenclature

- $\mathbb{P}^i$: The i-th PDE in the PDE family.
- $\mathbf{u}_t$: The PDE solution at time step t.
- $\mathbf{z}_t$: The latent representation of $\mathbf{u}_t$.
- $\mathcal{E}$: The encoder network that maps the current state $\mathbf{u}_t$ to a latent representation $\mathbf{z}_t$.
- $\mathcal{D}$: The decoder network that reconstructs the state from the latent representation $\mathbf{z}_t$.
- $\mathcal{A}$: The adversarial network (discriminator) that distinguishes between real and reconstructed states.
- $\mathcal{S}$: The sub-network used for downstream tasks or specific applications.
- λ_{KL}: The weight for the KL-divergence term in the loss function.
- B: The batch dimension.
- C: The number of channels in the input data.
- X, Y, Z: The spatial dimensions (height, width, depth) of the 3D input data.
- H_i: The final crop size along spatial dimension i.
- H_i': The pre-crop size along spatial dimension i.
- W_0: The pre-trained weights of the Tadpole model.
- r: The LoRA rank used in fine-tuning.
- A, B: The LoRA adaptation matrices.
- D: The dimension of the 3D PDE data, denoted as $D = C \times X \times Y \times Z$.
- γ: The scale factor for the skip connections.

E.2 Abbreviations

- PDE: Partial Differential Equation.
- PEFT: Parameter-efficient Fine-tuning.
- NLP: Natural Language Processing.
- CV: Computer Vision.
- FM: Foundation Model.
- FPFT: Full-parameter fine-tuning.
- `Iso`: Isotropic turbulence test case.
- `TCF`: Turbulent Channel Flow test case.
- `MHD`: Magnetohydrodynamics test case.
- `TBL`: Transitional Boundary Layer test case.
- LLM: Large Language Models.
- ICL: In Context Learning.

- FIFO: First In First Out.
- MFU: Most Frequently Used

F Changelog

Version 1: Initial submission to arXiv on 14 May 2026.

Version 2 (current): Camera-ready version for NeurIPS 2026. Compared to the initial version, the major modifications are listed below:

- Fig. 8 and Fig. 14 add an additional ablation examining the effect of zero initialization of the DFT weights. The results demonstrate that zero initialization further enhances the performance of Tadpole-DFT.
- The dynamics prediction results of Tadpole-L have been updated (Fig. 5 and Tables 3 to 6) following a bug fix in the training script. In our scripts, we employed activation checkpointing to reduce GPU memory consumption when training the L-size model. However, our original implementation contained an issue in how the PyTorch checkpointing function was invoked.

  Specifically, for each encoder and decoder layer, we used `x, res = checkpoint(encoder_level, x)` and `x = checkpoint(decoder_level, x)`, respectively. With the default setting `use_reentrant=True`, PyTorch employs a reentrant checkpointing implementation that runs the forward computation under `no_grad` and requires at least one input and output to participate in gradient computation. Because the input tensor `x` at the checkpoint boundary did not require gradients (`requires_grad=False`), gradients were not properly propagated through the checkpointed encoder. The residual tensors were returned within Python lists, which the reentrant implementation does not recognize as participating in autograd. Consequently, these residuals were detached from the computational graph (`grad_fn=None`), preventing gradient propagation through the skip connections to the encoder parameters.

  We resolved this issue by explicitly setting `use_reentrant=False` in both checkpoint calls: `x, res = checkpoint(encoder_level, x, use_reentrant=False)` and `x = checkpoint(decoder_level, x, use_reentrant=False)`. Unlike the reentrant implementation, non-reentrant checkpointing records the autograd graph and supports tensors nested within Python data structures. After this correction, we verified that the residual tensors had valid `grad_fn` attributes and that all parameter groups exhibited matching gradient norms with and without activation checkpointing.

  The actual performance of Tadpole-L is better than the original value shown in the previous version of the manuscript. Now, Tadpole-L also beats the Walrus model with many fewer trainable parameters in the `Iso` case for one-step predictions, again highlighting the effectiveness of Tadpole pretraining.
- Following the NeurIPS reviewer's request, additional experiments comparing the performance of DPOT and Morph models trained from scratch and their pre-trained counterparts are included in Appendix A.7, which help clarify the contributions of the backbone and pretraining to Tadpole's superior performance.
- Additional experiments evaluating the computation scaling of Tadpole have been included in Appendix A.8. These results further demonstrate that the single-channel cropping strategy reduces the model's memory cost.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The main claims made in the abstract and introduction reflect the paper's contributions and scope.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: The limitations are discussed in Section 6.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [N/A]

Justification: The paper does not include theoretical results.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: We have provided all the technical details needed to reproduce the results in Apps. B to C.

   Guidelines:

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

5. **Open access to data and code**

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

Answer: [Yes]

Justification: The code is available at `https://github.com/tum-pbs/tadpole`. The dataset is too large to share directly, but it is easy to download the original data as described in the Appendix.

Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: We have provided all the technical details needed to reproduce the results in Apps. B to C.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [No]

   Justification: Error bars are not reported due to the high computational expense of 3D experiments and foundation model pre-training runs. However, all experiments use EMA weights during training to reduce the impact of randomness and exhibited stable convergence behavior.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: The training cost of the current study is discussed in Appendix C.4.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The research reported in the paper complies with the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [N/A]

    Justification: This paper presents work to advance the field of machine learning for physical simulations. We don't foresee any significant societal impacts directly associated with our work to be highlighted here.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The current paper poses no such risks.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: The current paper uses datasets from other research work, which are correctly cited in Appendix B.3, and the original licenses have been explicitly mentioned and properly respected.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [N/A]

    Justification: The current submission doesn't release new assets yet. The current pre-trained weights of Tadpole foundation models are available on `https://huggingface.co/thuerey-group/tadpole`.

    Guidelines:

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

14. **Crowdsourcing and research with human subjects**

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.