

---

# Multiple Physics Pretraining for Spatiotemporal Surrogate Models

---

Michael McCabe<sup>\*,1,2</sup>, Bruno Régaldo-Saint Blancard<sup>1</sup>, Liam Parker<sup>1</sup>, Ruben Ohana<sup>1</sup>,  
Miles Cranmer<sup>3</sup>, Alberto Bietti<sup>1</sup>, Michael Eickenberg<sup>1</sup>, Siavash Golkar<sup>1</sup>,  
Geraud Krawezik<sup>1</sup>, Francois Lanusse<sup>1,4</sup>, Mariel Pettee<sup>1,5</sup>,  
Tiberiu Tesileanu<sup>1</sup>, Kyunghyun Cho<sup>6,7,8</sup>, Shirley Ho<sup>1,6,9</sup>

The Polymathic AI Collaboration

<sup>1</sup> Flatiron Institute, <sup>2</sup> University of Colorado Boulder, <sup>3</sup> University of Cambridge,

<sup>4</sup> Université Paris-Saclay, Université Paris Cité, CEA, CNRS, AIM,

<sup>5</sup> Physics Division, Lawrence Berkeley National Laboratory, <sup>6</sup> New York University,

<sup>7</sup> Prescient Design, Genentech, <sup>8</sup> CIFAR Fellow, <sup>9</sup> Princeton University

## Abstract

We introduce multiple physics pretraining (MPP), an autoregressive task-agnostic pretraining approach for physical surrogate modeling of spatiotemporal systems with transformers. In MPP, rather than training one model on a specific physical system, we train a backbone model to predict the dynamics of multiple heterogeneous physical systems simultaneously in order to learn features that are broadly useful across systems and facilitate transfer. In order to learn effectively in this setting, we introduce a shared embedding and normalization strategy that projects the fields of multiple systems into a shared embedding space. We validate the efficacy of our approach on both pretraining and downstream tasks over a broad fluid mechanics-oriented benchmark. We show that a single MPP-pretrained transformer is able to match or outperform task-specific baselines on all pretraining sub-tasks without the need for finetuning. For downstream tasks, we demonstrate that finetuning MPP-trained models results in more accurate predictions across multiple time-steps on systems with previously unseen physical components or higher dimensional systems compared to training from scratch or finetuning pre-trained video foundation models. We open-source our code and model weights trained at multiple scales for reproducibility.

## 1 Introduction

In recent years, the fields of natural language processing and computer vision have been revolutionized by the success of large models pretrained with task-agnostic objectives on massive, diverse datasets (Chen et al., 2020; Devlin et al., 2018; He et al., 2021). This has, in part, been driven by the development of self-supervised pretraining methods which allow models to utilize far more training data than would be accessible with supervised training (Balestrieri et al., 2023). These so-called “foundation models” have enabled transfer learning on entirely new scales. Despite their task-agnostic pretraining, the features they extract have been leveraged as a basis for task-specific finetuning, outperforming supervised training alone across numerous problems especially for transfer to settings that are insufficiently data-rich to train large models from scratch (Bommasani et al., 2021).

---

<sup>\*</sup>Contact: [mmccabe@flatironinstitute.org](mailto:mmccabe@flatironinstitute.org)

Deep learning for computational science has begun to see first steps in this direction. Large domain-specific pretrained models have emerged in diverse fields such as chemistry (Bran et al., 2023; Chithrananda et al., 2020), medicine (Jiang et al., 2023; Tu et al., 2023), astrophysics (Leung & Bovy, 2023; Nguyen et al., 2023b), and climate (Nguyen et al., 2023a) and the trend only seems to be growing as more and more models are developed for new fields both as refined versions of existing large language models and as new models trained entirely on field-specific data.

In this work, we demonstrate that similar approaches can be extended to the surrogate modeling of spatiotemporal physical systems. Spatiotemporal prediction tasks, like those found in fluids, solids, or general continuum mechanics, have attracted significant attention from the deep learning community. From direct prediction methods (Dang et al., 2022; Li et al., 2020; Lusch et al., 2018; Pfaff et al., 2021; Stachenfeld et al., 2022) to neural PDE solvers (Bruna et al., 2022; Raissi et al., 2019), researchers have sought to develop fast, accurate models for physics either as faster surrogates for the partial differential equation (PDE) solvers that dominate the field or to simulate systems that cannot be exactly described or resolved by current mechanistic models and available hardware. While directly outperforming PDE solvers is difficult (Grossmann et al., 2023), deep learning has already begun to impact fields like atmospheric science (Ben-Bouallegue et al., 2023; Bi et al., 2023; Lam et al., 2023; Pathak et al., 2022) and cosmology (Cranmer et al., 2021; He et al., 2019; Jamieson et al., 2023), where the systems are too large or too imprecisely described to be simulated exactly.

Unfortunately, outside of a few observation-rich outliers, settings where numerical simulation is expensive or unreliable also tend to be settings where the difficulty of acquiring training data makes it impractical to train surrogates conventionally. Most deep learning-based surrogates thus far have focused on specific physical systems or families of parameterized PDEs where data can easily be acquired. However, for the low-data settings often found in simulation-driven exploration and design, it would be valuable to have large, task-agnostic models with a broad understanding of common physical behavior to act as a foundation for finetuning.

**Contributions.** To address this need, we introduce *Multiple Physics Pretraining* (MPP), a new approach for task-agnostic pretraining of physical surrogate models. Our method enables large-scale pretraining for transfer across diverse physics which we study using fluid-oriented benchmarks. Our specific contributions are:

- We develop MPP, a pretraining approach in which we embed multiple heterogeneous physical systems into a shared embedding space and learn to autoregressively predict the dynamics of all systems simultaneously.
- We show that single transformer models pretrained with MPP are able to match or surpass modern task-specific baselines without applying task-specific finetuning to the MPP models.
- We demonstrate the transfer capabilities of models trained with MPP to systems with limited training examples (referred to as *low-data systems* thereafter) displaying new physics in the form of previously unseen parameter regimes generating notably different qualitative behavior and inflated to higher dimensions.
- We open-source our code and provide our pretrained models at a variety of sizes for the community to experiment with on their own tasks.

## 2 Background

**Notation.** Let  $S$  be an arbitrary physics-driven spatiotemporal dynamical system, either described by a parameterized family of PDEs with fixed parameters, or where snapshots are gathered from observation of a unique physical phenomenon. To simplify notation, we discuss systems with a single state variable in one spatial dimension. A continuous state variable for system  $S$  is represented as  $u^S(x, t) : [0, L_S] \times [0, \infty) \rightarrow \mathbb{R}$ . We discretize the system uniformly in space and time at resolutions  $N_S, T_S$  respectively. A snapshot  $\mathbf{u}_t^S \in \mathbb{R}^{N_S}$  represents the value of state variable  $u^S$  at all  $N_S$  spatial discretization points at time  $t$ . Our pretraining task is then to learn a single model  $\mathcal{M}$  that can take a uniformly spaced sequence of  $T_S$  snapshots  $\mathbf{U}_t^S = [\mathbf{u}_{t-T_S \Delta t_S}^S, \dots, \mathbf{u}_t^S]$  from system  $S$  sampled from some distribution over systems and predict  $\mathcal{M}(\mathbf{U}_t^S)$  such that  $\mathcal{M}(\mathbf{U}_t^S) \approx \mathbf{u}_{t+\Delta t_S}^S$ .

**Autoregressive Pretraining.** In vision and language, the dominant pretraining strategies include autoregressive prediction (Radford et al., 2018), masked reconstruction (Devlin et al., 2018; He

et al., 2021), and contrastive learning (Chen et al., 2020). In language, autoregressive generation emerged as a convenient self-supervised task. In surrogate modeling of dynamical systems, next-step prediction is often a primary goal. This makes autoregressive pretraining a natural choice of objective for training time-dependent surrogate models.

We note that it is common to use the simulation parameters to condition the predictions of models operating on PDE-generated data (Gupta & Brandstetter, 2022; Subramanian et al., 2023; Takamoto et al., 2023). Our goal is not to develop a new PDE solver, but rather to design an approach that is broadly applicable to both observed and simulated dynamics. Thus, we do not assume a known functional form in MPP and the model must instead implicitly infer the impact of these parameters on the dynamics from the history provided in  $U_t^S$ .

**Surrogate Modeling for Spatiotemporal Physical Systems.** We are primarily concerned with modeling dynamical systems varying in both time and space, where the time evolution of the system is intrinsically tied to spatial relationships amongst the state variables according to physical laws. PDEs are one of the primary modeling tools for this setting. They are often derived from fundamental conservation laws of properties such as mass, momentum, and energy (Farlow, 1993). Many PDEs describe variations of the same physical laws, which is why concepts like diffusion, advection, reactivity, and connections between time and spatial gradients appear in many different PDEs. These shared underlying principles suggest we can extract features relevant to multiple physical systems.

### 3 Related Work

**Foundation models.** Massive pretrained models dubbed “foundation models” (Bommasani et al., 2021), particularly large transformer-based architectures (Vaswani et al., 2017), have recently attracted significant attention. The most prevalent foundation models are pretrained language models like GPT (Brown et al., 2020; Radford et al., 2018, 2019) and BERT (Devlin et al., 2018). Emergent abilities (Wei et al., 2022) demonstrated by large language models highlight the importance of scale in manifesting higher-order capabilities absent at smaller scales. Vision has seen similar developments with the growth of masked (He et al., 2021; Tong et al., 2022) and contrastive (Chen et al., 2020) pretraining. The data in this work is insufficiently diverse to call the resulting models “foundational”. However, we provide the first large-scale implementation of successful multiple nonlinear physics pretraining for spatiotemporal systems.

**Scientific machine learning.** While a wide range of architectures have been employed for physical surrogate modeling (Bar & Sochen, 2019; Han et al., 2018; Sirignano & Spiliopoulos, 2018; Yu et al., 2018; Zang et al., 2020), we position our work with respect to three major classes. One prominent class is the neural-network-as-PDE-solution approach (Bruna et al., 2022; Raissi et al., 2019) which requires the PDE to be known and solves a single system on a single domain. Other methods do not learn the solution directly, but instead augment a PDE-solver as learned corrections (Dresdner et al., 2023; Rackauckas et al., 2021; Um et al., 2021), learned closures (Duraisamy et al., 2019; Sirignano & MacArt, 2023), or learned algorithmic components (Bar & Sochen, 2019; Kochkov et al., 2021). A broader, but less physically constrained approach, is learning a solution operator from the data without knowledge of the governing equations (Cao, 2021; Kovachki et al., 2023; Li et al., 2020, 2021; Lu et al., 2019). While these methods are often evaluated using PDE-generated benchmarks, these are designed to learn directly from data rather than learning to solve a PDE. Neural operators typically do not reach the accuracy of numerical PDE solvers, but they are applicable for domains without explicitly provided equations. This last family is the most similar to our approach, especially Cao (2021) as we use a transformer-based architecture. However, our pretraining procedure is developed for training across multiple operators.

The high cost of training scientific models from scratch has led to significant exploration of transfer learning. Prior work has explored transfer learning in operator networks in such scenarios as conditional shift (Goswami et al., 2022) or new domains, boundary conditions, or distributions over parameters (Li et al., 2021; Subel et al., 2023; Wang et al., 2022a; Xu et al., 2023). However, these too need to be retrained from scratch for new differential operators in the PDE. More recently, efforts have been made to explore transfer across operators and benefits from training on multiple physical systems simultaneously. Subramanian et al. (2023) explores how transfer scales in this setting. However, their study is limited to steady-state linear systems with periodic boundary conditions.

Other works have explored similarly restricted classes or low dimensional, low resolution systems (Desai et al., 2022; Yang et al., 2023).

## 4 Scalable Multiple Physics Pretraining

### 4.1 Compositionality and Pretraining

Many specialized PDEs demonstrate a form of compositionality, as a range of physical phenomena can be described by core components like nonlinear advection or diffusion, but then are augmented or restricted by specialized terms representing concepts like buoyancy or system constraints. To motivate a useful pretraining procedure from this compositionality, we examine two hypotheses:

1. Single models can learn the dynamics for multiple classes of physical behavior.
2. Learning partially overlapping physics is beneficial for transfer learning.

Since many real-world systems share core components, under these hypotheses, training single models on many distinct systems is a natural approach for developing foundation models for physical dynamics. We therefore start by eliminating the complexity related to hypothesis (1) in order to isolate hypothesis (2). We do this by choosing a simple problem setting with one shared scalar field along with consistent scales and geometry: constant-coefficient linear advection-diffusion on a periodic 1D domain. Let  $\psi(x, t)$  be a scalar-valued function defined on a periodic spatial domain,  $v$  a constant one-dimensional velocity coefficient and  $\delta$  a constant diffusion coefficient, then:

$$\begin{aligned} \text{Advection:} \quad & \frac{\partial \psi}{\partial t} + \nabla \cdot (v\psi) = 0 \\ \text{Diffusion:} \quad & \frac{\partial \psi}{\partial t} + \nabla \cdot (-\delta \nabla \psi) = 0 \\ \text{Advection-Diffusion:} \quad & \frac{\partial \psi}{\partial t} + \nabla \cdot (v\psi - \delta \nabla \psi) = 0. \end{aligned}$$

If hypothesis (2) holds, we would expect pretraining on advection and diffusion systems individually could be beneficial for transfer to advection-diffusion systems.

We find that this is indeed the case. We pretrain a spatiotemporal transformer model on a large amount of trajectories (100,000 each) with uniformly sampled coefficients ( $v \in [-3, -1] \cup [.1, 3]$ ,  $\delta \in [10^{-3}, 1.]$ ) generated from the advection and diffusion equations while finetuning on restricted samples from advection-diffusion simulations. The pretrained model is able to achieve much lower error with far fewer samples (Figure 1) without observing advection and diffusion together in the same trajectory during pretraining.

However, to validate hypothesis (1), we must handle much larger spatial resolutions, varying scales, and heterogeneous relationships between fields. Over the rest of this section, we develop an approach for handling these challenges.

### 4.2 Architecture

**Axial Attention.** Given the success of large transformer models in other domains, we employ a scalable axial attention (Dong et al., 2022; Ho et al., 2019; Huang et al., 2019) transformer backbone. For a (2+1)-dimensional system with  $T \times H \times W$  tokens, conventional dense attention attends over all tokens simultaneously and has cost  $O((HWT)^2)$ . Axial attention instead performs a series of attention operations over each axis in turn, limiting the cost to  $O(H^2 + W^2 + T^2)$ . In Figure 2, it can be seen that while we perform attention on each axis independently, the spatial  $K$ ,  $Q$ ,  $V$  projections are shared between the height (y) and width (x) axes.

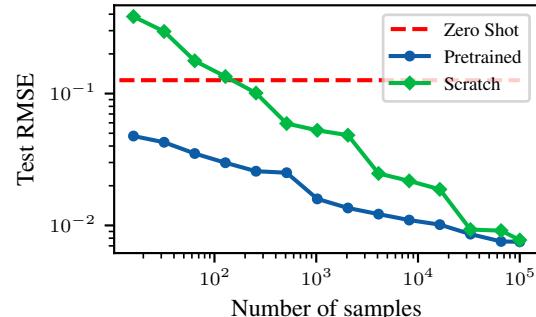


Figure 1: Finetuning a model pretrained on large amounts of advection and diffusion data outperforms models trained from scratch on advection-diffusion data across a wide range of data availability (16-100K examples).

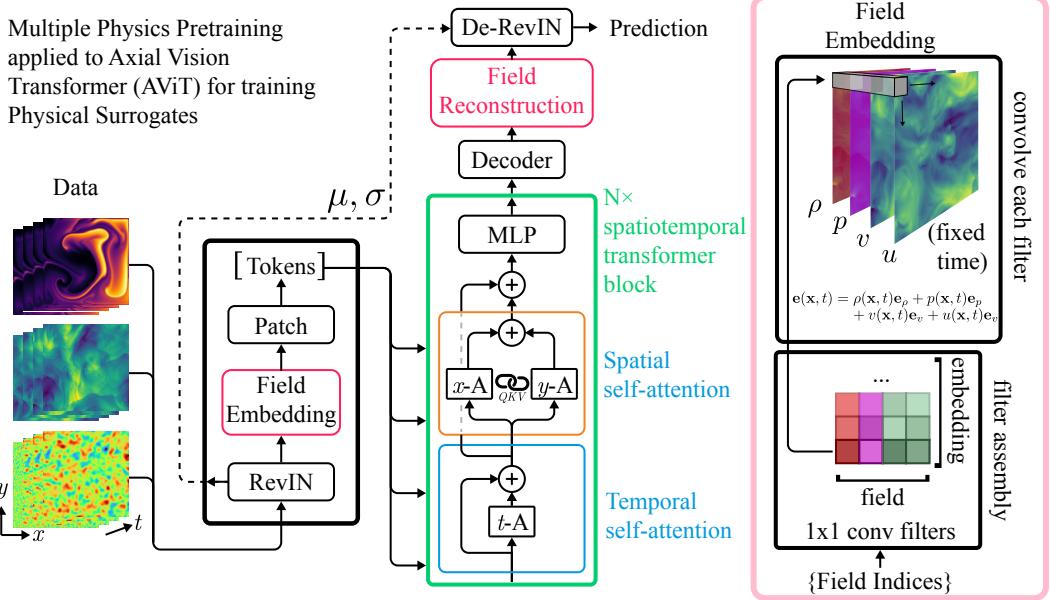


Figure 2: (Left) MPP works by individually normalizing each example using Reversible Instance Normalization (RevIN) then embedding each field individually into a shared, normalized space. A single transformer backbone can then predict the next step for multiple sets of physics. We use an AViT backbone which attends over space and time axis sequentially. Spatial attention is further split by axis, though these share linear projection weights. (Right) The embedding and reconstruction matrices are formed by subsampling a larger  $1 \times 1$  convolutional filter based on input fields.

Axial attention has been used in video transformers (Arnab et al., 2021; Bertasius et al., 2021) due to the improved scalability in higher dimensions. While the tools used in our transformer backbone were introduced in prior work, our choice of using fully axial attention differs from ViViT which opted to only separate space and time attention. We favor scalability over maximizing accuracy and so chose fully axial attention. In the following, we refer to this architecture as an Axial ViT (AViT).

**Field Embedding and Normalization.** Embedding multiple physical systems into a single shared representation is complicated by the fact that fields from different systems may operate on entirely different scales in terms of both magnitude and resolution. This is one of the primary challenges that must be addressed for multiple-physics pretraining.

To unify magnitudes, we use Reversible Instance Normalization (Kim et al., 2022, RevIN). We compute the mean and standard deviation of each channel over space-time dimensions and use them to normalize input fields. These statistics are saved and used to denormalize model outputs. While this approach was initially developed for time-series forecasting, in practice the effect is similar to that of the norm scaling utilized in Subramanian et al. (2023).

After rescaling, the data is projected into a shared embedding space. This is the only component with unique weights for each source system. Given a system  $S$  with state variables  $u(x, t), v(x, t), p(x, t) \in \mathbb{R}$ , we project each point or “pixel” into a space of dimension  $D^{\text{emb}}$ :

$$e(x, t) = u(x, t)e_u + v(x, t)e_v + p(x, t)e_p \quad (1)$$

where  $e$  are embedding vectors in  $\mathbb{R}^{D^{\text{emb}}}$ . This can be seen as a convolution with  $1 \times 1$  filters where the input channels of the filter are sub-selected to correspond to the fields present within a given dataset. On the right side of Figure 2, the filter is assembled by sub-selected columns of the larger filter corresponding to the provided fields. It is important to note that this initial projection setup is amenable to fine-tuning to unseen field types. This can be achieved by adding new channels to the initial embeddings, and training them from random initialization. In our models, the shared full resolution space is converted into patched tokens by a sequence of strided convolutions separated by pointwise nonlinearities as in Touvron et al. (2022).

The predictions are reconstructed from the output tokens by reversing this process. The tokens are decoded by a sequence of transposed convolution blocks and projected onto the output fields by taking coordinate-wise inner products with reconstruction vectors  $\mathbf{r}$ :

$$u(x, t + \Delta t) = \langle e(x, t + \Delta t), \mathbf{r}_u \rangle. \quad (2)$$

This can similarly be implemented as a  $1 \times 1$  convolution with the *output* channels of the convolution filter sub-selected. The mean and standard deviation computed from the inputs are then applied to these normalized outputs to produce the final de-normalized predictions as in Kim et al. (2022).

**Position Biases and Boundaries.** While in most cases, we would like the model to infer boundary conditions from the provided history, we make an exception to this policy for periodic boundaries as they change the continuity of the domain. Transformers are inherently permutation equivariant, and it is essential to include position biases so that the model can learn locality.

With a slight modification, we can use our position biases to capture the change in locality imposed by periodic boundaries. T5-style (Raffel et al., 2020) relative position encodings (RPE) utilize a lookup table to access learned embeddings corresponding to ranges of “relative distance”. For periodic boundary conditions, we modify the relative distance computation to account for neighbors across the periodic boundary. In Appendix D.1, we examine simple systems that differ only in boundary conditions and find that this minor change improves generalization in the case where we must learn both periodic and non-periodic conditions.

### 4.3 Balancing Objectives During Training

**Task Sampling.** Our pretraining procedure operates on multiple levels of sampling. The task distribution varies in system  $S$ , spatial resolution  $N_S$ , and time resolution  $T_S$  and we want diverse batches that accurately capture the signal this provides. However, sampling a full batch from multiple systems at different resolutions simultaneously would be inefficient on modern hardware as it would require batch processing of differently shaped tensors. Multi-GPU training adds an additional complication as the variance in execution time due to unbalanced workloads can lead to inefficient hardware usage.

We mitigate both of these concerns with a simple randomization scheme involving gradient accumulation. Gradient accumulation utilizes multiple backward passes per synchronization step. We therefore sample a single system  $S$  uniformly from  $\mathcal{S}$  for each *micro-batch*. With  $m$  micro-batches per synchronization step, we reduce the work-per-GPU variance  $\sigma_B^2$  to  $\frac{1}{m}\sigma_B^2$ , significantly reducing the average lost cycles due to work discrepancies. This could likely be further reduced by an approximate packing problem solution (Cormen et al., 2022), but we found the random approach was sufficient for our needs. As we employ gradient accumulation in order to increase our batch sizes, this sampling procedure incurs no additional cost.

**Scaled Training Objective.** The simplest approach to obtaining updates from the different tasks is to add their gradients. However, as the magnitudes of the state variables can vary significantly between systems, unweighted losses will result in the gradients from the problems with the largest scales drowning out losses on smaller scales (Yu et al., 2020). To partially control this behavior, we train using the normalized MSE (NMSE) defined as:

$$\mathcal{L}_{\text{NMSE}} = \frac{1}{|\mathcal{B}|} \sum_{S \in \mathcal{S}} \frac{\|\mathcal{M}(\mathbf{U}_t^S) - \mathbf{u}_{t+1}^S\|_2^2}{\|\mathbf{u}_{t+1}^S\|_2^2 + \epsilon} \quad (3)$$

where  $\mathcal{B} \subset \mathcal{S}$  denotes the micro-batch and  $\epsilon$  is a small number added for numerical stability. This does not account for the full variation in difficulty. Even if sub-task losses have similar magnitudes at the start of training, it is possible for some systems to converge quickly while other losses remain high. Nonetheless, we found that this allows our training process to produce strong results on multiple systems simultaneously.

## 5 Experiments

We design our experiments to probe three vital questions about the utility of MPP:

1. Can large transformer models learn the dynamics of multiple physical systems simultaneously?

Table 1: NRMSE comparison between MPP-pretrained models and dedicated baselines on shallow water equations (SWE), 2D Diffusion-Reaction (DiffRe2D), and compressible Navier-Stokes (CNS) at Mach numbers  $M = .1$  and  $M = 1$ . Complex parameters counted as two real. Top performing within size range and overall are bolded. Dashes indicate precision not provided by source.

MODEL	#PARAM	SWE	DIFFRE2D	CNS M1.0	CNS M0.1
MPP-AViT-T1	7.6M	0.0066	<b>0.0168</b>	<b>0.0442</b>	<b>0.0312</b>
UNET	7.7M	0.083-	0.84-	0.4725	1.6650
FNO	927K	<b>0.0044</b>	0.12-	0.1685	0.2425
FNO-B	115M	0.00246	0.0599	0.1451	0.1978
ORCA-SWIN-B	88M	0.00600	0.82-	—	—
AViT-B					
TASK-SPECIFIC	116M	0.00047	0.0110	0.0316	0.0261
MPP	116M	0.00240	0.0106	0.0281	0.0172
MPP + FINETUNED	116M	<b>0.00043</b>	<b>0.0087</b>	<b>0.0187</b>	<b>0.0079</b>
MPP-AViT-S	29M	0.0039	0.0112	0.0319	0.0213
MPP-AViT-L	409M	0.0022	0.0098	0.0208	0.0147

2. Does pretraining lead to improved accuracy on previously unseen physics?
3. Does MPP provide a finetuning advantage over existing spatiotemporal foundation models?

**Data.** We use the full collection of two-dimensional time-dependent simulations from PDEBench (Takamoto et al., 2022) as our primary source for diverse pretraining data. This includes systems governed by four unique nonlinear PDEs defined over a variety of state variables, resolutions, initial conditions, boundary conditions, and simulation parameters. The specific PDEs are the compressible and incompressible Navier-Stokes equations (CNS/INS), the shallow-water equations (SWE), and a 2D Diffusion-Reaction equation (DiffRe2D). Full details on the data used can be found in Appendix B.1.

**Training settings.**  $T_S$  is fixed at 16 for all experiments as our VideoMAE comparison in Section 5.2 was unable to scale to larger sizes without gradient checkpointing. Autoregressive training is performed only one step ahead—no longer rollouts, noise corruption, or post-processing are included for stability. Training from scratch and MPP pretraining are always performed on the AViT architecture described in section 4.2. Full training details including data splits, optimization details, and hardware are documented in Appendix C.

## 5.1 Pretraining Representations

First, we evaluate whether pretraining on multiple task actually leads to effective representations by comparing MPP-pretrained models to dedicated baselines from prior work across all available systems. The models are pretrained at a variety of sizes so we can begin to explore the benefits of scaling our approach. Precise model sizes can be found in Appendix C.1. Unlike the baselines which are trained on only one system and so must only learn one parameter regime, our models (denoted by MPP-AViT-\*) must handle all systems and regimes without finetuning. The effect of physical parameters, forcing, and simulation parameters must be inferred from context  $U_t^S$ . The UNet (Ronneberger et al., 2015) and FNO (Li et al., 2020) results are sourced from Takamoto et al. (2022) while the results from Shen et al. (2023) with a finetuned SWIN (Liu et al., 2021) are used for ORCA. As the lightweight FNO proved to be the most competitive comparison, we train an additional FNO beyond the PDEBench results that has been scaled to 115M parameters (labeled “FNO-B”) for fairness. Results are reported in terms of Normalized RMSE (NRMSE, the square root of Equation 3) averaged over fields and examples, as in Takamoto et al. (2023). Our Compressible Navier-Stokes results are aggregated based on the mach number here due to space limitations. Fully granular results can be found in Appendix D.4. Ablation results demonstrating the importance of balanced losses and normalization are shown in Appendix D.2.

Our pretrained models are able to achieve high-end performance on all datasets (Table 1) despite the difficulty of multi-task training (Yu et al., 2020) while showing improved performance with scale. Our smallest pretrained model, the MPP-AViT-Ti outperforms the PDEBench baselines on all problems except for SWE. However, though both models improve in absolute performance with scale, the

pretrained AViT-B catches up to the FNO-B. It is important to clarify that we are not claiming the pretrained models are optimal—with a series of comparisons on the AViT-B models, we show that at times, the multi-task training does hurt performance on individual tasks and that we can improve upon the pretrained model performance by finetuning our own models on specific tasks. What this experiment answers affirmatively is that large transformers can learn multiple sets of dynamics simultaneously. Trajectories from pretrained models are displayed in Appendix D.6.

## 5.2 Transfer to Low-data Domains

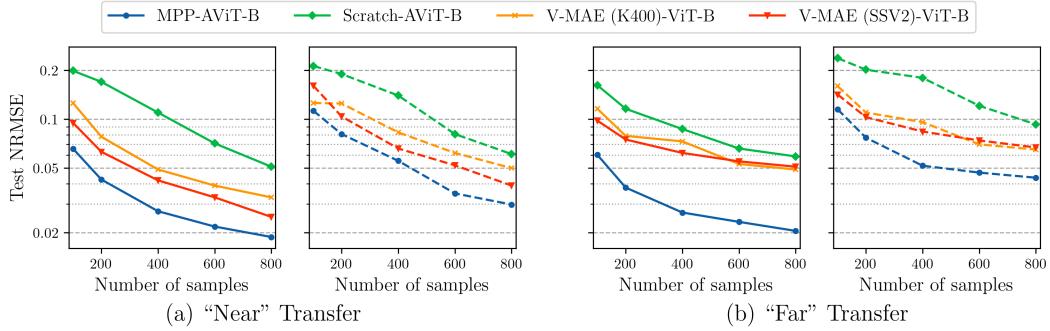


Figure 3: NRMSE for transfer learning tasks. Solid lines are one-step error. Dashed lines are averaged error over five step rollouts. The MPP model shows clear performance benefits in both cases. The more turbulent behavior of “far” seems to be difficult to learn from scratch or from video data, but pretraining on physical data leads to much stronger results.

Harkening back to Section 4.1, since we have now shown that we can learn multiple sets of dynamics with a single model, we return to the question of whether these multiple physics models are well suited to transfer learning. For a more realistic exploration of transfer, we construct a setting where the model must learn new physical behavior by removing all compressible fluid data from the pretraining corpus and pre-training on the three remaining spatiotemporal systems. We then evaluate transfer to two specific compressible Navier-Stokes datasets:

- “Near”:  $M = 0.1$ , viscosity=  $10^{-2}$ , Random Periodic Initial Conditions
- “Far”:  $M = 1.0$ , viscosity=  $10^{-8}$ , Turbulent Initial Conditions

Snapshots of the kinetic energy for the finetuning systems and incompressible pretraining data are shown in Figure 4. While quantitatively evaluating the physics gap is an unsolved problem, the names reflect both prior physical knowledge and qualitative evaluation. “Near” features a low Mach number, the dimensionless quantity that correlates with compressible behavior, and viscosity similar to that of the incompressible simulation. “Far” has wildly different turbulent behavior that induces small scale structure never seen during training. However, despite the similarity in physical behavior, the simulations are still quite different: the compressible and incompressible simulations in PDEBench differ in spatial and temporal resolution, initial condition distribution, boundary conditions, viscosity, and velocity range in addition to the difference in compressibility. We use these sets to compare the finetuning performance of MPP, training from scratch, and an existing pretrained spatiotemporal transformer, VideoMAE (Tong et al., 2022) pretrained on both K400 (Kay et al., 2017) and SSV2 (Goyal et al., 2017) datasets. Details on the finetuning procedure followed can be found in Appendix C.3.1.

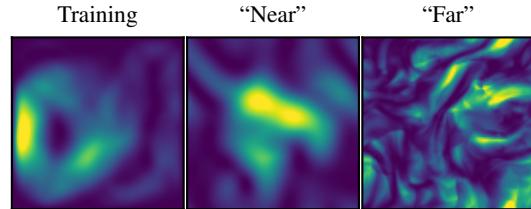


Figure 4: Kinetic energy for incompressible pre-training and compressible finetuning examples. The “near” compressible snapshot resembles the pretraining snapshot while “far” displays new turbulent small scales.

Figure 3 shows that the MPP models outperform VideoMAE and training from scratch by a large margin in the low-data regime. Numerical results are listed in Appendix C. VideoMAE displays surprisingly strong finetuning performance given that the pretraining data is conventional video, but it is unable to match the much lower memory (VideoMAE at 79.3 GB vs. AViT-B at 24.7 GB peak VRAM for batch size 1) MPP-AViT-B in either setting. Predictably, both pretraining approaches are less accurate in the long-run on the turbulent “far” dataset. However, in the short-term the physical pretraining seems to provide an even larger advantage in this regime compared to the far smoother “near” data. Rollout visualizations are included in Appendix D.7.

One possible explanation for the strong performance of finetuning for “far” is that this experiment can be viewed as a more realistic example of the compositionality exploration in Section 4.1 from the perspective of classification of second-order PDEs. The solution to Navier-Stokes in the vanishing viscosity limit represents one possible weak solution to the Euler equations which are classically hyperbolic (LeVeque & Leveque, 1992). While the viscous incompressible flow from pretraining is governed by the same transport equations as “far”, those solutions are dominated by smoothing locally parabolic behavior. However, the inviscid shallow water simulations in the training set are hyperbolic. The model has therefore seen two major components of “far”, but has never seen them within one system until finetuning.

### 5.3 Inflation to 3D

While 2D problems offer a compelling middle ground between complexity and cost for experimentation, most physical phenomena of real-world interest are fundamentally three-dimensional. We therefore examine the usefulness of our pretrained models when “inflated” to 3D. Inflation techniques were first demonstrated in Carreira & Zisserman (2017) and have seen use for extending 2D visual (image) classifiers to 3D spatiotemporal (video) settings (Nguyen et al., 2022; Xie et al., 2018). Here we employ the technique to add an additional spatial dimension.

The factored architecture of the AViT is well-suited to inflation. We initialize the projection weights discussed in Section 4.2 to those from the 2D compressible Navier-Stokes data seen during training with the new velocity direction initialized as the average of the previous two velocity projections. Since the transformer backbone acts on each spatial axis independently, the only dimension-dependent operations are the learned downsampling or “patching” convolutions. These convolutional layers are modified by following the inflation procedure of Carreira & Zisserman (2017): a 2D kernel of size  $P \times P$  is inflated into a 3D kernel by repeating the 2D kernel  $P$  times along the new axis and rescaling by  $\frac{1}{P}$ . This gives us a 3D convolutional operation that is equivalent to applying previous 2D filters then average pooling in the new direction. In practice, we find performance can be slightly improved by adding low magnitude Gaussian noise to the resulting filter. Due to the low resolution of the “Turbulent” dataset, we additionally reduced the stride of the first convolution in the hMLP by a factor of 2 for a total downsampling factor of 8 rather than the 16 used elsewhere. Standard training details are found in Appendix C.4

We compare these inflated 2D to 3D models to both training an identical architecture from scratch and PDEBench baselines. Unlike in Section 5.2 where we held out the 2D Compressible Navier-Stokes data, we use the full pretrained models from section 5.1 here. Nonetheless, this remains a significant physical gap as 2D and 3D turbulence are well-understood to have major differences in behavior (Ouellette, 2012). Due to the difficulty of scaling 3D training, results in this section are reported at the “Ti” scale.

Table 2 demonstrates significant improvements on both the turbulently and randomly initialized Compressible Navier-Stokes datasets from PDEBench with a 11.7% improvement for the smaller dataset of randomly initialized simulations and even a 4.1% improvement for the larger turbulently initialized dataset where both the dimensionality and sampling are adjusted. Training on high resolution 3D data is an enormously expensive procedure. Our results suggest that pretraining on 2D data and inflating to 3D is a promising strategy for developing models that can be used in this space.

Table 2: NRMSE for 2D to 3D inflation. Sub-headings are initial condition type.

SIZE (B, T, N)	TURBULENT	RANDOM
	(600, 21, 64 <sup>3</sup> )	(100, 21, 128 <sup>3</sup> )
FNO	0.240	0.370
UNET	0.230	1.000
AViT-SCRATCH	0.098	0.299
AViT-MPP	<b>0.094</b>	<b>0.264</b>

## 6 Conclusion

We introduced an autoregressive pretraining strategy, Multiple Physics Pretraining, for the development of multi-use physical surrogates. Through per-sample normalization, field embeddings, appropriately scaled losses, and efficient task sampling, we are able to train scalable transformer models capable of predicting multiple sets of independent dynamics simultaneously. We evaluated several sizes of model and observed that the approach benefits from scale. MPP models were able to match dedicated modern baselines on benchmarks containing fluid and reaction simulations derived from multiple equations, simulation parameters, and boundary conditions from pretraining alone with even stronger performance after undergoing task-specific finetuning. Our pretrained models also showed positive transfer by outperforming both training from scratch and existing video models on previously unseen physics. Furthermore, through kernel inflation approaches, we were able to demonstrate improved results on 3D simulation compared to training from scratch.

**Limitations and Future Work.** The focus of our work is on transfer and learning from multiple data sources, however, many interesting questions remain before we can develop true foundation models for spatiotemporal physics. For example, our choice of architecture enabled us to scale to some of the higher resolution benchmarks available today, but it also limits our models to uniform grids. Extensions to the non-uniform geometries frequently encountered in physical simulation will require further work balancing efficiency and flexibility. Extending to 3D, for example, is an enormously expensive task for many families of models. Our inflation approach is promising and future work can help identify better approaches for performing this inflation. In terms of more physically motivated concerns, we must also explore the role of constraints and conservation laws in models developed for multiple physical systems. Hard constraints that are necessary for a closed system may ensure error in an open system, thus adaptable approaches may need to be developed.

Similarly, many physically interesting tasks require handling incomplete or noisy data. While certain datasets in PDEBench do not provide all relevant physical fields (the Shallow Water equations, for instance, only contain  $h$  rather than full velocity fields needed to simulate the data) all the data that is available is observed at every grid point without noise. It remains to be seen if these approaches generalize to settings without this type of clean data.

Finally, as these are early days in the development of foundation models for physics, the limits of transfer in these spaces are poorly understood. PDEBench, which we used here, is constructed from largely fluid-like data. While we demonstrated strong transfer benefits from pretraining, it remains to be seen how far away from the training distribution or training tasks these benefits persist. It is even an open question how to define distance in this space. Future work will need to expand data diversity to push these questions and others forward. MPP opens up many new research directions and paves the way for this development in the future.

## Acknowledgments

The computations in this work were, in part, run at facilities supported by the Scientific Computing Core at the Flatiron Institute, a division of the Simons Foundation. M.P. is supported by the Department of Energy, Office of Science under contract number DE-AC02-05CH11231. M.M. would like to thank Jed Brown for his valuable insight on the simulation data. Polymathic AI acknowledges support provided by the Simons Foundation and Schmidt Sciences, LLC.

## References

Arnab, A., Dehghani, M., Heigold, G., Sun, C., Lučić, M., and Schmid, C. Vivit: A video vision transformer, 2021.

Balestrieri, R., Ibrahim, M., Sobal, V., Morcos, A., Shekhar, S., Goldstein, T., Bordes, F., Bardes, A., Mialon, G., Tian, Y., Schwarzschild, A., Wilson, A. G., Geiping, J., Garrido, Q., Fernandez, P., Bar, A., Pirsavash, H., LeCun, Y., and Goldblum, M. A cookbook of self-supervised learning, 2023.

Bar, L. and Sochen, N. Unsupervised deep learning algorithm for pde-based forward and inverse problems. *arXiv preprint arXiv:1904.05417*, 2019.

Ben-Bouallegue, Z., Clare, M. C. A., Magnusson, L., Gascon, E., Maier-Gerber, M., Janousek, M., Rodwell, M., Pinault, F., Dramsch, J. S., Lang, S. T. K., Raoult, B., Rabier, F., Chevallier, M., Sandu, I., Dueben, P., Chantry, M., and Pappenberger, F. The rise of data-driven weather forecasting, 2023.

Bertasius, G., Wang, H., and Torresani, L. Is space-time attention all you need for video understanding? In *Proceedings of the International Conference on Machine Learning (ICML)*, July 2021.

Bi, K., Xie, L., Zhang, H., Chen, X., Gu, X., and Tian, Q. Accurate medium-range global weather forecasting with 3d neural networks. *Nature*, 619(7970):533–538, 2023.

Biewald, L. Experiment tracking with weights and biases, 2020. URL <https://www.wandb.com/>. Software available from wandb.com.

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Bran, A. M., Cox, S., White, A. D., and Schwaller, P. Chemcrow: Augmenting large-language models with chemistry tools, 2023.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Bruna, J., Peherstorfer, B., and Vanden-Eijnden, E. Neural galerkin scheme with active learning for high-dimensional evolution equations, 2022.

Cao, S. Choose a transformer: Fourier or galerkin, 2021.

Carreira, J. and Zisserman, A. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations, 2020.

Chithrananda, S., Grand, G., and Ramsundar, B. Chemberta: Large-scale self-supervised pretraining for molecular property prediction, 2020.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. *Introduction to algorithms*. MIT press, 2022.

Cranmer, M., Tamayo, D., Rein, H., Battaglia, P., Hadden, S., Armitage, P. J., Ho, S., and Spergel, D. N. A bayesian neural network predicts the dissolution of compact planetary systems. *Proceedings of the National Academy of Sciences*, 118(40):e2026053118, 2021. doi: 10.1073/pnas.2026053118. URL <https://www.pnas.org/doi/abs/10.1073/pnas.2026053118>.

Dang, Y., Hu, Z., Cranmer, M., Eickenberg, M., and Ho, S. Tnt: Vision transformer for turbulence simulations, 2022.

Defazio, A. and Mishchenko, K. Learning-rate-free learning by d-adaptation, 2023.

Dehghani, M., Djolonga, J., Mustafa, B., Padlewski, P., Heek, J., Gilmer, J., Steiner, A., Caron, M., Geirhos, R., Alabdulmohsin, I., Jenatton, R., Beyer, L., Tschannen, M., Arnab, A., Wang, X., Riquelme, C., Minderer, M., Puigcerver, J., Evci, U., Kumar, M., van Steenkiste, S., Elsayed, G. F., Mahendran, A., Yu, F., Oliver, A., Huot, F., Bastings, J., Collier, M. P., Gritsenko, A., Birodkar, V., Vasconcelos, C., Tay, Y., Mensink, T., Kolesnikov, A., Pavetić, F., Tran, D., Kipf, T., Lučić, M., Zhai, X., Keysers, D., Harmsen, J., and Houlsby, N. Scaling vision transformers to 22 billion parameters, 2023.

Desai, S., Mattheakis, M., Joy, H., Protopapas, P., and Roberts, S. One-shot transfer learning of physics-informed neural networks, 2022.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Dong, X., Bao, J., Chen, D., Zhang, W., Yu, N., Yuan, L., Chen, D., and Guo, B. Cswin transformer: A general vision transformer backbone with cross-shaped windows, 2022.

Dresdner, G., Kochkov, D., Norgaard, P. C., Zepeda-Nunez, L., Smith, J., Brenner, M., and Hoyer, S. Learning to correct spectral methods for simulating turbulent flows. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL <https://openreview.net/forum?id=wNBARGxoJn>.

Duraisamy, K., Iaccarino, G., and Xiao, H. Turbulence modeling in the age of data. *Annual Review of Fluid Mechanics*, 51(1):357–377, jan 2019. doi: 10.1146/annurev-fluid-010518-040547. URL <https://doi.org/10.1146%2Fannurev-fluid-010518-040547>.

Farlow, S. J. *Partial differential equations for scientists and engineers*. Courier Corporation, 1993.

Goswami, S., Kontolati, K., Shields, M. D., and Karniadakis, G. E. Deep transfer operator learning for partial differential equations under conditional shift. *Nature Machine Intelligence*, 4(12): 1155–1164, 2022.

Goyal, R., Kahou, S. E., Michalski, V., Materzyńska, J., Westphal, S., Kim, H., Haenel, V., Fruend, I., Yianilos, P., Mueller-Freitag, M., Hoppe, F., Thurau, C., Bax, I., and Memisevic, R. The “something something” video database for learning and evaluating visual common sense, 2017.

Grossmann, T. G., Komorowska, U. J., Latz, J., and Schönlieb, C.-B. Can physics-informed neural networks beat the finite element method?, 2023.

Gupta, J. K. and Brandstetter, J. Towards multi-spatiotemporal-scale generalized pde modeling. *arXiv preprint arXiv:2209.15616*, 2022.

Han, J., Jentzen, A., and E, W. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. Masked autoencoders are scalable vision learners, 2021.

He, S., Li, Y., Feng, Y., Ho, S., Ravanbakhsh, S., Chen, W., and Póczos, B. Learning to predict the cosmological structure formation. *Proceedings of the National Academy of Sciences*, 116(28): 13825–13832, 2019. doi: 10.1073/pnas.1821458116. URL <https://www.pnas.org/doi/abs/10.1073/pnas.1821458116>.

Hendrycks, D. and Gimpel, K. Gaussian error linear units (gelus), 2016.

Hernandez, D., Kaplan, J., Henighan, T., and McCandlish, S. Scaling laws for transfer. *arXiv preprint arXiv:2102.01293*, 2021.

Ho, J., Kalchbrenner, N., Weissenborn, D., and Salimans, T. Axial attention in multidimensional transformers, 2019.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Huang, Z., Wang, X., Huang, L., Huang, C., Wei, Y., and Liu, W. Ccnet: Criss-cross attention for semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 603–612, 2019.

Jamieson, D., Li, Y., de Oliveira, R. A., Villaescusa-Navarro, F., Ho, S., and Spergel, D. N. Field-level neural network emulator for cosmological n-body simulations. *The Astrophysical Journal*, 952(2): 145, 2023.

Jiang, L. Y., Liu, X. C., Nejatian, N. P., Nasir-Moin, M., Wang, D., Abidin, A., Eaton, K., Riina, H. A., Laufer, I., Punjabi, P., et al. Health system-scale language models are all-purpose prediction engines. *Nature*, pp. 1–6, 2023.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, P., Suleyman, M., and Zisserman, A. The kinetics human action video dataset, 2017.

Kim, T., Kim, J., Tae, Y., Park, C., Choi, J.-H., and Choo, J. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International Conference on Learning Representations*, 2022. URL <https://openreview.net/forum?id=cGDAkQo1C0p>.

Klaasen, G. A. and Troy, W. C. Stationary wave solutions of a system of reaction-diffusion equations derived from the fitzhugh–nagumo equations. *SIAM Journal on Applied Mathematics*, 44(1): 96–110, 1984.

Kochkov, D., Smith, J. A., Alieva, A., Wang, Q., Brenner, M. P., and Hoyer, S. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21):e2101784118, 2021. doi: 10.1073/pnas.2101784118. URL <https://www.pnas.org/doi/abs/10.1073/pnas.2101784118>.

Kovachki, N., Li, Z., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., and Anandkumar, A. Neural operator: Learning maps between function spaces, 2023.

Lam, R., Sanchez-Gonzalez, A., Willson, M., Wirnsberger, P., Fortunato, M., Alet, F., Ravuri, S., Ewalds, T., Eaton-Rosen, Z., Hu, W., Merose, A., Hoyer, S., Holland, G., Vinyals, O., Stott, J., Pritzel, A., Mohamed, S., and Battaglia, P. Learning skillful medium-range global weather forecasting. *Science*, 382(6677):1416–1421, 2023. doi: 10.1126/science.adi2336. URL <https://www.science.org/doi/abs/10.1126/science.adi2336>.

Leung, H. W. and Bovy, J. Towards an astronomical foundation model for stars with a transformer-based model, 2023.

LeVeque, R. J. and Leveque, R. J. *Numerical methods for conservation laws*, volume 214. Springer, 1992.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

Li, Z., Zheng, H., Kovachki, N., Jin, D., Chen, H., Liu, B., Azizzadenesheli, K., and Anandkumar, A. Physics-informed neural operator for learning partial differential equations. *arXiv preprint arXiv:2111.03794*, 2021.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=Bkg6RiCqY7>.

Lu, L., Jin, P., and Karniadakis, G. E. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.

Lusch, B., Kutz, J. N., and Brunton, S. L. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature Communications*, 9(1):4950, 2018. doi: 10.1038/s41467-018-07210-0. URL <https://doi.org/10.1038/s41467-018-07210-0>.

Mialon, G., Garrido, Q., Lawrence, H., Rehman, D., LeCun, Y., and Kiani, B. T. Self-supervised learning with lie symmetries for partial differential equations, 2023.

Nguyen, E., Goel, K., Gu, A., Downs, G. W., Shah, P., Dao, T., Baccus, S. A., and Ré, C. S4nd: Modeling images and videos as multidimensional signals using state spaces, 2022.

Nguyen, T., Brandstetter, J., Kapoor, A., Gupta, J. K., and Grover, A. Climax: A foundation model for weather and climate. *arXiv preprint arXiv:2301.10343*, 2023a.

Nguyen, T. D., Ting, Y.-S., Ciucă, I., O'Neill, C., Sun, Z.-C., Jabłońska, M., Kruk, S., Perkowski, E., Miller, J., Li, J., Peek, J., Iyer, K., Różański, T., Khetarpal, P., Zaman, S., Brodrick, D., Méndez, S. J. R., Bui, T., Goodman, A., Accomazzi, A., Naiman, J., Cranney, J., Schawinski, K., and UniverseTBD. Astrollama: Towards specialized foundation models in astronomy, 2023b.

Ouellette, N. T. Turbulence in two dimensions. *Physics Today*, 65(5):68–69, 05 2012. ISSN 0031-9228. doi: 10.1063/PT.3.1570. URL <https://doi.org/10.1063/PT.3.1570>.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL [https://proceedings.neurips.cc/paper\\_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf).

Pathak, J., Subramanian, S., Harrington, P., Raja, S., Chattopadhyay, A., Mardani, M., Kurth, T., Hall, D., Li, Z., Azizzadenesheli, K., Hassanzadeh, P., Kashinath, K., and Anandkumar, A. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022.

Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., and Battaglia, P. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021. URL [https://openreview.net/forum?id=r0NqYL0\\_XP](https://openreview.net/forum?id=r0NqYL0_XP).

Rackauckas, C., Ma, Y., Martensen, J., Warner, C., Zubov, K., Supekar, R., Skinner, D., Ramadhan, A., and Edelman, A. Universal differential equations for scientific machine learning, 2021.

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. Improving language understanding by generative pre-training. 2018.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer, 2020.

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

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5–9, 2015, Proceedings, Part III 18*, pp. 234–241. Springer, 2015.

Shen, J., Li, L., Dery, L. M., Staten, C., Khodak, M., Neubig, G., and Talwalkar, A. Cross-modal fine-tuning: Align then refine, 2023. URL <https://arxiv.org/abs/2302.05738>.

Sirignano, J. and MacArt, J. F. Deep learning closure models for large-eddy simulation of flows around bluff bodies. *Journal of Fluid Mechanics*, 966, jul 2023. doi: 10.1017/jfm.2023.446. URL <https://doi.org/10.1017%2Fjfm.2023.446>.

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

Stachenfeld, K., Fielding, D. B., Kochkov, D., Cranmer, M., Pfaff, T., Godwin, J., Cui, C., Ho, S., Battaglia, P., and Sanchez-Gonzalez, A. Learned coarse models for efficient turbulence simulation, 2022.

Subel, A., Guan, Y., Chattopadhyay, A., and Hassanzadeh, P. Explaining the physics of transfer learning in data-driven turbulence modeling. *PNAS nexus*, 2(3):pgad015, 2023.

Subramanian, S., Harrington, P., Keutzer, K., Bhimji, W., Morozov, D., Mahoney, M., and Gholami, A. Towards foundation models for scientific machine learning: Characterizing scaling and transfer behavior, 2023.

Takamoto, M., Praditia, T., Leiteritz, R., MacKinlay, D., Alesiani, F., Pflüger, D., and Niepert, M. PDEBench: An Extensive Benchmark for Scientific Machine Learning. In *36th Conference on Neural Information Processing Systems (NeurIPS 2022) Track on Datasets and Benchmarks*, 2022. URL <https://arxiv.org/abs/2210.07182>.

Takamoto, M., Alesiani, F., and Niepert, M. Learning neural pde solvers with parameter-guided channel attention, 2023.

Tong, Z., Song, Y., Wang, J., and Wang, L. VideoMAE: Masked autoencoders are data-efficient learners for self-supervised video pre-training. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL <https://openreview.net/forum?id=AhccnBXSne>.

Touvron, H., Cord, M., El-Nouby, A., Verbeek, J., and Jegou, H. Three things everyone should know about vision transformers. *arXiv preprint arXiv:2203.09795*, 2022.

Tu, T., Azizi, S., Driess, D., Schaeckermann, M., Amin, M., Chang, P.-C., Carroll, A., Lau, C., Tanno, R., Ktena, I., Mustafa, B., Chowdhery, A., Liu, Y., Kornblith, S., Fleet, D., Mansfield, P., Prakash, S., Wong, R., Virmani, S., Semturs, C., Mahdavi, S. S., Green, B., Dominowska, E., y Arcas, B. A., Barral, J., Webster, D., Corrado, G. S., Matias, Y., Singhal, K., Florence, P., Karthikesalingam, A., and Natarajan, V. Towards generalist biomedical ai, 2023.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. Instance normalization: The missing ingredient for fast stylization, 2017.

Um, K., Brand, R., Yun, Fei, Holl, P., and Thuerey, N. Solver-in-the-loop: Learning from differentiable physics to interact with iterative pde-solvers, 2021.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

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

Wang, T., Roberts, A., Hesslow, D., Scao, T. L., Chung, H. W., Beltagy, I., Launay, J., and Raffel, C. What language model architecture and pretraining objective work best for zero-shot generalization?, 2022b.

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

Xie, S., Sun, C., Huang, J., Tu, Z., and Murphy, K. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification, 2018.

Xie, X., Zhou, P., Li, H., Lin, Z., and Yan, S. Adan: Adaptive nesterov momentum algorithm for faster optimizing deep models, 2023.

Xiong, R., Yang, Y., He, D., Zheng, K., Zheng, S., Xing, C., Zhang, H., Lan, Y., Wang, L., and Liu, T.-Y. On layer normalization in the transformer architecture, 2020.

Xu, W., Lu, Y., and Wang, L. Transfer learning enhanced deeponet for long-time prediction of evolution equations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 10629–10636, 2023.

Yang, L., Liu, S., Meng, T., and Osher, S. J. In-context operator learning for differential equation problems. *arXiv preprint arXiv:2304.07993*, 2023.

Yu, B. et al. The deep ritz method: a deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1):1–12, 2018.

Yu, T., Kumar, S., Gupta, A., Levine, S., Hausman, K., and Finn, C. Gradient surgery for multi-task learning, 2020.

Zang, Y., Bao, G., Ye, X., and Zhou, H. Weak adversarial networks for high-dimensional partial differential equations. *Journal of Computational Physics*, 411:109409, 2020.

Zhai, X., Kolesnikov, A., Houlsby, N., and Beyer, L. Scaling vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12104–12113, 2022.

## A Impact Statement

This paper contributes to the field of machine learning for computational physics. In terms of positive impact, we would hope that our approach enables researchers and engineers to develop more accurate dynamics models from limited observations. While we see mostly positive implications from this, the danger of any computational physics research is potential use in weapons research. In this case, we feel that risk is fairly small as research in such spaces tends to focus on well-understood physics where numerical methods already obtain extremely high precision while our approach is more suited to the more scientifically oriented space of modeling poorly understood physics at lower precision.

## B Data Details

### B.1 PDEBench

To train and evaluate our models, we use the publicly available PDEBench dataset<sup>2</sup> (Takamoto et al., 2022). We summarize the data included in this section. This dataset comprises a suite of time dependent and time independent simulations based on common PDE systems, generated with varying parameters, initial conditions, and boundary conditions. Specifically, PDEBench uses a discretized ground-truth solver with high precision to evolve the vector-valued solution to a given PDE at one time step to the solution at one time step later. When compiled across time steps, the vector-valued solutions take the form  $x \in \mathbb{R}^{T \times C \times H \times W}$ , where  $T$  denotes the total number of time steps,  $H$  and  $W$  denote the spatial height and width of the simulation grid and  $C$  denotes the parameter space representing the velocity ( $v_x$  and  $v_y$ ), pressure ( $p$ ) and density ( $\rho$ ) fields, such that  $C = 4$ . For our study, we focus on the 2D fluid dynamics simulations in PDEBench. These are outlined loosely below; for more details, we refer the reader to Takamoto et al. (2022):

**Compressible Navier-Stokes:** These equations are used to model the pressure and velocity of both laminar and turbulent Newtonian fluids, and are applied to many real-world problems, from aerodynamics to interstellar gas dynamics. In the regime in which the density of the fluid can change due to pressure variation, the equations can be expressed:

$$\partial_t \rho + \nabla \cdot (\rho \mathbf{v}) = 0, \quad (4)$$

$$\rho (\partial_t \mathbf{v} + \mathbf{v} \cdot \nabla \mathbf{v}) = -\nabla p + \eta \nabla^2 \mathbf{v} + (\zeta + \eta/3) \nabla (\nabla \cdot \mathbf{v}) \quad (5)$$

$$\partial_t (\epsilon + \rho v^2/2) + \nabla \cdot [(p + \epsilon + \rho v^2/2) \mathbf{v} - \mathbf{v} \cdot \boldsymbol{\sigma}'] = 0, \quad (6)$$

where  $\rho$  is the fluid density,  $\mathbf{v}$  is the fluid velocity,  $p$  is the fluid pressure,  $\epsilon$  is the internal energy,  $\boldsymbol{\sigma}'$  is the viscous stress tensor,  $\eta$  is the shear viscosity, and  $\zeta$  is the bulk viscosity. For our transfer experiments, we use the following two sets of data in particular:

1. A set of 1,000 trajectories on a  $H \times W = 512 \times 512$  regular grid over  $T = 100$  time steps (where the separation between steps is  $\Delta t = 0.005$ ). Additionally,  $(M, \eta, \zeta) = (1.0, 10^{-8}, 10^{-8})$ , where  $M$ ,  $\eta$ ,  $\zeta$  denote the Mach number, the shear viscosity, and the bulk viscosity, respectively. The velocity field is initialized with a turbulent field, while the initial pressure and density fields are taken to be uniform.
2. A set of 10,000 trajectories on a  $H \times W = 128 \times 128$  regular grid with  $(M, \eta, \zeta) = (0.1, 0.01, 0.01)$ . The time steps and initializations are as above.

**Incompressible NS:** In the incompressible regime, which typically occurs in fluids with low Mach numbers (as it rules out density and pressure waves like sound or shock waves), the Navier-Stokes equations simplify to:

$$\nabla \cdot \mathbf{v} = 0, \quad (7)$$

$$\rho (\partial_t \mathbf{v} + \mathbf{v} \cdot \nabla \mathbf{v}) = -\nabla p + \eta \nabla^2 \mathbf{v} + \mathbf{f}, \quad (8)$$

where  $\mathbf{v}$  is the velocity,  $\rho$  is the density,  $p$  is the pressure,  $\eta$  is the viscosity, and  $\mathbf{f}$  is a spatially varying external force. The simulation in PDEBench is augmented by an immersed tracer that is transported by the velocity field:

$$\partial_t \rho_{\text{smoke}} = -\mathbf{v} \cdot \nabla \rho_{\text{smoke}} \quad (9)$$

<sup>2</sup><https://github.com/pdebench/PDEBench>

The system uses Dirichlet boundary conditions on the velocity field  $\mathbf{v} = 0$  and Neumann on the density  $\frac{\partial \rho_{smoke}}{\partial x} = 0$ ,  $x \in \{0, 1\}$ ,  $\frac{\partial \rho_{smoke}}{\partial y} = 0$ ,  $y \in \{0, 1\}$ . These equations are typically used to model a variety of hydrodynamics systems such as weather. This data is produced at resolution  $512 \times 512$  with time step of .0005. The dataset contains a total of 1000 trajectories with 1000 time steps each.

**Shallow water:** In the event that the horizontal length scale of the fluid is significantly greater than the vertical length scale, the incompressible Navier-Stokes equations can be depth-integrated to derive the shallow water equations. These describe flow below a pressure surface in a fluid, and are given by

$$\partial_t h + \nabla \cdot (h\mathbf{v}) = 0, \quad (10)$$

$$\partial_t(h\mathbf{v}) + \nabla \cdot \left( \frac{1}{2}h\mathbf{v}^2 + \frac{1}{2}g_r h^2 \right) = -g_r h \nabla b, \quad (11)$$

where  $h$  is the water depth,  $\mathbf{v}$  is the velocity,  $b$  is the bathymetry, and  $g_r$  is the reduced gravity. For our data, we use 1,000 trajectories on a  $H \times W = 128 \times 128$  regular grid over  $T = 100$  time steps. The specific simulation used is a 2D radial dam break scenario, where the water height is initialized as a circular bump in the center of the domain with a uniformly randomly sampled radius.

**Diffusion-Reaction:** The Diffusion-Reaction equations arise in systems with many interacting components and can be represented in the general form

$$\partial_t \mathbf{u} = \mathbf{D} \nabla^2 \mathbf{u} + \mathbf{R}(\mathbf{u}), \quad (12)$$

where  $\mathbf{u}$  is a vector of concentration variables,  $\mathbf{D}$  is a diagonal matrix of diffusion coefficients, and  $\mathbf{R}$  describes all local reaction kinetics. The most common application of diffusion-reaction equations is in chemical reactions, however they can also be used to describe a variety of dynamical processes. For our data, we use 1,000 trajectories on a  $H \times W = 128 \times 128$  regular grid over  $T = 100$  time steps. The reaction functions for the activator and inhibitor are defined by the Fitzhugh-Nagumo equation (Klaasen & Troy, 1984), and their diffusion coefficients are  $D_u = 1 \times 10^{-3}$  and  $D_v = 5 \times 10^{-3}$  respectively. The initial conditions are generated as standard Gaussian random noise.

## B.2 PDEArena

In addition to the 2D Incompressible Navier-Stokes data incorporated from PDEBench, we also include 2D Incompressible Navier-Stokes data from PDEArena (Gupta & Brandstetter, 2022).

These follow roughly Equation 7, with a minor variation:

$$\nabla \cdot \mathbf{v} = 0, \quad (13)$$

$$\rho (\partial_t \mathbf{v} + \mathbf{v} \cdot \nabla \mathbf{v}) = -\nabla p + \eta \nabla^2 \mathbf{v} + \begin{bmatrix} b \\ 0 \end{bmatrix}, \quad (14)$$

where  $b \in [0.2, 0.5]$  represents buoyancy in the  $y$  direction. Unlike PDEBench, this is a spatially constant term. This includes a set of 5,200 training trajectories (and 1,300 validation and test trajectories each) on a  $H \times W = 128 \times 128$  regular grid from which we take  $T = 16$  timesteps for prediction. As with the PDEBench simulations, the PDEArena simulations include a viscosity parameters of  $\nu = 0.01$  and Dirichlet boundary conditions.

## C Experiment Details

### C.1 Model Configurations

The following architectural decisions were used across all AViT models trained in this paper:

- **Pre/Post Norm:** Pre-norm (Xiong et al., 2020)
- **Normalization Type:** Instance Normalization (Ulyanov et al., 2017)
- **Activations:** GeLU (Hendrycks & Gimpel, 2016)
- **QK Norm:** Yes (Dehghani et al., 2023)
- **Patching:** hMLP (Touvron et al., 2022)

Table 3: Details of the various model architectures and scales explored.

MODEL	EMBED DIM.	MLP DIM.	# HEADS	# BLOCKS	PATCH SIZE	# PARAMS
AViT-TI	192	768	3	12	[16, 16]	7.6M
AViT-S	384	1536	6	12	[16, 16]	29M
AViT-B	768	3072	12	12	[16, 16]	116M
AViT-L	1024	4096	16	24	[16, 16]	409M

Table 4: Inference time for various models on A6000 GPU.

MODEL	TIME (MS)
UNET	67.7
FNO	7.2
AViT-TI	19.2
ORCA-SWIN-B	98.5
AViT-B	105.6

- **Decoder:** Transposed hMLP (this is equivalent to the transposed convolutions mentioned in the main text).
- **Causal Masking:** False - We only evaluate the loss on the  $T + 1$  prediction.

Furthermore, we examine the performance of our models on the aforementioned PDE systems when the size of the model is scaled. Vision transformers have a variety of parameters that control the model’s size, including the number of processor blocks, the dimensionality of patch embeddings and self-attention, the dimensionality of Multi-Layer Perceptron (MLP) blocks, the number of attention heads, and the patch size applied on the input tensors. In previous studies on language (Hernandez et al., 2021; Hoffmann et al., 2022; Kaplan et al., 2020) and vision (Zhai et al., 2022), it has generally been noted that model performance is typically only weakly dependent on shape parameters, and instead depends largely on non-embedding parameter count given a fixed compute budget and dataset size. As such, we follow the general scaled architectures set forth by Zhai et al. (2022) for vision, and scale all aspects of the model shapes simultaneously to select a variety of model sizes for testing. These are detailed in 3. In practice, the AViT models are on average slightly slower than the similarly sized baseline models from Table 1.

**Software.** All model development and training in this paper is performed using PyTorch 2.0 (Paszke et al., 2019).

**Hardware.** All training for both pretraining and finetuning is done using Distributed Data Parallel (DDP) across 8 Nvidia H100-80GB GPUs.

## C.2 Exp 1: Pretraining Performance

For MPP, we train using the following settings:

- **Training Duration:** 200K steps
- **Train/Val/Test:** .8/.1/.1 split per dataset on the trajectory level.
- **Task sampling:** Uniformly sample task, then uniformly sample trajectory from task without replacement. We treat every 400 model updates (1 model update=5 micro-batches) as an “epoch” and reset the task pool.
- **Micro-batch size:** 8
- **Accumulation Steps:** 5
- **Optimizer:** Adan (Xie et al., 2023)
- **Weight Decay:** 1E-3
- **Drop Path:** 0.1

- **Base LR:** DAdaptation (Defazio & Mishchenko, 2023)
- **LR Schedule:** Cosine decay
- **Gradient clipping:** 1.0

For training from scratch, no task sampling occurs and sampling without replacement continues until the dataset is exhausted as in a conventional epoch. Note, we use the automated learning selection strategy DAdaptation during pretraining runs in large part to avoid excessive hyperparameter tuning of our own models. In finetuning experiments, comparison models are tuned manually following the recommended settings from the model publishers to avoid differences being due to compatibility with the parameter-free method.

**FNO-B** For the scaled-up FNO, we modify the parameters used in PDEBench to 6 layers, 24 modes, and width 100. Training configurations were taken from PDEBench (500 full passes through the dataset with the Adam optimizer) with learning rate selected by DAdaptation as in the training of our models for consistency.

**Data** For pretraining, we use all 2D time-dependent PDEBench datasets. These are described in Section B.1. In particular, we use the compressible and incompressible Navier-Stokes, Diffusion-Reaction 2D, and Shallow Water data.

### C.3 Experiment 2: Transfer to Low-Data Domains

In this experiment, we compare the transferability of our MPP-Pretrained models to general-purposes pretrained video masked autoencoders (VideoMAE; Tong et al., 2022) for frame prediction on video-like PDEBench data (Takamoto et al., 2022).

**Data** We study transferability of VideoMAE models for spatiotemporal prediction on video-like scientific data.

MPP-labeled models are pretrained on datasets generated from three PDEs: Incompressible Navier-Stokes, Shallow Water, and Diffusion Reaction 2D. This is performed using the same training settings as in Section C.2.

We focus on transfer to the two datasets “Near” and “Far” (see Sect. 5.2) of fluid dynamics simulations taken from the PDEBench dataset (Takamoto et al., 2022). These simulations solve the compressible Navier-Stokes equations in a 2D geometry with periodic boundary conditions (see Appendix B.1 for additional details).

#### C.3.1 MPP Finetuning Procedure

For MPP and training from scratch, we use the following settings:

- **Training Duration:** 500 epochs (true epochs due to the restricted dataset)
- **Train/Val/Test:** X/.1/.1 split per dataset on the trajectory level. Note that X is due to the fact that we test varying amounts of training data. These are subsampled from the training split of 80%.
- **Batch size:** 8
- **Accumulation Steps:** 1 (No accumulation)
- **Optimizer:** Adan (Xie et al., 2023)
- **Weight Decay:** 1E-3
- **Drop Path:** 0.1
- **Base LR:** DAdaptation (Defazio & Mishchenko, 2023)
- **LR Schedule:** Cosine decay
- **Gradient clipping:** 1.0

**Channel expansion.** The only component of the architecture used that is aware of the particular fields being ingested is the field embedding and debedding projection layers. As the finetuning data adds previously unseen state variables, this matrix needs to be expanded. To be concrete, out of the four state variables present in the CNS data –  $v_x, v_y, P, \rho$  – the training data only contains  $v_x, v_y, \rho$ . If the original field embedding layer was therefore a  $1 \times 1$  convolution with 3 input channels, we follow the following procedure to add the new projection.

1. Instantiate a new `field_projection` and `field_reconstruction` layer with a sufficiently large number of channels to process both previously seen fields and new fields.
2. If  $W$  is a  $(..., C_{in}, C_{out})$  `field_projection` weight matrix, set  $W^{new}[:, :, C_{in}^{old}, :] = W^{old}$ . Perform the corresponding replacement within the `field_reconstruction` layer.

All other weights are loaded as normal without modification.

**Weights trained.** We found that partially frozen training resulted in noticeably worse performance and therefore finetuned the full model.

### C.3.2 VideoMAE Settings

While VideoMAE does utilize spatiotemporal information, it was developed for a different setting, so we fully document all details of our adaptation of it here both for reproducibility and fairness in our comparison.

VideoMAE models are video transformers that were proven to be efficient data-learners for self-supervised video pretraining (Tong et al., 2022). They rely on an asymmetric encoder-decoder architecture building on a vanilla ViT backbone with joint space-time attention. VideoMAE models are pretrained by learning to reconstruct masked videos using a random tube-masking strategy with a extremely high masking ratio ( $\sim 90\%$ ).

We make use of two publicly available models, hereafter called VideoMAE-K400 and VideoMAE-SSV2, that were pretrained on Kinetics-400 dataset (K400; Kay et al., 2017) and Something-Something V2 dataset (SSV2; Goyal et al., 2017), respectively. Both datasets are made of short videos (typically  $\leq 10$  s long) of human-object or human-human interactions. VideoMAE-K400 (respectively, VideoMAE-SSV2) was pretrained on  $\sim 240$ k ( $\sim 170$ k) videos. We focus on the models that build on a ViT-base backbone, so that their size (in terms of number of trainable parameters) remains comparable to that of MPP-AViT-B. After adaptation of the input and output linear layers as described below, the number of trainable parameters of these models reaches  $\sim 95$  M.

**Number of channels.** Same as the original pretraining procedure, the input data  $x \in \mathbb{R}^{C \times T \times H \times W}$  is divided into non-overlapping joint space-time cubes of size  $2 \times 16 \times 16$ . These are embedded through a Conv3d layer, resulting in  $\frac{T}{2} \times \frac{H}{16} \times \frac{W}{16}$  tokens. Since our PDEBench data has  $C = 4$  channels instead of 3 for the RGB videos from the pretraining set, we had to adapt the number of input channels of this Conv3d layer accordingly. The weights of this new layer were defined using a (rescaled) repetition of the pretrained weights from the original layer. Similarly, the output number of features of the final linear projection layer of the model had to be adapted to  $C = 4$  channels. The weights and biases of this layer were extended by consistently repeating the original pretrained weights and biases.

**Positional encoding.** The number of tokens resulting from our PDEBench data did not match the number of tokens resulting from the pretraining datasets. Consequently, we also had to adapt the pretraining positional encoding. We chose to interpolate accordingly the original 1D sine/cosine positional encoding (Vaswani et al., 2017) using a trilinear interpolation after having reshaped the token index axis onto a 3D grid.

### C.3.3 Video MAE Finetuning Procedure

We describe the finetuning procedure of the pretrained VideoMAE models for frame prediction. Frame prediction consists in predicting the next  $T_p$  frames of a video given a context of  $T_c$  frames. Since the pretrained models manipulates space-time cubes of size 2 in time, we naturally choose  $T_p = 2$ . The context size is taken to be  $T_c = 16$  for consistency with MPP-AViT models. We finetune

Table 5: Effective learning rate for the finetuning of VideoMAE.

	“NEAR”	“FAR”
VIDEOMAE (K400)	0.00039	0.00198
VIDEOMAE (SSV2)	0.00186	0.00150

the pretrained models for frame prediction by adapting the self-supervised training strategy in order to reconstruct the last  $T_p$  frames of a masked video of  $T = T_c + T_p$  frames.

**Masking strategy.** For frame prediction, instead of the random tube-masking strategy, we simply mask the last  $T_p$  frames of the input data.

**Loss.** We finetune our models by minimizing a NMSE loss. In this context, denoting by  $x, y \in \mathbb{R}^{C \times T_p \times H \times W}$  the output of our model and the target (masked frames), respectively, the NMSE loss is defined by  $\mathcal{L}(x, y) = \sum_{c=1}^C \sum_{t=1}^{T_p} \|x_{c,t} - y_{c,t}\|_2^2 / \|y_{c,t}\|_2^2$ .

**Normalization of the data.** Each set of PDEBench simulations is globally and channel-wise rescaled so that pixel values all fit in  $[0, 1]$ . Additionally, we normalize channel-wise the targets  $y \in \mathbb{R}^{C \times T_p \times H \times W}$  by subtracting the global mean of the corresponding context frames and then dividing by their global standard deviation.

**Optimization.** We finetune the pretrained models over 500 epochs (full epochs due to restricted data size) and a (total) batch size of 8 using AdamW optimizer (Loshchilov & Hutter, 2019). Except for the learning rate, the remaining optimization hyperparameters are chosen to be consistent with those used in the finetuning experiments of Tong et al. (2022) (Table 10). In particular, we choose a weight decay  $\lambda = 0.05$ ,  $(\beta_1, \beta_2) = (0.9, 0.999)$ , a cosine learning rate decay scheduler with 5 warmup epochs, a drop path rate of 0.1, and a layer-wise learning rate decay parametrized by 0.75. In this setting, the learning rate is adjusted by performing a hyperparameter search monitored with WandB (Biewald, 2020). We report the resulting optimal values per pretrained model and dataset in Table 5.

#### C.4 Experiment 3: Inflation to 3D

For MPP and training from scratch, we use the following settings:

- **Training Duration:** 100 True Epochs
- **Train/Val/Test:** .8/.1/.1
- **Batch size:** 8
- **Accumulation Steps:** 1 (No accumulation)
- **Optimizer:** Adan (Xie et al., 2023)
- **Weight Decay:** 1E-3
- **Drop Path:** 0.1
- **Base LR:** DAdaptation (Defazio & Mishchenko, 2023)
- **LR Schedule:** Cosine decay
- **Gradient clipping:** 1.0

**Data** We use both 3D Compressible Navier-Stokes datasets provided by PDEBench for these experiments. The first has 600 total trajectories of 21 steps simulated with sheer and bulk viscosities of  $10^{-8}$  with explicitly turbulent initialization. The second contains 100 total trajectories of 21 steps with viscosities of  $10^{-8}$  and random initialization.

Table 6: Validation NRMSE for position bias comparison. Compares training performance on data that differs only in boundary conditions.

TRAINING	PERIODIC	ABSORBING
PERIODIC BASELINE	0.032	—
ABSORBING BASELINE	—	0.295
COMBINED		
STANDARD RPE	0.188	0.189
PERIODIC-ADJUSTED RPE	0.081	0.143

### C.5 Appendix Experiment: Broader Usage of Pretrained Representations

For MPP and training from scratch, we use the following settings:

- **Training Duration:** 20K Optimization Steps
- **Train/Val/Test:** 1000/100/1000 taken from original validation set or randomly depending on whether data was used for training.
- **Batch size:** 24
- **Accumulation Steps:** 1 (No accumulation)
- **Optimizer:** Adan (Xie et al., 2023)
- **Weight Decay:** 1E-3
- **Drop Path:** 0.1
- **Base LR:** DAdaptation (Defazio & Mishchenko, 2023)
- **LR Schedule:** Cosine decay
- **Gradient clipping:** 1.0

## D Additional and Extended Results

### D.1 Position Bias Evaluation

We isolate the impact of position biases on our multi-task training objectives by constructing an experiment that isolates their influence. Recall the advection equation from Equation 1:

$$\frac{\partial \psi}{\partial t} + \nabla \cdot (v\psi) = 0 \quad (15)$$

We will define two sets of physics. In both cases, the function is defined on the 1D domain  $x \in [0, 1]$ . We sample  $v \sim \text{Unif}(-1, 1)$  and use initial conditions sampled from the set of circular Gaussians with variances sampled from  $\text{Unif}(1/160, 1/5)$  and means sampled from  $\text{Unif}(.25, .75)$ . The two systems vary only in the choice of boundary conditions. The first uses periodic boundary conditions, implying  $\phi(0) = \phi(1)$ . The second uses absorbing boundary conditions in which waves are not reflected back into the solution space. The restricted functional form allows us to implement this exactly by extending the domain and solving the periodic equations such that the constant velocity implies the waves exiting the solution space never return.

In this experiment, we first train models (AViT-Ti with 1D patches) on each system individually using 10,000 examples each for 100 epochs to get a sense of the baseline performance. We then train models with and without our modified position biases on the two systems jointly (20,000 examples) to evaluate the impact of our change.

Table 6 shows that our modified position biases are more effective at training in the joint setting. Both RPE schemes are able to improve on absorbing boundary with the additional data. Standard RPE on the other hand struggles to learn the periodic baseline. Our Periodic-adjusted variant is much more effective at learning the periodic data, though it does not outperform the baseline.

It is interesting to note how large the effect of boundary conditions is on this problem. The model trained on only periodic condition reaches nearly an order of magnitude higher precision. While

Table 7: Ablation table showing changes in NRMSE as our proposed modifications to the architecture and model are removed.

SETTING	SWE	DIFFRE2D	CNS M1.0	CNS M0.1
MPP-AViT-B	0.00240	0.0106	0.0281	0.0172
REMOVE NRMSE TRAINING	0.01353	0.1502	0.1245	0.1213
REMOVE REVIN	0.02651	0.0661	0.2601	0.3266
REMOVE BOTH	0.03619	0.2952	0.4049	1.5892

Table 8: RMSE comparison between MPP-pretrained models and dedicated baselines on shallow water equations (SWE), 2D Diffusion-Reaction (DiffRe2D). Complex parameters counted as two real.

MODEL	#PARAM	SWE	DIFFRE2D
MPP-AViT-T1	7.6M	6.9E-3	1.1E-3
UNET	7.7M	8.6E-2	6.1E-2
FNO	927K	4.5E-3	8.1E-3
AViT-B			
TASK-SPECIFIC	116M	4.9E-4	8.8E-4
MPP	116M	3.6E-3	6.6E-4
MPP + FINETUNED	116M	4.2E-4	6.2E-4
MPP-AViT-S	29M	4.0E-3	7.8E-4
MPP-AViT-L	409M	2.3E-3	5.0E-4

absorbing boundaries are complicated for numerical solvers, it seems as though attention should be able to simply not attend to waves passing out of the domain. The interaction of boundary conditions with attention therefore seems to be an important direction for future study.

## D.2 Normalization Ablations

We perform several ablations to record the impact of our proposed normalization and loss balancing approaches to the task of multiple physics pretraining. These results are shown in Table 7 using AViT-B as the baseline. As we can see both modifications greatly improve the ability of our base model to learn in this setting. Note that all of these experiments are performed in the multiple-physics settings. Results for dedicated models are listed in the main text.

Note that while training on a normalized loss appears to be important for MPP, the ordinal value of the final results is agnostic to the metric chosen. For example, we include Table 8, a version of Table 1 which reports loss in RMSE instead of NRMSE on the smaller datasets to demonstrate that while not all reported models are trained using the same metric, this does not significantly change the results. Note that not all sources report RMSE so this table contains fewer results compared to the main text.

## D.3 Broader Usage of Pretrained Representations

One of the fascinating aspects of large pretrained models is the utility of their learned features for entirely new types of prediction problems. We explore this behavior by comparing the ability of a pretrained MPP-AViT-B model to one trained from scratch to solve the inverse problem of parameter estimation for two parameters.

**Forcing Identification for Incompressible Navier-Stokes** The two sources of variation in the Incompressible Navier-Stokes simulations (Equation 7) are the initial conditions and the spatially varying forcing  $f$  applied to the velocity evolution at each step. We compare the performance between the pretrained the constant forcing term used in the incompressible Navier-Stokes simulation from an input trajectory  $U_t^S$ . We divide the validation set from pretraining, taking 1,000 trajectories as the new training set and using the rest for validation. Results are reported on the original test set.

**Buoyancy for Incompressible Navier-Stokes** For this, we turn to an additional fluid mechanics benchmark, PDEArena (Gupta & Brandstetter, 2022). This benchmark includes an incompressible

Table 9: RMSE for inverse problem tasks. Error from constant prediction included for context.

TRAINING	FORCING	BUOYANCY
MPP	$0.20^{\pm.008}$	$0.078^{\pm.006}$
SCRATCH	$0.43^{\pm.012}$	$0.077^{\pm.005}$
MIALON ET AL. (2023)	—	$0.062^{\pm.010}$
PREDICT MEAN	$1.00^{\pm.000}$	$0.088^{\pm.000}$

Table 10: Per dataset NRMSE comparison for  $M = 0.1$  Compressible Navier-Stokes data. R/T denote “random” and “turbulent” initial conditions from PDEBench.  $\eta = \zeta$  are the bulk and sheer viscosity.

MODEL	$R-\eta = 10^{-8}$	$R-\eta = 10^{-2}$	$R-\eta = 10^{-1}$	$T-\eta = 10^{-8}$
MPP-AViT-T1	0.0493	0.0274	0.0116	0.0339
UNET	0.66–	0.71–	5.1—	0.19–
FNO	0.28–	0.17–	0.36–	0.16–
MPP-AViT-S	0.0335	0.0176	<b>0.0071</b>	0.0217
FNO-B	0.1810	0.1129	0.3800	0.1175
MPP-AViT-B	0.0286	0.0162	0.0078	0.0169
MPP-AViT-L	<b>0.0234</b>	<b>0.0145</b>	0.0099	<b>0.0136</b>

Navier-Stokes simulation with variable buoyancy ( $b$  from Equation 13). Since this set was not used during training, we take 1,000 randomly sampled trajectories for train, 100 for validation, and a further 1,000 for testing. Since we are now predicting a scalar, we train a linear probe on top of the final hidden representation consisting of global average pooling and a linear head.

We observe mixed results (Table 9). Pretraining reduces the error in the forcing task by nearly half, but shows no improvement over training from scratch in the scalar prediction. Prior work (Mialon et al., 2023) was able to achieve better performance on buoyancy through Lie-transformation based contrastive pretraining using a convolutional architecture. MPP does not seem to hurt performance on this task, as the AViT trained from scratch also barely outperforms a mean prediction. However, we would expect the scalar prediction task to be easier. It is plausible that the dense prediction pretraining task is not well-suited for scalar inference or that the pooled representation frequently used for finetuning classification models in vision is not well suited to parameter inference, but the comparison of performance on this non-generative task also echoes prior work in NLP (Wang et al., 2022b) where autoregressive training has underperformed on non-generative tasks.

#### D.4 Exp1: Expanded Results

Here we break out the Compressible Navier-Stokes (CNS) results from Table 1. Table 1 shows the comparison between our pretrained models and task-specific baselines; however, due to space limitations the CNS was aggregated by mach number in the main text, so we share the full CNS results here. M0.1 can be seen in Table 10. M1.0 can be seen in Table 11. Note that while it is conventional to describe these simulations in terms of dimensionless numbers like the Reynolds number, these simulations are performed at relatively low resolution, so it is likely they incur significant numerical diffusion. Thus we report the results in terms of the nominal diffusion coefficients without making claims about the Reynolds numbers of the simulation.

In examining the full CNS data, one interesting result jumps out - the most viscous systems  $\eta = .1$  seem to perform relatively worse with scale. For both subsets, S was the top performing model at the highest viscosity. All other viscosities seem to benefit from scale. This does seem to have a limit, however, as Ti again loses performance. It is also important to remember that these results occur during multi-task training, so they cannot be directly interpreted in the single-task setting.

Table 11: Per dataset NRMSE comparison for  $M = 1.0$  Compressible Navier-Stokes simulations. R/T denote “random” and “turbulent” initial conditions from PDEBench.  $\eta = \zeta$  are the bulk and sheer viscosity.

MODEL	R- $\eta = 10^{-8}$	R- $\eta = 10^{-2}$	R- $\eta = 10^{-1}$	T- $\eta = 10^{-8}$
MPP-AViT-T1	0.0615	0.0327	0.0171	0.0594
UNET	0.47-	0.36-	0.92-	0.14-
FNO	0.35-	0.096-	0.098-	0.13-
MPP-AViT-S	0.0451	0.0223	<b>0.0108</b>	0.0425
FNO-B	0.3216	0.0573	0.0914	0.1100
MPP-AViT-B	0.0386	0.0195	0.0119	0.0365
MPP-AViT-L	<b>0.0314</b>	<b>0.0171</b>	0.0132	<b>0.0282</b>

Table 12: Test NRMSE for “Near” Compressible Navier-Stokes M0.1,  $\eta = .01$ .

MODEL	# TRAINING SAMPLES (NRMSE $\times 10^{-1}$ )									
	100		200		400		600		800	
	T+1	T+5	T+1	T+5	T+1	T+5	T+1	T+5	T+1	T+5
VIDEOMAE (K400)	1.26	1.98	0.78	1.25	0.49	0.83	0.39	0.62	0.33	0.50
VIDEOMAE (SSV2)	0.95	1.61	0.63	1.04	0.42	0.66	0.33	0.52	0.25	0.39
MPP-AViT-B	0.66	1.13	0.42	0.81	0.27	0.55	0.22	0.35	0.19	0.30

## D.5 Exp2: Numerical Results

We provide numerical results corresponding to Figure 3 in Tables 12 and 13. We refer to Sect. 5.2 for discussion.

## D.6 Pretraining Trajectories

Here we show example trajectories from pretrained models. Videos are included in the attached supplementary material. After pretraining, we find that the model initially produces strong predictions, but patch artifacts creep in over time.

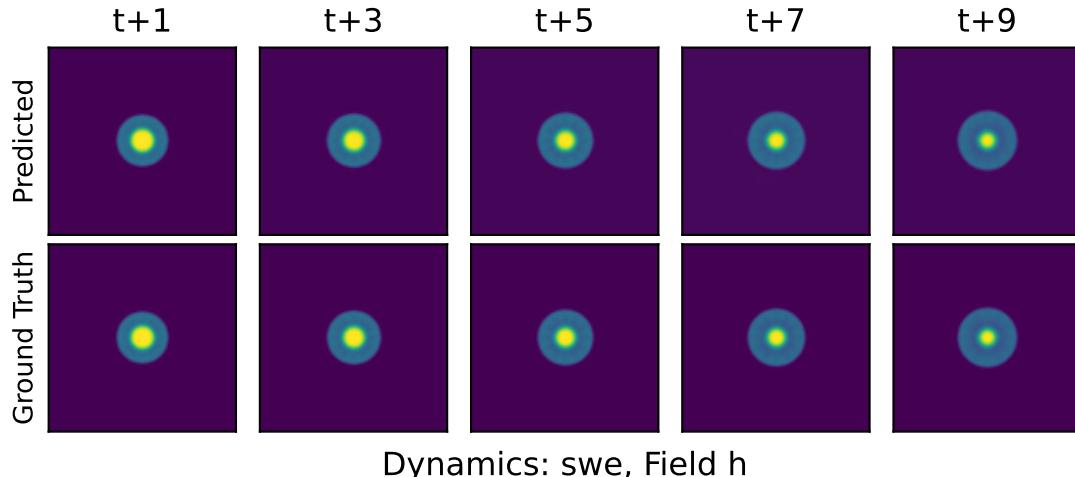
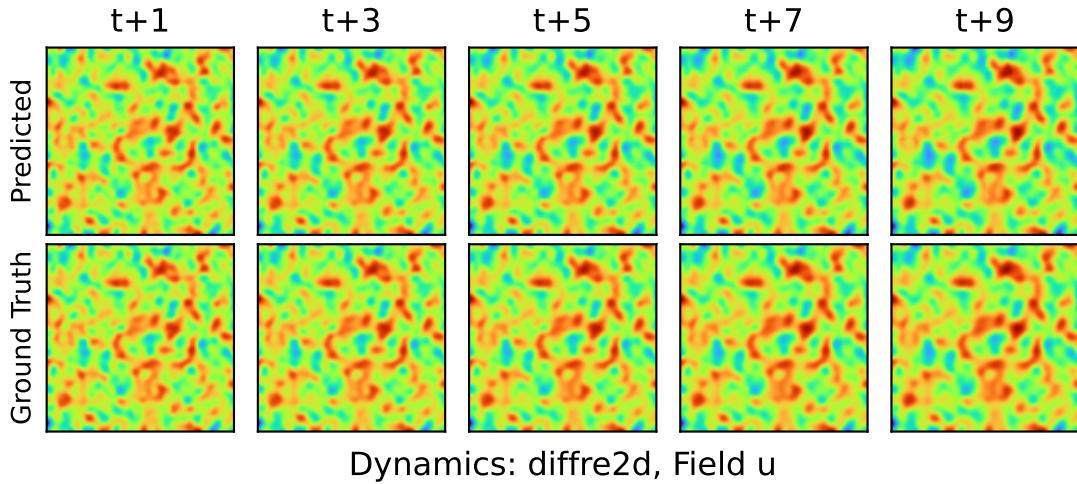


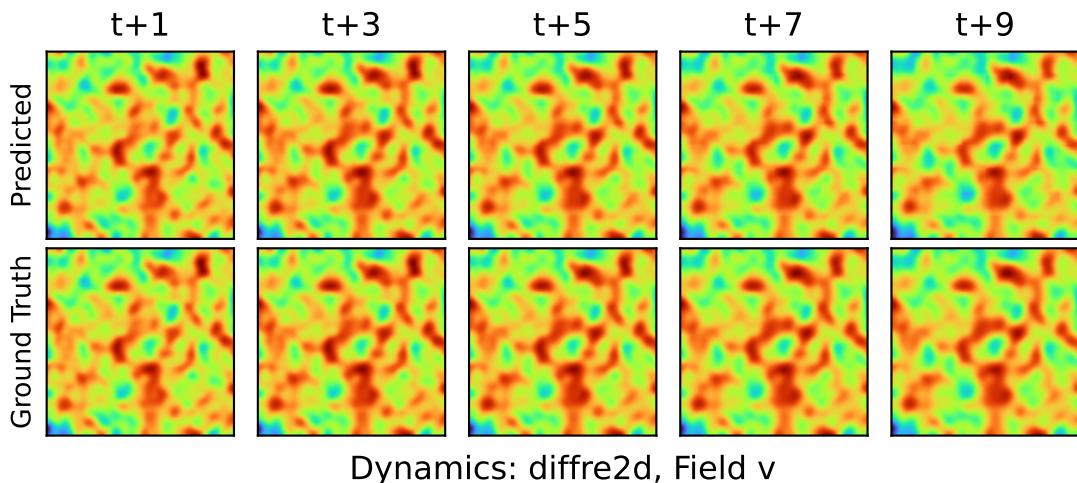
Table 13: Test NRMSE for “Far” Compressible Navier-Stokes

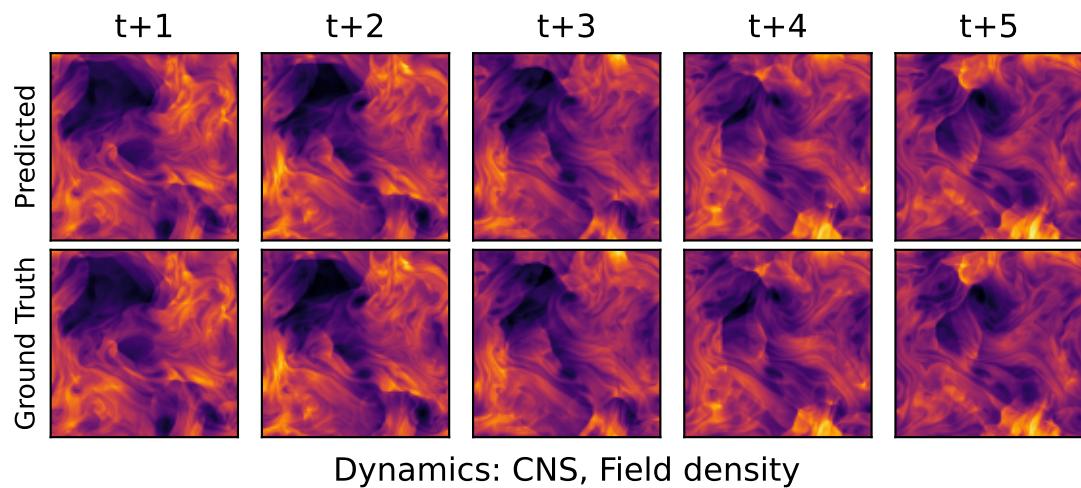
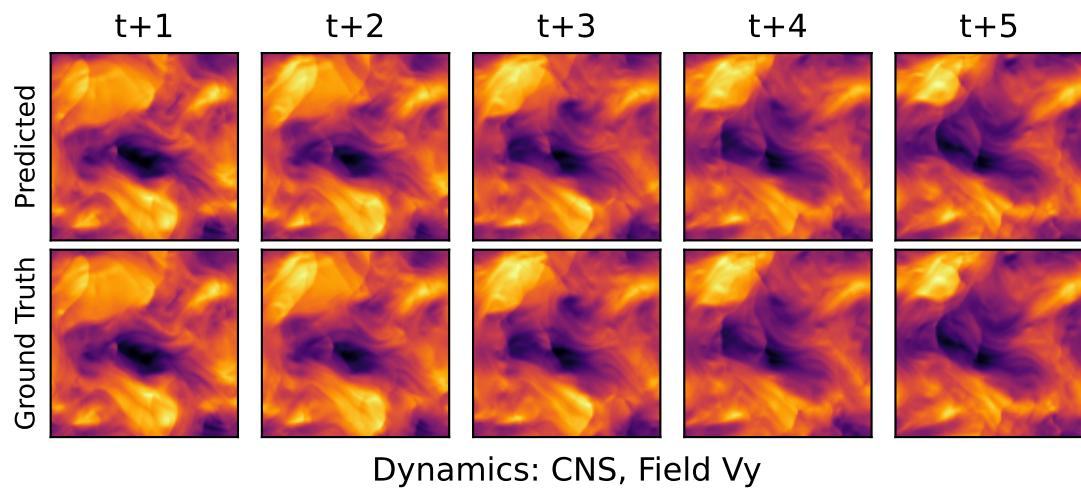
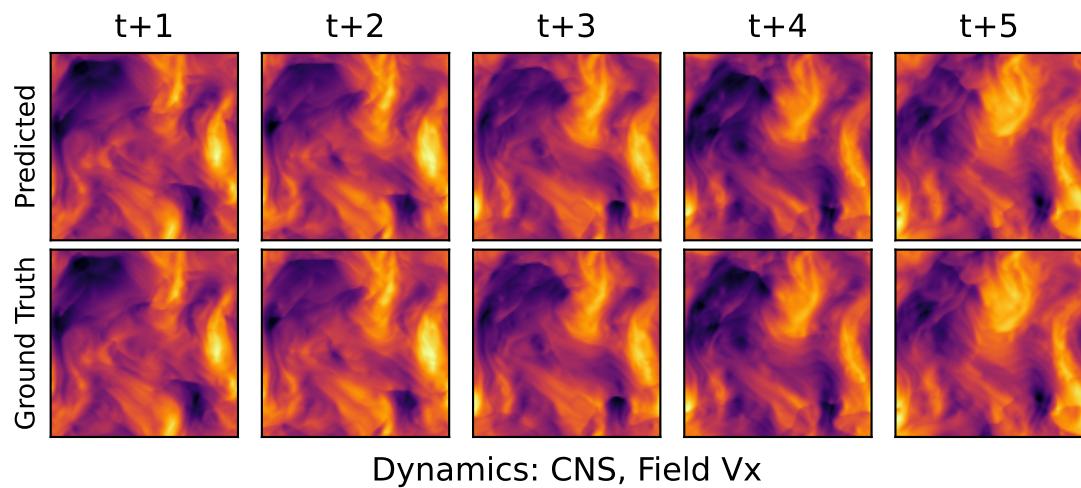
MODEL	# TRAINING SAMPLES (NRMSE $\times 10^{-1}$ )									
	100		200		400		600		800	
	T+1	T+5	T+1	T+5	T+1	T+5	T+1	T+5	T+1	T+5
VIDEOMAE (K400)	1.16	1.60	0.79	1.10	0.73	0.96	0.53	0.70	0.49	0.65
VIDEOMAE (SSV2)	0.98	1.42	0.75	1.03	0.62	0.84	0.55	0.74	0.51	0.67
MPP-AViT-B	0.60	1.15	0.37	0.77	0.27	0.66	.32	0.63	0.24	0.48

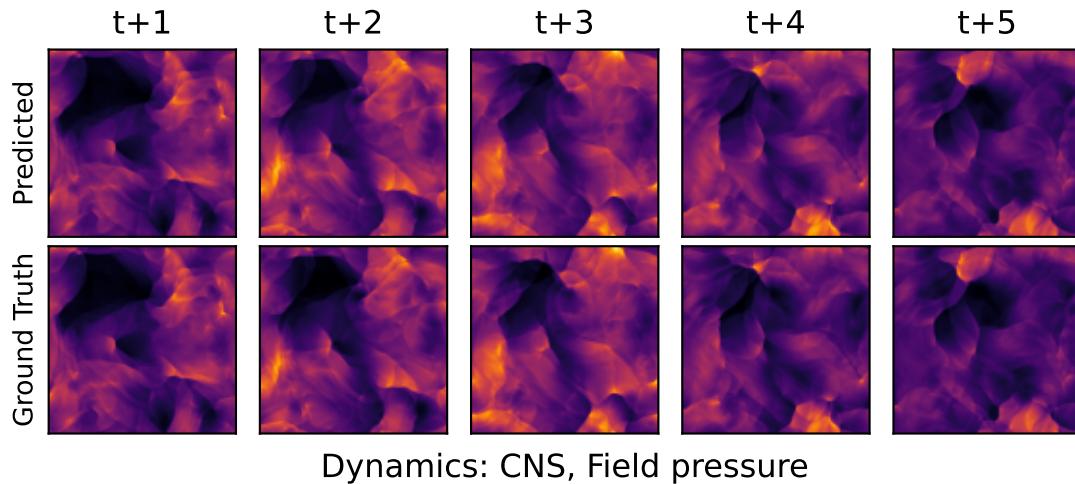


#### D.7 Finetuning Trajectories

After finetuning, we find that the patch-based instability mostly disappears. Again, videos displaying longer trajectories are available in the supplementary material.







## 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 paper claims the following. 1. We develop an approach that allows a single transformer model to learn from heterogeneous physical systems. This is developed in 4.2 and demonstrated in 5.1. 2. Training on heterogeneous systems can be beneficial in transfer settings on new physical behavior. This is shown on a toy example in 4.1 and more complex examples in 5.2 and 5.3. 2.

Guidelines:

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

### 2. Limitations

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

Answer: [\[Yes\]](#)

Justification: The conclusion extensively discusses the limits in terms of geometry and compute as well as more physically based limitations in terms of conservation laws. It also discusses the limited setting in which this has been applied thus far.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.

- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

### 3. Theory Assumptions and Proofs

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

Answer: [NA]

Justification: There are no theoretical results.

Guidelines:

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

### 4. Experimental Result Reproducibility

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

Answer: [Yes]

Justification: Code is provided and the settings for every experiment are recorded in the appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often

one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.

- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

## 5. Open access to data and code

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

Answer: [\[Yes\]](#)

Justification: Code is directly provided and the data is gathered from open well-established datasets.

Guidelines:

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

## 6. Experimental Setting/Details

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

Answer: [\[Yes\]](#)

Justification: Experimental settings are listed in the appendix.

Guidelines:

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

## 7. Experiment Statistical Significance

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

Answer: [No]

Justification: As these are explorations of large-scale models with multiple terabytes of training data, experiments are too expensive to repeat a sufficient number of times to gather useful estimates of statistical significance.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

## 8. Experiments Compute Resources

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

Answer: [Yes]

Justification: These are listed in the appendix.

Guidelines:

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

## 9. Code Of Ethics

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

Answer: [Yes]

Justification: The submission obeys copyright law, does not involve human subjects, and has no immediate negative impacts on society.

Guidelines:

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

## 10. Broader Impacts

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

Answer: [\[Yes\]](#)

Justification: Impact statement included in appendix.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

## 11. Safeguards

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

Answer: [\[NA\]](#)

Justification: There are no major risk associated with model use.

Guidelines:

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

## 12. Licenses for existing assets

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

Answer: [\[Yes\]](#)

Justification: The only existing asset in use is the training data which is appropriately cited.

Guidelines:

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

## 13. New Assets

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

Answer: [\[Yes\]](#)

Justification: The only new asset included is the model which is covered by the repository license.

Guidelines:

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

## 14. Crowdsourcing and Research with Human Subjects

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

Answer: [\[NA\]](#)

Justification: Submission does not involve crowdsourcing.

Guidelines:

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

**15. Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

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

Answer: [NA]

Justification: No crowdsourcing or human subject research.

Guidelines:

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