# BCAT: A Block Causal Transformer for PDE Foundation Models for Fluid Dynamics

YUXUAN LIU, University of California, Los Angeles, United States
JINGMIN SUN, Johns Hopkins University, United States
HAYDEN SCHAEFFER*, University of California, Los Angeles, United States

**Background:** Foundation models have demonstrated remarkable generalization in vision and language, but their accurate use for scientific computing remains challenging. For two-dimensional fluid-dynamics prediction, current transformer-based models are often hindered by limited context and rigid causality information, as they typically map fixed windows to the next step.
**Objectives:** We aim to develop a unified architecture for heterogeneous fluid-dynamics families. The goal is to better capture the complex spatio-temporal dependencies inherent in nonlinear fluid dynamics.
**Methods:** We introduce BCAT, a block-causal transformer for autoregressive prediction of two-dimensional fluid-dynamics solutions. Our approach adapts block-causal attention to next-frame prediction, leveraging previous frames as contextual priors. The model is trained on 69K trajectory sequences from PDEBench, PDEArena, and CFDBench using the Muon optimizer.
**Results:** BCAT achieved an average relative $L^2$ error of 1.18% across the six evaluated fluid families, reducing error by 45% relative to a 3.4-times-larger baseline. In ablations, next-frame prediction was 3.5x more accurate than the corresponding next-token variant. A five-seed analysis, noisy-input stress test, long-rollout analysis, and physics diagnostics further characterize the model.
**Conclusions:** The block-causal framework captures spatial interactions within frames while enforcing temporal causality across frames. BCAT provides an accurate and computationally efficient model for the two-dimensional fluid-dynamics benchmarks considered here.

**JAIR Associate Editor:** Tongliang Liu

**JAIR Reference Format:**
Yuxuan Liu, Jingmin Sun, and Hayden Schaeffer. 2026. BCAT: A Block Causal Transformer for PDE Foundation Models for Fluid Dynamics. *Journal of Artificial Intelligence Research* 87, Article 19 (November 2026), 27 pages. DOI: 10.1613/jair.1.22089

## 1 Introduction

Fluid mechanics is a fundamental area of study within physics and engineering, describing a wide range of phenomena through modeling of pressure, velocity, and viscosity. It is used in various applications, including climate forecasting, energy generation in wind and hydropower systems, aerodynamics and aircraft design, blood circulation analysis, and more. Due to the highly nonlinear interactions among multiple physical scales, predicting and simulating fluid behavior remains a challenging task. The computational problems become even

*Corresponding Author.

Authors' Contact Information: Yuxuan Liu, ORCID: 0000-0001-5788-1329, yxliu@ucla.edu, University of California, Los Angeles, Los Angeles, California, United States; Jingmin Sun, ORCID: 0000-0002-5728-2606, jsun143@jh.edu, Johns Hopkins University, Baltimore, Maryland, United States; Hayden Schaeffer, ORCID: 0000-0003-1379-1238, hayden@math.ucla.edu, University of California, Los Angeles, Los Angeles, California, United States.



This work is licensed under a Creative Commons Attribution International 4.0 License.

© 2026 Copyright held by the owner/author(s).
DOI: 10.1613/jair.1.22089

more difficult in settings with limited observations or noisy measurements. Hence, accurately predicting fluid behavior in such regimes remains a challenge for scientific machine learning.

While foundation models like GPT (Brown et al. 2020; Radford, Narasimhan, et al. 2018; Radford, J. Wu, et al. 2019), DALL-E (Ramesh, Dhariwal, et al. 2022; Ramesh, Pavlov, et al. 2021), Stable Diffusion (Rombach et al. 2022), LLAMA (Touvron, Lavril, et al. 2023; Touvron, Martin, et al. 2023), Phi (Abdin, Aneja, Awadalla, et al. 2024; Abdin, Aneja, Behl, et al. 2024), Gemma (Team et al. 2024), and Claude have demonstrated remarkable generalization capabilities in language and vision tasks (Bommasani et al. 2021), they have not accurately addressed scientific computing tasks such as simulating partial differential equations (PDEs). These tasks not only require precise predictions but also the ability to learn the underlying physical rules. PDE foundation models aim to address this challenge by using a single deep neural network (DNN) to simultaneously encode and approximate the behavior of a variety of physical models. The objective is to develop a unified DNN capable of predicting multiple physical systems with the ability to simulate solutions in unseen parameter regimes, i.e. out-of-domain predictions. Recent progress in PDE foundation models includes the Predicting Operators and Symbolic Expressions (PROSE) (Jollie et al. 2024; Y. Liu, Sun, et al. 2024; Y. Liu, Z. Zhang, et al. 2024; Sun, Y. Liu, et al. 2024; Sun, Z. Zhang, et al. 2024), Multiple Physics Pretraining (MPP) (McCabe et al. 2023), In-Context Operator Network (ICON) (Cao et al. 2024; L. Yang, S. Liu, et al. 2023; L. Yang, Meng, et al. 2023; L. Yang and Osher 2024), DPOT (Hao et al. 2024), Fourier Forecasting Network (FourCastNet) (J. Pathak et al. 2022), Poseidon (Herde et al. 2024), and Aurora (Bodnar et al. 2024).

**Main Contributions:** One of the key technical components of foundation models is the transformer architecture, known for its ability to efficiently model complex and long-range relationships in data through attention mechanisms. However, current transformer structures utilized in PDE foundation models are hindered by limited context and causality information, since many existing approaches learn to map a fixed window to the next step. In this work, we introduce the **BCAT** model, based on the **B**lock **CA**usal **T**ransformer architecture[1], adapting the attention mechanism for spatio-temporal dynamics modeling. The BCAT model performs next frame prediction instead of next token prediction, which enhances computational efficiency, allows for more token interactions to capture complex dependencies, and utilizes flexible history context windows. Our contributions are listed below.

- We adapt a block-causal transformer structure to autoregressive prediction of two-dimensional fluid dynamics. BCAT uses next-frame prediction to model spatiotemporal context and achieves a 3.5x accuracy improvement over the corresponding next-token variant.
- BCAT's next-frame objective enables flexible history windows and parallel training loss computation. At inference, key-value caching further reduces measured wall-clock time for the stated hardware configuration.
- Across six two-dimensional fluid families from PDEBench, PDEArena, and CFDBench, BCAT obtains the best average relative $L^2$ error in our zero-shot comparison. Relative to the 3.4-times-larger DPOT-L baseline, it reduces the average error by 45%. We further evaluate low-data adaptation on a turbulence dataset.

## 2 Related Works

### 2.1 PDE Foundation Models

Foundation models are large-scale, pre-trained models that act as flexible bases for a wide range of tasks, leveraging vast amounts of data and computational resources to enable adaptability and generalization across diverse applications (Achiam et al. 2023; Bommasani et al. 2021; Brooks et al. 2024; Dubey et al. 2024; A. Yang et al. 2024). However, their applications to scientific computing are limited, where high precision is needed for numerical tasks. Recently, several approaches have been proposed to explore and develop foundation models for scientific computing. PROSE (Jollie et al. 2024; Y. Liu, Sun, et al. 2024; Y. Liu, Z. Zhang, et al. 2024; Sun, Y. Liu,

[1]The code is available at: https://github.com/felix-lyx/bcat.

et al. 2024; Sun, Z. Zhang, et al. 2024) takes multimodal input to construct solution operators, encoding PDEs as symbolic input to provide additional information and allows zero-shot generalization. ICON (Cao et al. 2024; L. Yang, S. Liu, et al. 2023; L. Yang, Meng, et al. 2023; L. Yang and Osher 2024) uses an in-context learning approach, allowing the model to perform few-shot learning by observing input-output pairs. Some methods focus on the sequential modeling aspect, and various models have been designed to autoregressively predict future states. As additional information is not included, these methods are not able to zero-shot generalize to out-of-distribution data, but can leverage pretraining data to be efficiently fine-tuned (Subramanian et al. 2024). Examples of such models include MPP (McCabe et al. 2023), which uses axial attention to map a fixed context window to future states, and DPOT (Hao et al. 2024), which uses Fourier attention and stabilizes the autoregressive rollout process through a denoising training process. Poseidon (Herde et al. 2024) is a swin-transformer-based model, and the model requires fine-tuning for downstream tasks.

### 2.2 Transformers for Time Series

Time series data takes a similar form as language modeling data, where numerical inputs replace discrete tokens. LLMs and other transformer-based models have been applied to various time series tasks, including forecasting (Jin et al. 2023), classification (M. Liu et al. 2021), and anomaly detection (Zhou and R. Yu 2024). One key challenge in adapting pretrained transformers for time series forecasting lies in the encoding of numerical data. In (Gruver et al. 2024), scalar time series are encoded as strings, which can be directly processed by pretrained LLMs. In (Chang et al. 2024), a separate alignment stage is used to allow LLMs to process patched time series. However, as shown in (Tan et al. 2024), while attention layers can be useful for time series, pretrained LLMs may not outperform simple transformers trained from scratch. Another challenge lies in processing spatio-temporal dynamics, due to the complex interplay between spatial and temporal dependencies. Accurately capturing these interactions often requires models that can handle high-dimensional data, irregular sampling, and non-linear relationships, all while maintaining scalability and computational efficiency. Time-dependent PDEs, climate modeling, and videos can all be seen as examples of spatio-temporal data. The high-dimensional nature requires specific tokenization and processing techniques, such as graph (L. Liu et al. 2024) and Fourier mixer (J. Pathak et al. 2022) for spatial dependencies.

## 3 Methods

### 3.1 Problem Setup

Consider parametric families of two-dimensional time-dependent nonlinear PDEs, whose state-variables of interest are represented by $\boldsymbol{u}(\boldsymbol{x}, t) \in \mathbb{R}^d$, where $d$ can be up to 4 in our tests, with $\boldsymbol{x} \in \Omega \subseteq \mathbb{R}^2$. Given data up to $T_0$ frames, i.e. the sequence:

$$\{\boldsymbol{u}(\cdot, t_i) \mid 0 \le i < T_0\},$$

the goal is to predict the subsequent $T$ frames:

$$\{\boldsymbol{u}(\cdot, t_i) \mid T_0 \le i < T_0 + T\}.$$

In our experiments, we set $T_0 = T = 10$, that is, 10 frames are given as inputs and the model predicts 10 future frames.

### 3.2 Model Overview

The BCAT model is illustrated in Figure 1. The model comprises two main components: (1) image tokenization and de-tokenization through patch embedding, and (2) the standard decoder-only transformers similar to GPT-2 (Radford, J. Wu, et al. 2019). The input data, $\{\boldsymbol{u}(\cdot, t_i) \mid 0 \le i < T_0\}$, are first partitioned into patches, which are subsequently flattened and projected into a sequence of visual tokens. The transformer backbone serves as a

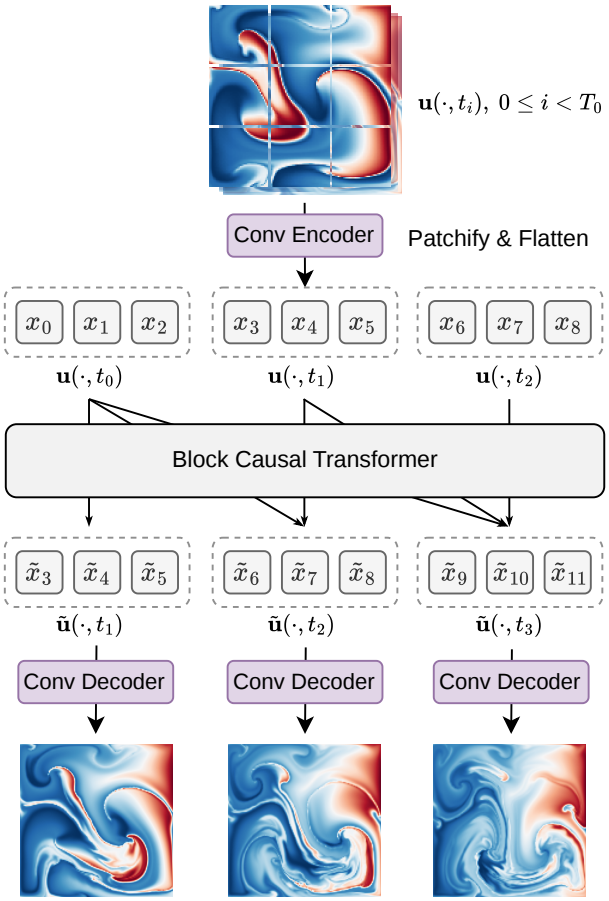



Fig. 1. **BCAT model overview.** The inputs to the model are the initial frames sampled from the datasets, which are patchified and converted into a sequence of features. Transformer layers then take the input sequence to perform next-frame prediction, where a block-causal mask allows spatial interactions within a frame and temporal causality across varying-length context windows. The processed features are projected with a convolutional patch-reconstruction layer to form the final predictions.

token mixer, processing these tokens to autoregressively predict future states $\{\boldsymbol{u}(\cdot, t_i) \mid T_0 \le i < T_0 + T\}$. We use RMS norm (B. Zhang and Sennrich 2019) in place of the usual layer norms and find query-key normalization (Henry et al. 2020) helpful in stabilizing the training process.

Training is performed end-to-end by minimizing the mean squared error between the (normalized) predicted and ground truth sequences. Detailed descriptions of the two components are provided in the following sections.

### 3.3 Image Tokenization via Patch Embedding

Inspired by (Dosovitskiy et al. 2021a), we employ a patch embedding layer to convert the input data into a sequence of visual tokens. Each frame in the input data, $\boldsymbol{u}(\cdot, t_i)$, is first divided into non-overlapping patches, which are then flattened and projected into a sequence of visual tokens. For a maximum of $(T_0 + T)$ data frames, we simply concatenate the visual tokens from each frame to form the input sequence to the transformer backbone.

Let $R_x$ and $R_y$ denote the space resolutions and $c$ be the number of channels (physical fields), i.e., $\boldsymbol{u}(\cdot, t) \in \mathbb{R}^{R_x \times R_y \times c}$. In the experiments, we fix the spatial resolution to be $R_x = R_y = 128$, consistent with the majority of datasets considered. Data with different spatial resolutions are upsampled via interpolation or downsampled to match this resolution. To unify the number of channels (physical fields) across various PDEs, we zero-pad all input data to $c = 4$ to match the data size with the largest number of channels.

Choosing a patch size of $P$ for each spatial dimension transforms each frame $\boldsymbol{u}(\cdot, t)$ into $(R_x/P) \times (R_y/P)$ patches, resulting in a sequence of up to $(T_0 + T) \times (R_x/P) \times (R_y/P)$ visual tokens. We observed that the choice of patch size $P$ significantly influences the performance of the model, which is quantified in Section 4.4. For our model, we select $P = 8$, the smallest size that maintains a reasonable sequence length. This choice results in a maximum sequence length of $(10 + 10) \times (128/8) \times (128/8) = 5120$.

After the transformer backbone, the sequence of tokens is projected back to physical space and then deflattened to reconstruct the predicted future states $\{\boldsymbol{u}(\cdot, t_i) \mid T_0 \le i < T_0 + T\}$.

### 3.4 Preliminary: Next Token Prediction

In both language modeling and autoregressive image generation, next token prediction is the key learning task. For language models, the model predicts the probability distribution of the next token given the previous tokens, enabling the generation of coherent sentences by sampling tokens sequentially (Radford, J. Wu, et al. 2019; Vaswani et al. 2017). For autoregressive image generation, images are treated as sequences of pixels or patches, with each element predicted based on prior elements in the sequence (M. Chen et al. 2020). Both applications rely on the autoregressive causal property, where future predictions depend only on prior context. More precisely, given a sequence of discrete tokens $X = (x_1, x_2, \ldots, x_S)$, the modeling assumption for most language models is that:

$$p(x_1, \ldots, x_S) = \prod_{s=1}^{S} p(x_s | x_1, x_2, \ldots, x_{s-1}). \tag{1}$$

This is a reasonable assumption for sentence generation, as language typically follows a left-to-right sequential structure. However, for image generation, such dependencies are not as clear. Unlike text, images exhibit spatial invariance: the pixel/patch in the top-left corner does not inherently depend on the pixel/patch in the bottom-right corner.

### 3.5 Next Frame Prediction

While the causal property is less intuitive for spatial dependencies, it is a natural assumption for temporal dynamics, especially for Markovian sequences. Next frame prediction is the natural solution that addresses these concerns, where visual tokens within the same frame can attend to each other and all tokens for the same frame will be generated in parallel conditioned on previous frames.

Denote the input token sequence as $X = (x_1, \ldots, x_S)$ and the BCAT model as $f_\theta$, where $S = (T + T_0) \times N$ is the total sequence length and $N = (R_x/P) \times (R_y/P)$ is the number of tokens per frame. Instead of learning to predict the immediate next token,

$$(x_1, x_2, \ldots, x_s) \xmapsto{f_\theta} (x_2, x_3, \ldots, x_{s+1}),$$

the BCAT model learns to predict the next frame given the previous frames:

$$(x_1, x_2, \ldots, x_s) \xmapsto{f_\theta} (x_{N+1}, x_{N+2}, \ldots, x_{s+N}).$$

To achieve this, similar to causal masks in transformers to mask out future tokens (Vaswani et al. 2017), we use a block lower triangular causal mask (Cao et al. 2024; Tian et al. 2024) to allow spatial dependencies and temporal causality:

$$M = \begin{bmatrix} \mathbf{1}_{N\times N} & \mathbf{0}_{N\times N} & \mathbf{0}_{N\times N} & \cdots & \mathbf{0}_{N\times N} \\ \mathbf{1}_{N\times N} & \mathbf{1}_{N\times N} & \mathbf{0}_{N\times N} & \cdots & \mathbf{0}_{N\times N} \\ \mathbf{1}_{N\times N} & \mathbf{1}_{N\times N} & \mathbf{1}_{N\times N} & \cdots & \mathbf{0}_{N\times N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{1}_{N\times N} & \mathbf{1}_{N\times N} & \mathbf{1}_{N\times N} & \cdots & \mathbf{1}_{N\times N} \end{bmatrix}. \tag{2}$$

Here $\mathbf{1}_{N\times N}$ and $\mathbf{0}_{N\times N}$ represent constant $1/0$ matrices of dimension $N \times N$.

We compare next-frame prediction and next-token prediction in Section 4.4, showing the accuracy gains from using next-frame prediction in the prediction tasks. For autoregressive inference, next-token prediction requires $T \times N$ prediction calls to generate $T$ frames, whereas next-frame prediction requires $T$ calls. Key-value caching further improves the inference efficiency; Section 4.5 reports wall-clock measurements for this setting.

A similar block lower-triangular mask has been used for "next function prediction" and "next scale prediction" in VICON and VAR (Cao et al. 2024; Tian et al. 2024). Our contribution is not the masking pattern itself: BCAT adapts this pattern to autoregressive two-dimensional fluid-dynamics forecasting, evaluates it against retrained PDE baselines, and studies its interaction with next-frame prediction and Muon optimization.

*Difference from fixed-window update rules.* MPP (McCabe et al. 2023) and DPOT (Hao et al. 2024) learn fixed-window update rules, whereas BCAT can condition on a flexible history window through temporal causality. This causal formulation permits parallel loss computation over future frames during training and supports key-value caching at inference.

### 3.6 Muon Optimizer

The Muon optimizer (Jordan et al. 2024) is a variant of SGD with preconditioned momentum. Muon is designed to optimize 2D parameters of the hidden layers, and the preconditioning step orthogonalizes the momentum matrix using Newton-Schulz matrix iteration (Schulz 1933). When a network is trained with Muon optimizer, scalar/vector parameters of the network, as well as the input and output layers, should be optimized by a standard optimizer, such as AdamW (Loshchilov and Hutter n.d.)

Empirically, Muon demonstrated strong performance in training large language models (Jordan et al. 2024), significantly outperforming AdamW in terms of convergence speed with $< 1\%$ computational overhead. More recently, (J. Liu et al. 2025) optimizes Muon's distributed implementation and studies the scaling law for training larger LLMs with Muon optimizer. Observing the success of Muon optimizer in training LLMs, we train BCAT using a combination of Muon optimizer (for transformer 2D parameters) and AdamW optimizer, which significantly outperforms BCAT trained only with AdamW optimizer. BCAT is the first model using Muon optimizer for scientific computing and prediction tasks, and we observe that Muon optimizer is able to better approximate high frequency or high contrast information. The comparison is shown in Section 4.4.

### 3.7 Implementation Details

The architecture utilizes a standard decoder-only transformer consisting of prenorm transformer blocks. We use RMS norm (B. Zhang and Sennrich 2019) in place of layer norms and add query-key normalizations (Henry et al. 2020) after the attention projections, which we found to help stabilize the training process. In the feedforward

network (FFN) of transformer layers, we use SwiGLU activation function (Shazeer 2020), and to keep a similar parameter count, the FFN hidden dimension is rescaled to be $\frac{8}{3}$x the feature dimension instead of the more common 4x. The patch embedding layer is efficiently implemented as a convolution layer with the same kernel size and stride (equivalent to a patchwise linear projection). The output reconstruction uses the corresponding convolutional projection, and inputs are zero-padded to four channels for a shared interface. More architecture and training details are included in Appendix B.

## 4 Experiments

In this section, we first introduce the experiment setup and the baseline models we compare against. We present the main results comparing zero-shot performance in Section 4.2, and then study a single in-distribution adaptation setting in Section 4.3. To demonstrate the efficiency of BCAT, we compare inference speed and GPU memory usage in Section 4.5. Various ablation studies for important architecture choices are included in Section 4.4. We include example visualizations of the BCAT model output and comparison of outputs from different models in Appendix D.

### 4.1 Experiment Setup

*Dataset.* The dataset we use consists of 6 parametric families of PDEs that model fluid dynamics in different regimes (some families contain subfamilies using different ranges of physical parameters, so different data distributions exist even within a family). We collect the dataset from 3 heterogeneous sources: PDEBench (Takamoto et al. 2022), PDEArena (Gupta and Brandstetter 2022), and CFDBench (Luo et al. 2023). The dataset includes shallow water equations and the Navier-Stokes system with incompressible and compressible flow, regular and complex geometries, and different buoyancy settings. For datasets that do not provide a train/val/test splitting, we use the standard 80%/10%/10% splitting. The total amount of data samples is about 69K trajectory sequences (time homogeneity can be used to create more training input/output pairs), which is the current amount used in comparable models. For a shared experimental interface, all data are resampled to $128 \times 128$ and retain their dataset-specific temporal resolutions. More details are included in Appendix A.

*Evaluation Metric.* We use the relative $L^2$ norm as the evaluation metric. More precisely, given the ground truth $\boldsymbol{u}$ and model's prediction $\tilde{\boldsymbol{u}}$, we compute the (time-averaged) relative $L^2$ error:

$$\frac{1}{T}\sum_{i=T_0}^{T_0+T-1}\frac{\|\boldsymbol{u}(\cdot,t_i)-\tilde{\boldsymbol{u}}(\cdot,t_i)\|_2}{\|\boldsymbol{u}(\cdot,t_i)\|_2+\varepsilon}, \tag{3}$$

where $T_0 = 10$ is the number of input steps, $T = 10$ is the number of output steps, and $\varepsilon = 10^{-7}$. Note that for the Navier-Stokes dataset from PDEArena, the temporal grid resolution is only 14, thus we set $T = 4$ for this dataset only. The average used in the last column of Table 1 is the average of the relative $L^2$ errors over the 6 families, i.e., the average over each row of the table.

*Baselines and Comparisons.* We evaluate our BCAT model against the following baseline methods. DeepONet (L. Lu et al. 2019) and FNO (Z. Li, Kovachki, et al. 2020) are two widely used single-operator learning techniques designed to efficiently approximate solution operators for PDEs. UNet (Ronneberger et al. 2015) is a classic convolutional neural network model for image processing, known for its symmetric hierarchical structure that enables it to capture both contextual information and fine details for pixel-wise predictions. ViT (Dosovitskiy et al. 2021b), a transformer-based image processing model, is notable for its ability to capture global dependencies in images and its scalability across various image and video processing tasks. Lastly, we test against three recent PDE foundation models, PROSE-FD (Y. Liu, Sun, et al. 2024), MPP (McCabe et al. 2023), and DPOT (Hao et al. 2024). PROSE-FD fuses data input and symbolic equation encoding for multimodal operator learning. Both MPP

Table 1. **Main Results and Comparisons with Baselines**. The numbers reported are relative $L^2$ errors (%). The averages are taken with respect to the 6 distinct families listed in the columns of the table. We **bold** the best result in each column.

| Model | Param | PDEBench | | | PDEArena | | CFDBench | Average |
|---|---|---|---|---|---|---|---|---|
| | | SWE | CNS* | INS | NS | NS-cond | - | |
| DeepONet | 3.5M | 3.55 | 7.41 | 64.61 | 35.33 | 51.85 | 12.50 | 29.21 |
| FNO | 0.6M | 3.71 | 6.31 | 36.84 | 38.67 | 55.63 | 8.52 | 24.95 |
| UNet | 5.6M | 0.33 | 3.19 | 3.43 | 12.56 | 16.82 | 0.76 | 6.18 |
| MPP-B | 116M | 1.02 | 1.90 | 7.52 | 5.71 | 12.57 | 1.23 | 4.99 |
| ViT | 162M | 0.25 | 1.49 | 2.82 | 7.05 | 12.41 | 0.55 | 4.10 |
| MPP-L | 407M | 0.47 | 1.53 | 6.42 | 4.64 | 9.64 | 0.73 | 3.91 |
| DPOT-M | 122M | 0.54 | 1.01 | 5.20 | 4.92 | 8.55 | 0.64 | 3.47 |
| PROSE-FD | 165M | 0.28 | 1.41 | 2.75 | 5.27 | 9.61 | 0.61 | 3.32 |
| DPOT-L | 523M | 0.15 | 0.89 | 4.08 | 2.21 | 5.29 | **0.34** | 2.16 |
| BCAT | 156M | **0.10** | **0.39** | **1.34** | **1.59** | **3.13** | 0.52 | **1.18** |

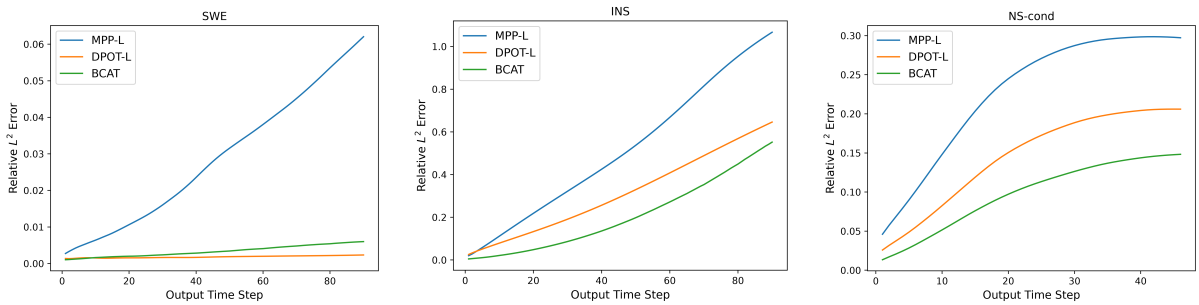
and DPOT autoregressively predict PDE solutions, with MPP based on Axial-ViT and DPOT based on Adaptive Fourier Neural Operators.

Additional details about these baselines can be found in Appendix B.3. We did not include Poseidon (Herde et al. 2024) as it does not generate meaningful results without fine-tuning, i.e., Poseidon cannot produce zero-shot predictions. Additionally, Poseidon cannot process history context, and thus is unable to handle partial observations of the state variables (e.g. the pressure and inhomogeneous forcing terms are not observed in PDEArena, but can be implicitly inferred from context). Some other transformer-based methods (Z. Li, Meidani, et al. 2022; H. Wu et al. 2024) are not considered as their model sizes are much smaller, both MPP and DPOT have comparable or larger model sizes than BCAT. Note that DeepONet and FNO are designed for single tasks but can be extended to multi-task prediction using fine-tuning (Subramanian et al. 2024; Z. Zhang et al. 2024). Except for the in-distribution adaptation in Section 4.3, all comparisons are performed in a zero-shot setting, which is also used by MPP and DPOT. As a reference, in Appendix B.5, we train separate single-task models for each family, and the results show no significant difference.

## 4.2 Main Results

We present the main results in Table 1, where we report the relative $L^2$ error (%) for each family of equations and their average. We use the same zero-shot setting for all models: a single model is trained to predict all families of equations without any fine-tuning. In this comparison, BCAT has the best average and the best result on five of six families. Appendix C reports five-seed BCAT variability. On the more complicated PDEArena NS-cond dataset, BCAT has a large margin over the compared baselines.

We further study long-horizon autoregressive rollouts on longer datasets PDEBench SWE, PDEBench INS, and PDEArena NS-cond. For each, we recursively feed generated outputs back as inputs and compare BCAT with MPP-L and DPOT-L in Figure 2. BCAT has the lowest per-step error for INS and NS-cond throughout the shown rollout; on SWE, DPOT-L remains slightly lower while both are substantially more stable than MPP-L. The trajectories have different dynamics: NS-cond approaches a plateau, whereas SWE and INS continue accumulating error.




(a) PDEBench SWE (b) PDEBench INS (c) PDEArena NS-cond

Fig. 2. **Long-horizon rollout error.** Relative $L^2$ error at each generated time step for the three datasets with trajectories long enough for extrapolative rollouts. All models use their generated outputs as subsequent inputs. The models are ordered from lower complexity (SWE) to higher complexity (NS). NS-cond plateaus late in the trajectory, whereas SWE and INS continue accumulating error.

Table 2. **Comparing In-Distribution Adaptation on the PDEBench Turbulence Dataset.** Each model is fine-tuned for 500 gradient updates with a batch size of 64 (20 epochs).

| Model | Param | Testing Relative $L^2$ Error (%) |
|---|---|---|
| MPP-L | 407M | 7.35 |
| DPOT-L | 523M | 6.76 |
| BCAT | 156M | 3.87 |

## 4.3 In-Distribution Adaptation

We focus on and compare the zero-shot performance of the models considered, since zero-shot performance is an unbiased metric for the intrinsic capabilities of foundation models. For example, when GPT-2 (Radford, J. Wu, et al. 2019) was introduced, only the zero-shot performance was studied. Additionally, it may be difficult to directly compare PDE foundation models fine-tuned with different algorithms, such as full/partial parameter fine-tune and LoRA (E. J. Hu et al. 2022). Fine-tuning could obscure a model's weakness through overfitting on task-specific objectives.

We evaluate low-data, in-distribution adaptation using the same full-parameter fine-tuning method for a fair comparison on the PDEBench Turbulence dataset, where the initial conditions lead to turbulent flow. We fine-tune pretrained BCAT, DPOT-L, and MPP-L for 500 gradient updates with a batch size of 64. Given the relatively small 1.6K-example training dataset, this is equivalent to 20 epochs and tests a low-data, low-compute regime. Table 2 shows that BCAT outperforms the two larger baselines in this setting. Training BCAT from scratch for the same 500 steps gives 29.95% relative $L^2$ error, demonstrating the benefit of pretraining.

BCAT's next-frame objective is well suited to this low-data adaptation setting. With a fixed compute budget, its causal formulation yields loss terms for multiple future frames in one forward pass, whereas the fixed-window MPP and DPOT updates yield one output step per pass. In this experiment, BCAT fine-tuning took about 10 minutes, compared with more than 20 minutes for MPP-L and DPOT-L.

Table 3. **Compare BCAT model trained with AdamW and Muon optimizer.** The numbers reported are relative $L^2$ errors (%). The averages are taken with respect to the 6 distinct families listed in the columns of the table.

| Optimizer | Param | PDEBench | | | PDEArena | | CFDBench | Average |
|---|---|---|---|---|---|---|---|---|
| | | SWE | CNS* | INS | NS | NS-cond | - | |
| AdamW | 156M | 0.26 | 0.61 | 2.19 | 2.61 | 5.15 | 0.70 | 1.92 |
| Muon | 156M | 0.10 | 0.39 | 1.34 | 1.59 | 3.13 | 0.52 | 1.18 |

## 4.4 Ablation Studies

In this section, we present the results of our ablation studies to validate some key optimizer and architecture choices.

*Muon Optimizer vs. AdamW Optimizer.* We compare the BCAT model trained using a combination of Muon and AdamW optimizer (we refer to this as Muon for brevity) versus AdamW optimizer alone. When trained with Muon optimizer, the transformer layer matrix weights are optimized using Muon, and the other weights (scalar, vector, embedding, and initial/final projection layer weights) are optimized using AdamW, following (Jordan et al. 2024). We kept the same optimizer hyperparameters (originally tuned for AdamW), and only tuned the learning rate scheduler hyperparameters. We found that more decay steps help stabilize the training process for Muon.

The results are shown in Table 3, where Muon outperforms AdamW in all datasets, improving BCAT's performance in all cases. In terms of average errors, Muon reduces the error from 1.92% to 1.18%, at the cost of <1% computational overhead. We visualize and compare the output of BCAT trained using Muon and AdamW in Figure 3. We observe that, while the spatio-temporal error distributions are similar when trained with the two optimizers (in high contrast and high frequency regions), the error magnitude is much lower when trained with Muon. Appendix C.4 further reports controlled convergence and frequency-domain diagnostics.

*Next Frame Prediction vs. Next Token Prediction.* We directly compare the performance of BCAT with next frame prediction and next token prediction on the PDEArena NS dataset. We use the same model and training setup for both models, except that the next-token-prediction variant is trained to predict the next token instead of the next frame (and the mask is causal instead of block causal). When trained on a mixture of all 6 families, the next-token-prediction variant fails to converge, so we only train it on the PDEArena NS dataset (the shorter dataset with 14 frames).

The results are shown in Table 4. We observe that the next-frame-prediction model achieves a 3.5x better relative $L^2$ error compared to the next-token-prediction variant, demonstrating the effectiveness of our proposed training objective. We also compute the error on the training dataset and the "next-token error." As the PDEArena NS dataset is a relatively small dataset, both models are overfitting to the training dataset to some degree; however, there is a performance difference between the two as shown by the error gap.

The "next-token error" is computed assuming all previous tokens are exactly known (i.e. no prior error). More precisely, during testing time, we input the full trajectory to the model with proper masks to prevent data leaking, similar to training time. Both models are only asked to predict the immediate next token/frame and thus no autoregressive rollout is needed. We are thus measuring the amount of error over one step, i.e., the "local error". In this case, we observe that the performance of both models are relatively similar. This shows that the two models have similar (local) approximation power, but the next-frame-prediction model is better conditioned for multi-step prediction. Specifically, the next-frame-prediction has a slower accumulation of error rate than the next-token-prediction approach. As discussed in Section 3.5, to autoregressively generate $T$ future frames, the next-frame-prediction model only needs to be recursively called $T$ times, while the next-token-prediction

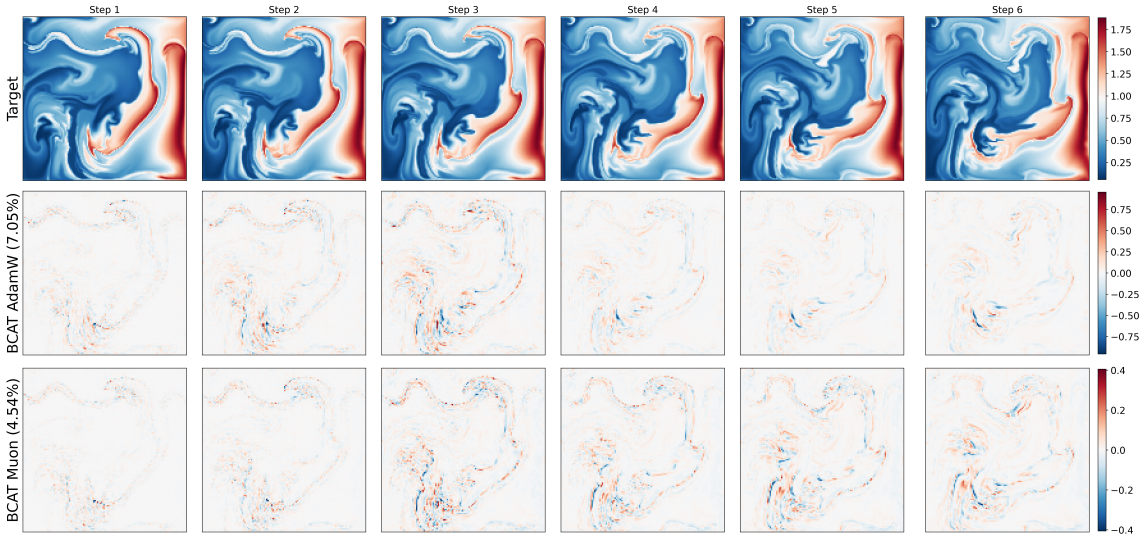



Fig. 3. **Comparing BCAT trained with Muon vs. AdamW optimizer.** The target (first row) is the first 6 output steps from the PDEArena Navier-Stokes (conditioned) dataset (particle density channel). For each optimizer (each row), we display the difference between the target and model output. Relative $L^2$ errors for the full trajectories are listed after the optimizer names.

Table 4. **Comparing next frame prediction and next token prediction.** The errors shown are relative $L^2$ errors (%) for PDEArena NS dataset. *The Next-Token error is computed assuming all previous tokens are exactly known (i.e., the full trajectory is inputed to the model, similar to training time). It is not a standard metric and is only used to provide a comparison.

| Method | Param | Testing Error | Training Dataset Error | Next-Token Error* |
|---|---|---|---|---|
| Next token prediction | 156M | 5.49 | 1.62 | 1.01 |
| Next frame prediction | 156M | 1.59 | 0.91 | 0.81 |

variant needs to be recursively called $T \times N$ times (where $N$ is the number of tokens per frame). This may be the mechanism that leads to a significant reduction in error accumulation for the next-frame-prediction model.

*Comparison with Spatio-Temporal Attention.* The forward prediction task for 2D fluid dynamics is similar to a video generation task, where the state variable at each timestamp is akin to a frame in the video. In (Bertasius et al. 2021), several different forms of self-attention for processing spatial-temporal dynamics are compared for video classification task. Our BCAT model flattens the patches from all frames into a single sequence, and can be seen as a variant of joint space-time self-attention discussed in (Bertasius et al. 2021). Similarly, MPP (McCabe et al. 2023) is a variant of axial self-attention. Another approach that has not appeared in the literature as a backbone to PDE foundation models is the divided space-time self-attention, where temporal attention and spatial attention are separately applied one after the other. We implement this variant and refer to it as the Time-then-Space model. The temporal attention is a causal self-attention and the spatial attention is a full self-attention. All other configurations and hyperparameters are kept the same as the BCAT model. Due to more attention layers, the Time-then-Space model is larger than BCAT (206M vs 156M). The results are shown in

Table 5. **Comparing the performance of BCAT and Time-then-Space Model.** The errors shown are relative $L^2$ errors (%).

| Model | Param | Average Testing Error |
|---|---|---|
| Time-then-Space | 206M | 1.19 |
| BCAT | 156M | 1.18 |

Table 6. **Comparing the performance of BCAT and ViT with different patch size.** The errors shown are relative $L^2$ errors (%).

| Model | Patch Size | Param | Average Testing Error |
|---|---|---|---|
| ViT | 16 | 162M | 4.10 |
| ViT | 8 | 155M | 3.34 |
| BCAT | 32 | 191M | 3.61 |
| BCAT | 16 | 163M | 1.99 |
| BCAT | 8 | 156M | 1.18 |

Table 5, where the BCAT model slightly outperforms the spatio-temporal variant. Consider the difference in model sizes, this shows that the joint space-time self-attention is more effective for the forward prediction task.

*Effect of Patch Size.* In the image tokenization step, the patch size $P$ to divide the image into non-overlapping patches is a key hyperparameter. Note that this is a tradeoff between model complexity and feature spatial resolution, as half the patch size makes the sequence length 4x longer, thus 16x the time complexity and 4x space complexity (assuming memory efficient attention), even though the model parameter size remains similar, since models with smaller patch size have fewer parameters in patch embedding due to smaller dimensions per patch. However, we observe that having a smaller patch size and hence more visual tokens significantly improves the performance for both BCAT and ViT. The results are shown in Table 6. For patch size 4 or below, we refer to (F. Wang et al. 2025) for a more comprehensive study on patch size scaling law (note that linear attention is used when the sequence length is beyond 4000).

Table 7. **Additional Ablation Studies.** Each row represents a different variant and we underline the difference with the BCAT model (last row). The errors shown are relative $L^2$ errors (%). *The training loss diverges when using Muon without QK-Norm, thus for this row only, we report the result when trained using AdamW.

| Activation | QK-Norm | Mask | Param | Average Testing Error |
|---|---|---|---|---|
| SwiGLU | ✗ | block causal | 155.6M | 2.04* |
| SwiGLU | ✓ | causal | 155.6M | 1.97 |
| GeLU | ✓ | block causal | 154.8M | 1.34 |
| SwiGLU | ✓ | block causal | 155.6M | 1.18 |

*Additional Architecture Ablation Study.* We validate some architecture details by comparing the average testing error. The results are shown in Table 7. We found query-key normalization (Henry et al. 2020) helpful in stabilizing the training process, especially towards the end of the training phase. Notably, when the Muon optimizer is used, the training loss diverges without query-key normalization, demonstrating the effectiveness of this technique.

Table 8. **Comparing the inference speed and GPU memory usage for all models.** We measure the resource usage for different models and variants to generate 10 output steps for a single trajectory.

| Model | Param | Inference Speed (ms) | GPU Memory Usage (MB) |
|---|---|---|---|
| DeepONet | 3.5M | 2.7 | 190 |
| FNO | 0.6M | 1.7 | 372 |
| UNet | 5.6M | 1.3 | 135 |
| ViT | 155M | 5.4 | 2362 |
| MPP-B | 116M | 207.9 | 3087 |
| MPP-L | 407M | 438.3 | 10869 |
| DPOT-M | 122M | 95.7 | 1864 |
| DPOT-L | 523M | 194.3 | 7985 |
| Time-then-Space Variant | 206M | 175.5 | 3168 |
| Next-token-prediction Variant | 156M | 13165 | 2402 |
| BCAT (without KV cache) | 156M | 150.8 | 943 |
| BCAT (with KV cache) | 156M | 70.9 | 2499 |

This also suggests that the Muon optimizer has different optimization dynamics for query and key projection matrices. For this ablation experiment only, we report the result using the AdamW optimizer. Under similar parameter counts, we found SwiGLU activation function (Shazeer 2020) better than GeLU. Note that the parameter count is not exactly the same since the MLP hidden dimension is rounded to the nearest multiple of 64. For the attention mask, to prevent data leaking in next frame prediction, any submask of the block causal mask (2) can work. We compare the BCAT model and a variant using a standard causal mask to perform next frame prediction. We observe that the block causal mask leads to better performance, demonstrating that having more token interactions within the same frame is beneficial for the forward prediction task.

## 4.5 Resource Usage Comparison

In Table 8, we compare the inference speed and GPU memory usage for all models. To obtain these results, on a single NVIDIA H100 GPU, we use a batch size of 1 and measure the time required to generate 10 output steps for the shallow water equation. To ensure stability, we average run time over 50 repeats after some warmup. While smaller models (FNO, DeepONet, and UNet) have a faster run time, BCAT outperforms all models with comparable or larger sizes except ViT, which shares the same transformer layer without autoregressive rollout.

## 4.6 Experimental Summary and Model Novelty

We summarize the experimental results and highlight the significance of the BCAT model.

*Performance.* BCAT improves the average zero-shot relative $L^2$ error in the six evaluated two-dimensional fluid families. Compared to the 3.4-times-larger DPOT-L baseline, BCAT reduces the average error by 45%. It also performs well in low-data, in-distribution adaptation.

*Learning objective.* BCAT's next-frame prediction adapts causal sequence modeling to spatiotemporal fluid data. It permits parallel training loss computation and, with a key-value cache, reduces the stated inference time from 150.8 to 70.9 ms for ten SWE output steps on one H100 GPU. The large speed gap with the next-token variant is enabled by predicting an entire frame at each autoregressive step.

*Architecture improvements.* Architecture improvements in BCAT are helpful for the broader community. (1) In Section 4.4, we show that decreasing the patch size improves the performance of both BCAT and ViT. (2) We drop-in replaced PROSE-FD's optimizer by Muon, and the performance improved from 3.32% to 3.04%. (3) Compared to standard activation functions like ReLU and GeLU, GLU variants have better smoothness and similar convergence speed. BCAT is the first to test and demonstrate their effectiveness in scientific computing. (4) BCAT shows the benefit of causality in terms of efficiency and performance. While models like MPP and DPOT are designed to be fixed window update rules, causality can also be integrated to improve their performance, similar to the Time-then-space variant of BCAT.

*Engineering considerations.* BCAT retains a standard decoder-only transformer backbone, which permits implementations using `bfloat16`, `torch.compile`, and `FlexAttention` (J. Dong et al. 2024). On the stated hardware and configurations, BCAT trained in 42 hours, versus 120 hours for MPP-B and 80 hours for DPOT-M.

## 5 Conclusion

In this work, we introduced BCAT, a block-causal transformer for autoregressive prediction of two-dimensional fluid-dynamics solutions. Its next-frame objective permits spatial interactions within a frame while enforcing temporal causality across frames. Across six fluid families from PDEBench, PDEArena, and CFDBench, BCAT achieved the best average relative $L^2$ error in our zero-shot comparison while using fewer parameters than the largest baselines. The ablations show the benefit of next-frame prediction in the evaluated setting and of Muon relative to AdamW; the convergence and spectral analyses further characterize this optimizer effect.

The present evaluation focuses on two-dimensional fluid dynamics at a shared $128 \times 128$ resolution and dataset-specific temporal resolutions. Extending block-causal foundation models to additional PDE classes, higher-dimensional dynamics, new geometries, partial observations and variable temporal resolutions is a promising direction for future work.

## Acknowledgments

Yuxuan Liu and Hayden Schaeffer are supported in part by NSF DMS 2427558. The work was supported by the NVIDIA Academic Grant Program 2026.

## References

M. Abdin, J. Aneja, H. Awadalla, et al.. 2024. "Phi-3 technical report: A highly capable language model locally on your phone." *arXiv preprint arXiv:2404.14219*.

M. Abdin, J. Aneja, H. Behl, et al.. 2024. "Phi-4 technical report." *arXiv preprint arXiv:2412.08905*.

J. Achiam et al.. 2023. "Gpt-4 technical report." *arXiv preprint arXiv:2303.08774*.

G. Bertasius, H. Wang, and L. Torresani. 2021. "Is space-time attention all you need for video understanding?" In: *ICML* 3. Vol. 2, 4.

C. Bodnar et al.. 2024. "Aurora: A foundation model of the atmosphere." *arXiv preprint arXiv:2405.13063*.

R. Bommasani et al.. 2021. "On the opportunities and risks of foundation models." *arXiv preprint arXiv:2108.07258*.

T. Brooks et al.. 2024. *Video generation models as world simulators*. (2024).

T. Brown et al.. 2020. "Language models are few-shot learners." *Advances in neural information processing systems*, 33, 1877–1901.

Y. Cao, Y. Liu, L. Yang, R. Yu, H. Schaeffer, and S. Osher. 2024. "VICON: Vision In-Context Operator Networks for Multi-Physics Fluid Dynamics Prediction." *arXiv preprint arXiv:2411.16063*.

C. Chang, W.-Y. Wang, W.-C. Peng, and T.-F. Chen. 2024. "Llm4ts: Aligning pre-trained llms as data-efficient time-series forecasters." *arXiv preprint arXiv:2308.08469*.

M. Chen, A. Radford, R. Child, J. Wu, H. Jun, D. Luan, and I. Sutskever. 2020. "Generative pretraining from pixels." In: *International conference on machine learning*. PMLR, 1691–1703.

J. Dong, B. Feng, D. Guessous, Y. Liang, and H. He. 2024. "Flex Attention: A Programming Model for Generating Optimized Attention Kernels." *arXiv preprint arXiv:2412.05496*.

A. Dosovitskiy et al.. 2021a. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale." In: *International Conference on Learning Representations*.

A. Dosovitskiy et al.. 2021b. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale." *ICLR*.
A. Dubey et al.. 2024. "The llama 3 herd of models." *arXiv preprint arXiv:2407.21783*.
N. Gruver, M. Finzi, S. Qiu, and A. G. Wilson. 2024. "Large language models are zero-shot time series forecasters." *Advances in Neural Information Processing Systems*, 36.
J. K. Gupta and J. Brandstetter. 2022. "Towards Multi-spatiotemporal-scale Generalized PDE Modeling." *arXiv preprint arXiv:2209.15616*.
Z. Hao, C. Su, S. Liu, J. Berner, C. Ying, H. Su, A. Anandkumar, J. Song, and J. Zhu. 2024. "Dpot: Auto-regressive denoising operator transformer for large-scale pde pre-training." *arXiv preprint arXiv:2403.03542*.
A. Henry, P. R. Dachapally, S. Pawar, and Y. Chen. 2020. "Query-key normalization for transformers." *arXiv preprint arXiv:2010.04245*.
M. Herde, B. Raonić, T. Rohner, R. Käppeli, R. Molinaro, E. de Bézenac, and S. Mishra. 2024. "Poseidon: Efficient Foundation Models for PDEs." *arXiv preprint arXiv:2405.19101*.
E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, et al.. 2022. "Lora: Low-rank adaptation of large language models." *ICLR*, 1, 2, 3.
S. Hu et al.. 2024. "Minicpm: Unveiling the potential of small language models with scalable training strategies." *arXiv preprint arXiv:2404.06395*.
M. Jin et al.. 2023. "Time-llm: Time series forecasting by reprogramming large language models." *arXiv preprint arXiv:2310.01728*.
D. Jollie, J. Sun, Z. Zhang, and H. Schaeffer. 2024. "Time-series forecasting, knowledge distillation, and refinement within a multimodal pde foundation model." *arXiv preprint arXiv:2409.11609*.
K. Jordan, Y. Jin, V. Boza, J. You, F. Cesista, L. Newhouse, and J. Bernstein. 2024. *Muon: An optimizer for hidden layers in neural networks*. (2024). https://kellerjordan.github.io/posts/muon/.
Z. Li, K. Meidani, and A. B. Farimani. 2022. "Transformer for partial differential equations' operator learning." *arXiv preprint arXiv:2205.13671*.
Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. 2020. "Fourier neural operator for parametric partial differential equations." *arXiv preprint arXiv:2010.08895*.
J. Liu et al.. 2025. "Muon is scalable for llm training." *arXiv preprint arXiv:2502.16982*.
L. Liu, S. Yu, R. Wang, Z. Ma, and Y. Shen. 2024. "How can large language models understand spatial-temporal data?" *arXiv preprint arXiv:2401.14192*.
M. Liu, S. Ren, S. Ma, J. Jiao, Y. Chen, Z. Wang, and W. Song. 2021. "Gated transformer networks for multivariate time series classification." *arXiv preprint arXiv:2103.14438*.
Y. Liu, J. Sun, X. He, G. Pinney, Z. Zhang, and H. Schaeffer. 2024. "PROSE-FD: A Multimodal PDE Foundation Model for Learning Multiple Operators for Forecasting Fluid Dynamics." *arXiv preprint arXiv:2409.09811*.
Y. Liu, Z. Zhang, and H. Schaeffer. 2024. "PROSE: Predicting Multiple Operators and Symbolic Expressions using multimodal transformers." *Neural Networks*, 180, 106707.
I. Loshchilov and F. Hutter. N.d. "Decoupled Weight Decay Regularization." In: *International Conference on Learning Representations*.
L. Lu, P. Jin, and G. E. Karniadakis. 2019. "Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators." *arXiv preprint arXiv:1910.03193*.
Y. Luo, Y. Chen, and Z. Zhang. 2023. "Cfdbench: A comprehensive benchmark for machine learning methods in fluid dynamics." *arXiv preprint arXiv:2310.05963*.
M. McCabe et al.. 2023. "Multiple physics pretraining for physical surrogate models." *arXiv preprint arXiv:2310.02994*.
J. Pathak et al.. 2022. "Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators." *arXiv preprint arXiv:2202.11214*.
A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al.. 2018. "Improving language understanding by generative pre-training."
A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. 2019. "Language Models are Unsupervised Multitask Learners."
A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen. 2022. "Hierarchical text-conditional image generation with clip latents." *arXiv preprint arXiv:2204.06125*, 1, 2, 3.
A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever. 2021. "Zero-shot text-to-image generation." In: *International conference on machine learning*. Pmlr, 8821–8831.
R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. 2022. "High-resolution image synthesis with latent diffusion models." In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10684–10695.
O. Ronneberger, P. Fischer, and T. Brox. 2015. "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*. Springer, 234–241.
G. Schulz. 1933. "Iterative berechung der reziproken matrix." *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 13, 1, 57–59.
N. Shazeer. 2020. "Glu variants improve transformer." *arXiv preprint arXiv:2002.05202*.
S. Subramanian, P. Harrington, K. Keutzer, W. Bhimji, D. Morozov, M. W. Mahoney, and A. Gholami. 2024. "Towards foundation models for scientific machine learning: Characterizing scaling and transfer behavior." *Advances in Neural Information Processing Systems*, 36.

J. Sun, Y. Liu, Z. Zhang, and H. Schaeffer. 2024. "Towards a Foundation Model for Partial Differential Equation: Multi-Operator Learning and Extrapolation." *arXiv preprint arXiv:2404.12355*.
J. Sun, Z. Zhang, and H. Schaeffer. 2024. "LeMON: Learning to Learn Multi-Operator Networks." *arXiv preprint arXiv:2408.16168*.
M. Takamoto, T. Praditia, R. Leiteritz, D. MacKinlay, F. Alesiani, D. Pflüger, and M. Niepert. 2022. "Pdebench: An extensive benchmark for scientific machine learning." *Advances in Neural Information Processing Systems*, 35, 1596–1611.
M. Tan, M. A. Merrill, V. Gupta, T. Althoff, and T. Hartvigsen. 2024. "Are language models actually useful for time series forecasting?" In: *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
G. Team et al.. 2024. "Gemma 2: Improving open language models at a practical size." *arXiv preprint arXiv:2408.00118*.
K. Tian, Y. Jiang, Z. Yuan, B. Peng, and L. Wang. 2024. "Visual autoregressive modeling: Scalable image generation via next-scale prediction." *arXiv preprint arXiv:2404.02905*.
H. Touvron, T. Lavril, et al.. 2023. "Llama: Open and efficient foundation language models." *arXiv preprint arXiv:2302.13971*.
H. Touvron, L. Martin, et al.. 2023. "Llama 2: Open foundation and fine-tuned chat models." *arXiv preprint arXiv:2307.09288*.
A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. 2017. "Attention is all you need." In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*.
F. Wang, Y. Yu, G. Wei, W. Shao, Y. Zhou, A. Yuille, and C. Xie. 2025. "Scaling Laws in Patchification: An Image Is Worth 50,176 Tokens And More." *arXiv preprint arXiv:2502.03738*.
H. Wu, H. Luo, H. Wang, J. Wang, and M. Long. 2024. "Transolver: A fast transformer solver for pdes on general geometries." *arXiv preprint arXiv:2402.02366*.
A. Yang et al.. 2024. "Qwen2. 5 Technical Report." *arXiv preprint arXiv:2412.15115*.
L. Yang, S. Liu, T. Meng, and S. J. Osher. 2023. "In-context operator learning with data prompts for differential equation problems." *Proceedings of the National Academy of Sciences*, 120, 39, e2310142120.
L. Yang, T. Meng, S. Liu, and S. J. Osher. 2023. "Prompting in-context operator learning with sensor data, equations, and natural language." *arXiv preprint arXiv:2308.05061*.
L. Yang and S. J. Osher. 2024. "PDE Generalization of In-Context Operator Networks: A Study on 1D Scalar Nonlinear Conservation Laws." *arXiv preprint arXiv:2401.07364*.
B. Zhang and R. Sennrich. 2019. "Root mean square layer normalization." *Advances in Neural Information Processing Systems*, 32.
Z. Zhang, C. Moya, L. Lu, G. Lin, and H. Schaeffer. 2024. "DeepONet as a Multi-Operator Extrapolation Model: Distributed Pretraining with Physics-Informed Fine-Tuning." *arXiv preprint arXiv:2411.07239*.
Z. Zhou and R. Yu. 2024. "Can LLMs Understand Time Series Anomalies?" *arXiv preprint arXiv:2410.05440*.

## A Dataset Details

The data was obtained from the PDEBench (Takamoto et al. 2022), PDEArena (Gupta and Brandstetter 2022), and CFDBench (Luo et al. 2023) datasets. Unless otherwise specified, the space resolution is $128 \times 128$.

### A.1 PDEBench

*Shallow Water Equation.* The quantity of interest is the water depth $h(\boldsymbol{x}, t)$ on domain $[-2.5, 2.5]^2 \times [0, 1]$ with Neumann boundary condition. The temporal resolution is 101. The equations are:

$$\partial_t h + \nabla h \boldsymbol{u} = 0, \tag{4}$$

$$\partial_t h \boldsymbol{u} + \nabla \left( h\, \boldsymbol{u} \cdot \boldsymbol{u} + \frac{1}{2} g_r h^2 \right) = -g_r h \nabla b. \tag{5}$$

*Incompressible Navier-Stokes Equation.* The quantities of interest are the velocities $\boldsymbol{u}(\boldsymbol{x}, t)$ and particle density $c(\boldsymbol{x}, t)$ on domain $[0, 1]^2 \times [0, 5]$ with Dirichlet boundary condition. The temporal resolution is 1000. The equations are:

$$\rho(\partial_t \boldsymbol{u} + \boldsymbol{u} \cdot \nabla \boldsymbol{u}) = -\nabla p + \mu \Delta \boldsymbol{u} + \mathbf{F}, \tag{6}$$

$$\nabla \cdot \boldsymbol{u} = 0, \tag{7}$$

$$\partial_t c + \nabla \cdot (c\boldsymbol{u}) = 0. \tag{8}$$

The forcing term $\mathbf{F}$ is randomly sampled.

*Compressible Navier-Stokes Equation.* The quantities of interest are the velocities $\boldsymbol{u}(\boldsymbol{x}, t)$, pressure $p(\boldsymbol{x}, t)$, and density $\rho(\boldsymbol{x}, t)$ on domain $[0, 1]^2 \times [0, 1]$ with periodic boundary conditions. The temporal resolution is 21. For equations with low viscosities, the dataset is provided on a finer $512 \times 512$ space grid, which is downsampled to $128 \times 128$ for consistency (through average pooling). The equations are:

$$\partial_t \rho + \nabla \cdot (\rho \boldsymbol{u}) = 0, \tag{9}$$

$$\rho(\partial_t \boldsymbol{u} + \boldsymbol{u} \cdot \nabla \boldsymbol{u}) = -\nabla p + \eta \Delta \boldsymbol{u} + (\zeta + \eta/3)\nabla(\nabla \cdot \boldsymbol{u}), \tag{10}$$

$$\partial_t \left( \varepsilon + \frac{\rho u^2}{2} \right) = -\nabla \cdot \left( \left( \varepsilon + p + \frac{\rho u^2}{2} \right) \boldsymbol{u} - \boldsymbol{u} \cdot \sigma' \right). \tag{11}$$

## A.2 PDEArena

*Incompressible Navier-Stokes Equation.* The quantities of interest are the velocities $\boldsymbol{u}(\boldsymbol{x}, t)$ and particle density $c(\boldsymbol{x}, t)$ on domain $[0, 32]^2 \times [18, 102]$ with Dirichlet boundary conditions for velocity and Neumann boundary condition for particle field. The temporal resolution is 14. The equations are:

$$\rho(\partial_t \boldsymbol{u} + \boldsymbol{u} \cdot \nabla \boldsymbol{u}) = -\nabla p + \mu \Delta \boldsymbol{u} + \mathbf{F}, \tag{12}$$

$$\nabla \cdot \boldsymbol{u} = 0, \tag{13}$$

$$\partial_t c + \nabla \cdot (c\boldsymbol{u}) = 0. \tag{14}$$

The forcing term $\mathbf{F}$ takes the form $\mathbf{F} = (0, f)$ with $f = 0.5$.

*Incompressible Navier-Stokes Equation (Conditioned).* The quantities of interest are the velocities $\boldsymbol{u}(\boldsymbol{x}, t)$ and particle density $c(\boldsymbol{x}, t)$ on domain $[0, 32]^2 \times [18, 102]$ with Dirichlet boundary conditions for velocity and Neumann boundary condition for particle field. The temporal resolution is 56. The equations are:

$$\rho(\partial_t \boldsymbol{u} + \boldsymbol{u} \cdot \nabla \boldsymbol{u}) = -\nabla p + \mu \Delta \boldsymbol{u} + \mathbf{F}, \tag{15}$$

$$\nabla \cdot \boldsymbol{u} = 0, \tag{16}$$

$$\partial_t c + \nabla \cdot (c\boldsymbol{u}) = 0. \tag{17}$$

The forcing term $\mathbf{F}$ takes the form $\mathbf{F} = (0, f)$ where $f$ is uniformly sampled in $[0.2, 0.5]$.

## A.3 CFDBench

*Incompressible Navier-Stokes Equation.* The quantities of interest are the velocities $\boldsymbol{u}(\boldsymbol{x}, t)$ and pressure $p(\boldsymbol{x}, t)$. This dataset contains irregular geometries with Dirichlet boundary conditions. The raw space resolution is $64 \times 64$ which is upsampled to $128 \times 128$ via interpolation. The equations are:

$$\rho(\partial_t \boldsymbol{u} + \boldsymbol{u} \cdot \nabla \boldsymbol{u}) = -\nabla p + \mu \Delta \boldsymbol{u}, \tag{18}$$

$$\nabla \cdot \boldsymbol{u} = 0. \tag{19}$$

# B Experiment Details

## B.1 Training

We perform data normalization during the training process. Given the input sequence of data $\{\boldsymbol{u}(\cdot, t_i) \mid 0 \le i < T_0\}$, we compute the mean and standard deviation of each input trajectory, which are used to normalize both the input and ground truth sequence. The loss function is the standard mean squared error in the normalized space. The models are trained using a combination of Muon optimizer (Jordan et al. 2024; J. Liu et al. 2025) and AdamW optimizer (Loshchilov and Hutter n.d.) with a global batch size of 128 for 40 epochs where each epoch is 4,000 steps. The Muon optimizer is applied to all transformer matrix parameters, while AdamW optimizer is applied

to the rest of the parameters (scalar, embedding, initial and final projection layer). The warmup-stable-decay learning rate scheduler (S. Hu et al. 2024) is used with 10% warmup and 50% decay. We use learning rate $10^{-3}$ and weight decay $10^{-4}$. On four NVIDIA H100 GPUs, the training takes about 42 hours.

## B.2 Model Hyperparameters

We summarize the model hyperparameters in Table 9. An illustration of a transformer layer used in BCAT model is shown in Figure 4.

Table 9. **Model hyperparameters.** FFN means feedforward network.

| | | | |
|---|---|---|---|
| Hidden dimension - attention | 1024 | Hidden dimension - FFN | 2752 |
| Number of attention heads | 8 | Number of layers | 12 |
| Activation | SwiGLU | Dropout | 0 |
| Patch Size | 8 | | |

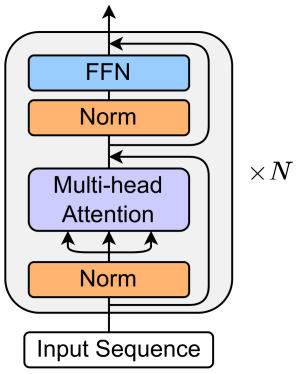



Fig. 4. Transformer Layers used in BCAT model.

## B.3 Baselines and Comparisons

In this section, we include more details about the compared models.

*DeepONet (L. Lu et al. 2019).* We employ the unstacked DeepONet architecture, consisting of a single trunk network and a single branch network. Initially, the input data is divided into $8 \times 8$ patches, with each patch being embedded into a 128-dimensional vector. These vectors are then passed through the branch network, producing an output with a basis dimension of $p = 50$. Simultaneously, the query point is processed through the trunk network, which also outputs a vector with the same dimension, $p$. The output solution at the query point is obtained by taking the inner product of the outputs from the two networks.

*FNO (Z. Li, Kovachki, et al. 2020).* We use 4 layers of standard 3d FNO to process the input data. The number of modes to keep in each dimension is set to 8, and the number of hidden channels is set to 16. The 3d FNO model directly maps 10 input steps to 10 output steps.

*UNet (Ronneberger et al. 2015).* We use 8 layers of 3d UNet with GeLU activation and 32 hidden dimensions. The 3d UNet model directly maps 10 input steps to 10 output steps (Padding is added to the input to make it a power of 2, which is needed for the shrinkage in dimension).

*ViT (Dosovitskiy et al. 2021b).* For ViT, we use 12 transformer layers similar to the one shown in Figure 4. The patch size is set to be 16, the hidden dimension for attention is 1024, the hidden dimension for the feedforward network is 4096 with GeLU activation, and the number of heads is 8. The ViT model learns a fixed window update rule, directly mapping 10 input steps to 10 output steps.

*PROSE-FD (Y. Liu, Sun, et al. 2024).* We directly use the results for PROSE-FD in (Y. Liu, Sun, et al. 2024), since the same dataset and same context window is used.

*MPP (McCabe et al. 2023).* In (McCabe et al. 2023), MPP is trained on a different training dataset and different evaluation metrics are used. For a fair comparison, we retrained MPP-B and MPP-L on our training dataset keeping the same model configurations and optimizer hyperparameters. We evaluate MPP-B and MPP-L using the same testing setup and metric, where rollout is used to obtain all 10 output steps.

*DPOT (Hao et al. 2024).* In (Hao et al. 2024), DPOT is trained on a different training dataset and different evaluation metrics are used. For a fair comparison, we retrained DPOT-M and DPOT-L on our training dataset keeping the same model configurations and optimizer hyperparameters. We evaluate DPOT-M and DPOT-L using the same testing setup and metric, where rollout is used to obtain all 10 output steps.

### B.4 Necessity of Retraining MPP and DPOT for Fairness

*Different Context Window.* MPP is trained using a context window of 16, while DPOT is trained using a context window of 10. As these models learn a fixed window update rule, it is neither possible nor fair to directly use MPP with a context window of 10. This also shows one benefit of BCAT's next frame prediction, i.e., flexible context window. Even though BCAT is trained with 10 input timestamps, it can directly handle longer input sequence lengths such as 16, while DPOT has to discard the first 6 timestamps.

*Different Dataset.* MPP and DPOT are trained on different datasets. MPP was not trained on PDEArena and CFDBench dataset, and DPOT was not trained on PDEBench incompressible NS dataset. Additionally, DPOT did not split a validation dataset, making its training dataset larger than MPP.

*Different Evaluation Metric.* While both MPP and DPOT report their errors using the term relative $L^2$ errors, the quantitative definition may differ, e.g., number of output steps, averaging in space versus space-time, etc. Thus we do not directly compare to the reported numbers in MPP and DPOT.

To address this, we retrain MPP and DPOT, using the same context window of 10, the same collection of training dataset, and evaluate them using the same metric. We utilized their code, model configuration, and hyperparameters, and only altered the necessary parts mentioned above. In Table 10, we compare the performance of DPOT-M and DPOT-L when we directly evaluate the pretrained checkpoints versus our retrained model. We observe that our retrained version has similar or better performance on all datasets compared to their released pretrained checkpoints.

### B.5 Remarks on Single-Task Models

In Table 1, we compare the zero-shot performance of training a single model for all families of datasets simultaneously. The main focus is to show that these single-task models are not capable of handling more complicated tasks and datasets we are considering using a single pretrained model. For completeness, we also train these single-task models separately on each dataset, and the results are shown in Table 11. We observe that, the performance does improve for almost all cases (as expected); however, the error's overall order does not change. All single-task models have difficulty predicting the PDEArena dataset, suggesting larger models and better learning algorithms are needed.

Table 10. **Evaluating Pretrained Checkpoints vs. Retrained Model.** * means directly evaluating released pretrained checkpoints. Numbers in () mean that the model is not pretrained on this dataset. The numbers reported are relative $L^2$ errors (%). The averages are taken with respect to the 6 distinct families listed in the columns of the table.

| Model | Param | PDEBench | | | PDEArena | | CFDBench | Average |
|---|---|---|---|---|---|---|---|---|
| | | SWE | CNS* | INS | NS | NS-cond | - | |
| DPOT-M | 122M | 0.54 | 1.01 | 5.20 | 4.92 | 8.55 | 0.64 | 3.47 |
| DPOT-M* | 122M | 0.32 | 2.82 | (81.57) | 4.49 | 8.40 | 3.20 | - |
| DPOT-L | 523M | 0.15 | 0.89 | 4.08 | 2.21 | 5.29 | 0.34 | 2.16 |
| DPOT-L* | 523M | 0.28 | 2.67 | (66.64) | 4.34 | 7.23 | 3.12 | - |

Table 11. **Single-tasked models trained on all data simultaneously v.s. on each class separately.** * means the results are obtained by training separate models on each dataset. The numbers reported are relative $L^2$ errors (%).

| Model | Param | PDEBench | | | PDEArena | | CFDBench |
|---|---|---|---|---|---|---|---|
| | | SWE | CNS* | INS | NS | NS-cond | - |
| DeepONet | 3.5M | 3.55 | 7.41 | 64.61 | 35.33 | 51.85 | 12.50 |
| DeepONet* | 6× 3.5M | 2.86 | 7.31 | 57.61 | 31.00 | 48.34 | 14.08 |
| FNO | 0.6M | 3.71 | 6.31 | 36.84 | 38.67 | 55.63 | 8.52 |
| FNO* | 6× 0.6M | 1.14 | 5.58 | 21.67 | 26.39 | 44.89 | 3.21 |
| UNet | 5.6M | 0.33 | 3.19 | 3.43 | 12.56 | 16.82 | 0.76 |
| UNet* | 6× 5.6M | 0.19 | 3.32 | 3.15 | 12.96 | 14.02 | 0.43 |

## C Additional Results

Unless stated otherwise, the results in this section use the same held-out test trajectories, autoregressive evaluation protocol, and relative $L^2$ metric as the main comparison in Section 4.2 (ten output steps, except four for PDEArena NS).

### C.1 Multi-Seed Variability

Table 12 reports the mean and sample standard deviation across five independently seeded BCAT training runs. All runs use the same model, training data, and hyperparameters as the main BCAT experiment; only the random seed changes. The table characterizes the variability of the final test error across the six fluid families. We observe that the standard deviation is small relative to the mean, indicating that BCAT's performance is robust to random initialization.

Table 12. **BCAT five seed runs.** The numbers reported are relative $L^2$ errors (%).

| PDEBench | | | PDEArena | | CFDBench |
|---|---|---|---|---|---|
| SWE | CNS* | INS | NS | NS-cond | - |
| 0.12 ± 0.02 | 0.40 ± 0.01 | 1.27 ± 0.07 | 1.65 ± 0.05 | 3.22 ± 0.12 | 0.44 ± 0.06 |

Table 13. **Evaluate BCAT with Noisy Input.** The numbers reported are relative $L^2$ errors (%). The averages are taken with respect to the 6 distinct families listed in the columns of the table.

| Noise Level | PDEBench | | | PDEArena | | CFDBench | Average |
|---|---|---|---|---|---|---|---|
| | SWE | CNS* | INS | NS | NS-cond | - | |
| 0 (no noise) | 0.10 | 0.39 | 1.34 | 1.59 | 3.13 | 0.52 | 1.18 |
| 0.01 | 0.17 | 0.52 | 1.43 | 1.70 | 3.17 | 3.49 | 1.75 |
| 0.05 | 1.02 | 1.50 | 2.84 | 3.24 | 5.29 | 6.52 | 3.40 |
| 0.10 | 2.90 | 2.82 | 5.19 | 5.61 | 12.77 | 11.14 | 6.74 |

## C.2 Input Noise

In Table 13, we report BCAT's robustness to multiplicative Gaussian noise applied to the physical-field input sequence before the model's standard normalization. For each input value $x$, we use $\tilde{x} = x(1 + \sigma\epsilon)$, where $\epsilon \sim \mathcal{N}(0, 1)$ is sampled independently over input times, spatial locations, and channels, and $\sigma$ is the noise level in the table. Only the observed input context is corrupted; evaluation uses the clean ground-truth future trajectory. This is a practical stress test for scientific ML settings where observations may be noisy or partially corrupted.

BCAT remains accurate under mild perturbations: at $\sigma = 0.01$, the average relative $L^2$ error is 1.75%, compared with 1.18% on clean inputs. Errors increase more substantially at higher noise levels, with the largest sensitivity on the CFDBench geometries.

## C.3 Physics Diagnostics

We report two physics diagnostics on the predicted velocity $(u, v)$ in physical space. Divergence MAE is the spatiotemporal mean of $|\nabla \cdot \hat{\mathbf{u}}|$, computed using a second-order interior finite-difference stencil; we also list the same quantity on the numerical ground truth as a solver floor. Kinetic-energy spectrum error is the relative $L^2$ distance between the radially averaged 2D kinetic-energy spectra of the prediction and the ground truth, averaged over the prediction horizon. We report divergence for the applicable incompressible, obstacle-free datasets and kinetic-energy spectra for velocity-field datasets.

For incompressible flows without immersed obstacles, we compute the interior finite-difference divergence of the predicted velocity and average its magnitude over space and time:

$$\mathrm{DivMAE} = \frac{1}{T(H-2)(W-2)} \sum_{t=1}^{T} \sum_{i=2}^{H-1} \sum_{j=2}^{W-1} \left| \frac{\hat{u}_{t,i+1,j} - \hat{u}_{t,i-1,j}}{2\Delta x} + \frac{\hat{v}_{t,i,j+1} - \hat{v}_{t,i,j-1}}{2\Delta y} \right|. \tag{20}$$

For the kinetic-energy diagnostic, let $\mathcal{F}$ denote the 2D DFT and let $\mathbf{q}$ index its frequency grid. We radially average the spectral kinetic energy into integer wavenumber bins:

$$\mathcal{E}_t(k) = \underset{\|\mathbf{q}\| \in [k, k+1)}{\mathrm{RadialMean}} \left[ \frac{1}{2} \left( |\mathcal{F}[u_t](\mathbf{q})|^2 + |\mathcal{F}[v_t](\mathbf{q})|^2 \right) \right]. \tag{21}$$

We then compare the predicted and ground-truth spectra at every forecast time:

$$\mathrm{SpecL2} = \frac{1}{T} \sum_{t=1}^{T} \frac{\left\| \mathcal{E}_t^{\mathrm{pred}} - \mathcal{E}_t^{\mathrm{gt}} \right\|_2}{\left\| \mathcal{E}_t^{\mathrm{gt}} \right\|_2 + 10^{-7}}. \tag{22}$$

On PDEBench INS, all models exceed the numerical ground-truth divergence floor ($8.15 \times 10^{-3}$), but BCAT has the lowest residual among the three methods (0.1336 versus 0.1655 for DPOT-L and 0.3050 for MPP-L). On both

Table 14. **Divergence MAE on incompressible flow benchmarks.** Values are averaged over the predicted horizon; the ground-truth residual is included as a solver floor.

| Model | PDEBench INS | PDEArena NS | PDEArena NS-cond |
|---|---|---|---|
| Ground Truth | 0.0082 | 0.0153 | 0.0303 |
| BCAT | 0.1336 | 0.0157 | 0.0310 |
| DPOT-L | 0.1655 | 0.0150 | 0.0299 |
| MPP-L | 0.3050 | 0.0149 | 0.0320 |

Table 15. **Kinetic-energy spectrum relative $L^2$ error (%).** The numbers reported are relative $L^2$ distances between the radially averaged 2D kinetic-energy spectra of the prediction and the ground truth, averaged over the predicted time horizon.

| Model | PDEBench | | PDEArena | | CFDBench |
|---|---|---|---|---|---|
| | CNS* | INS | NS | NS-cond | - |
| BCAT | 1.67 | 1.65 | 1.06 | 1.52 | 0.07 |
| DPOT-L | 1.51 | 1.26 | 0.95 | 1.56 | 0.21 |
| MPP-L | 3.10 | 2.07 | 3.15 | 3.60 | 0.11 |

PDEArena variants, all three methods are essentially at the solver floor, so divergence is not a useful ranking metric there. The KE-spectrum error is more discriminative: BCAT and DPOT-L are comparable, while MPP-L is consistently worse, especially on PDEArena NS-cond (1.52% vs. 3.60%) and PDEBench CNS (1.67% vs. 3.10%). Together with the main relative $L^2$ results, this suggests that BCAT's field accuracy is not obtained by sacrificing incompressibility or the kinetic-energy spectrum relative to the strongest baseline.

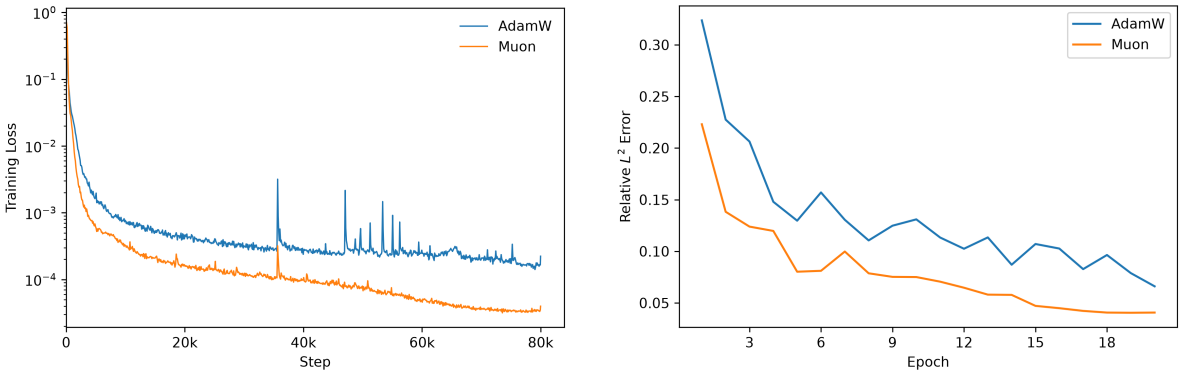
### C.4 Muon Optimization Diagnostics

To examine optimizer behavior in a controlled setting, we train the same BCAT architecture for 20 epochs on PDEArena NS-cond using AdamW or the mixed Muon-AdamW setup. The data split, batch size, and remaining training configuration follow the main experiment. Figure 5 shows lower training loss and validation error for Muon.

Figure 6 compares a speed-spectrum diagnostic for the AdamW and Muon predictions. Let $s = \sqrt{u^2 + v^2}$ be speed on a $H \times W$ grid. We take the orthonormal 2D DFT of $s$, shift the DC component to the center, form the power $P = |\hat{s}|^2$, and average this power within integer radial-wavenumber rings. After averaging over samples and predicted times, we plot for $k = 1, \ldots, \min(H, W)/2 - 1$ the relative error $|P_{\text{pred}}(k) - P_{\text{gt}}(k)|/(P_{\text{gt}}(k) + 10^{-10})$. This speed-based diagnostic is distinct from the kinetic-energy spectrum metric in Table 15.

## D Visualizations

In this section, we provide visualizations of the model outputs. Figure 7 and Figure 8 contain output visualizations from the BCAT model. In Figure 9 and Figure 10, we compare different model outputs on the same example. While patch effects and patch boundary discontinuities do exist, e.g. Figure 8a and Figure 10, which is common for patch-based methods, they are not the main source of error, e.g. high frequency and high contrast regions. To validate this, we compute the average error of interior points (masking out patch boundaries), which is 1.17% versus the total average error of 1.18%.




(a) Training loss (b) Validation error

Fig. 5. **Controlled optimizer convergence on PDEArena NS-cond.** BCAT is trained for 20 epochs with AdamW or mixed Muon-AdamW optimization.

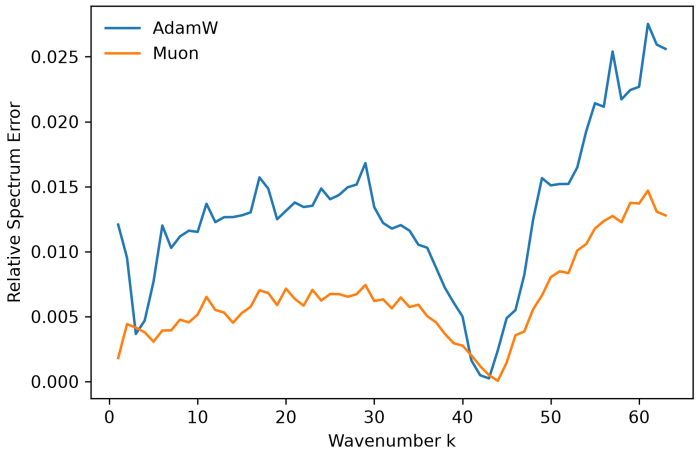



Fig. 6. **Relative speed-spectrum error for AdamW and Muon on PDEArena NS-cond.** Muon has lower error across most resolved wavenumbers, including the high-wavenumber range.

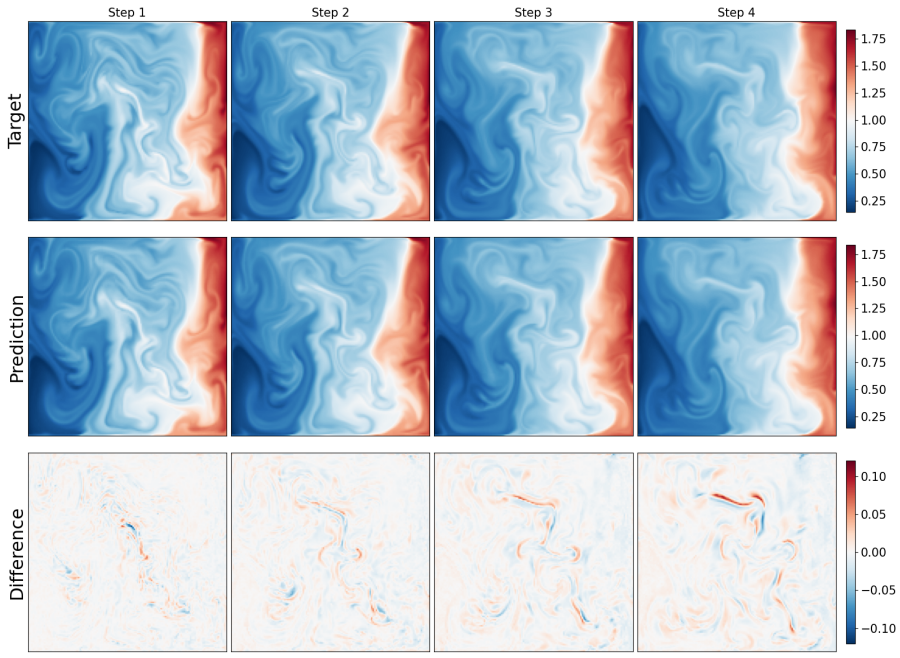
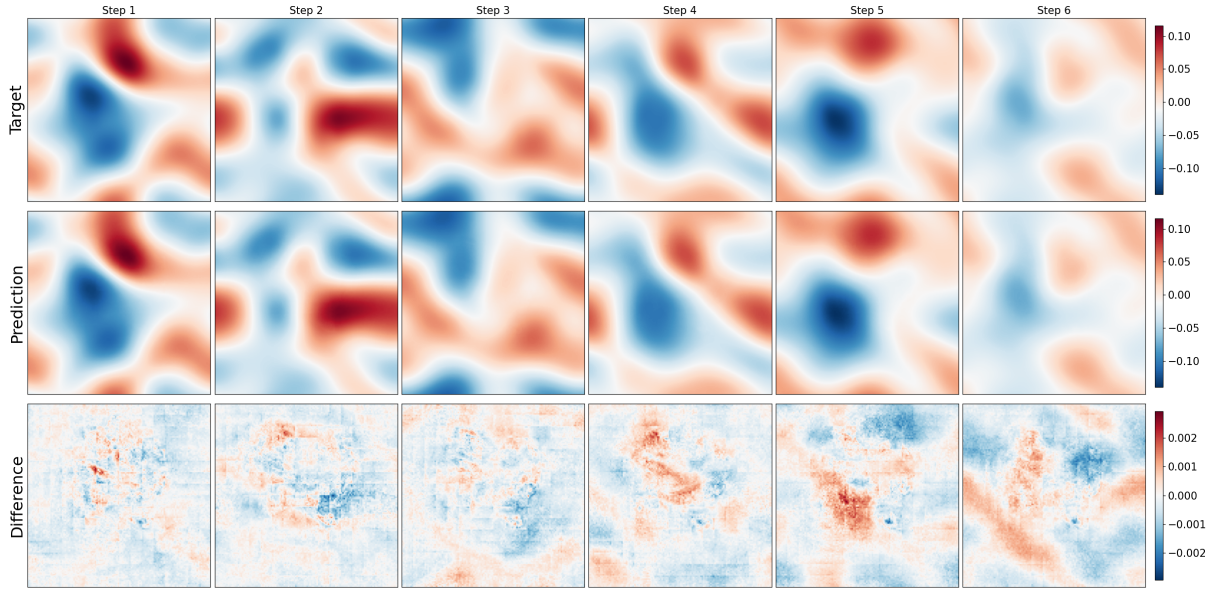
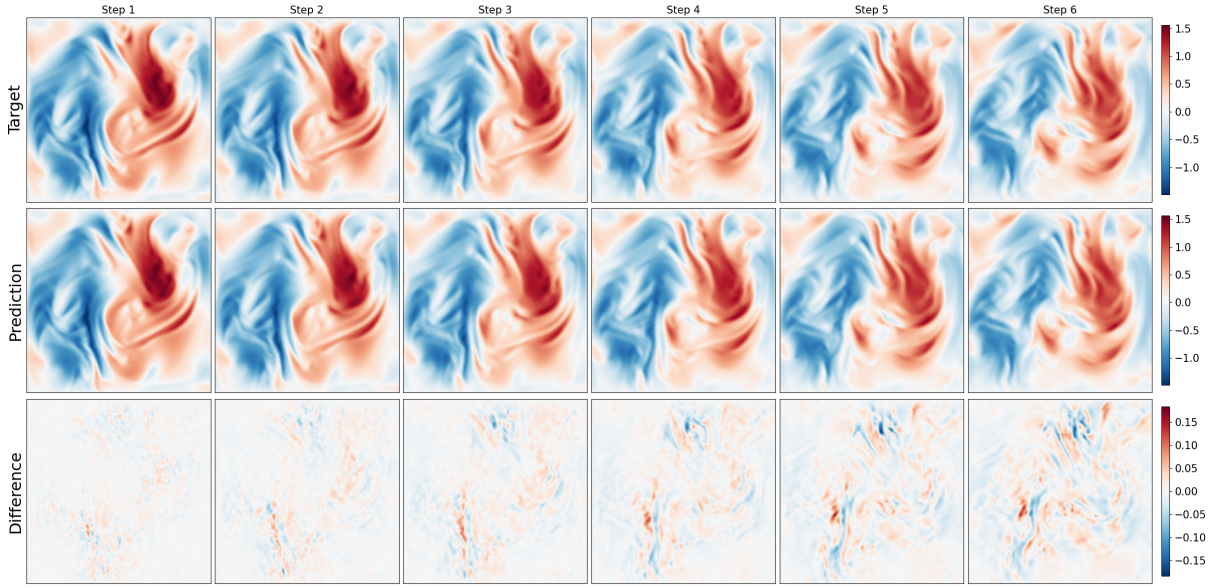



Fig. 7. **Example outputs from the BCAT model.** 4 output steps for PDEArena Navier-Stokes dataset. The channel plotted is the particle density in equation (12). Each column represents a different timestamp. For this trajectory, the relative $L^2$ error is 1.93%.




(a) 6 output steps for PDEBench Compressible Navier-Stokes dataset. The channel plotted is the $x$-velocity in equation (9). Each column represents a different timestamp. For this trajectory, the relative $L^2$ error is 0.03%.




(b) 6 output steps for PDEArena Navier-Stokes (conditioned) dataset. The channel plotted is the $x$-velocity in equation (12). Each column represents a different timestamp. For this trajectory, the relative $L^2$ error is 4.08%.

Fig. 8. **Two example outputs from the BCAT model.**

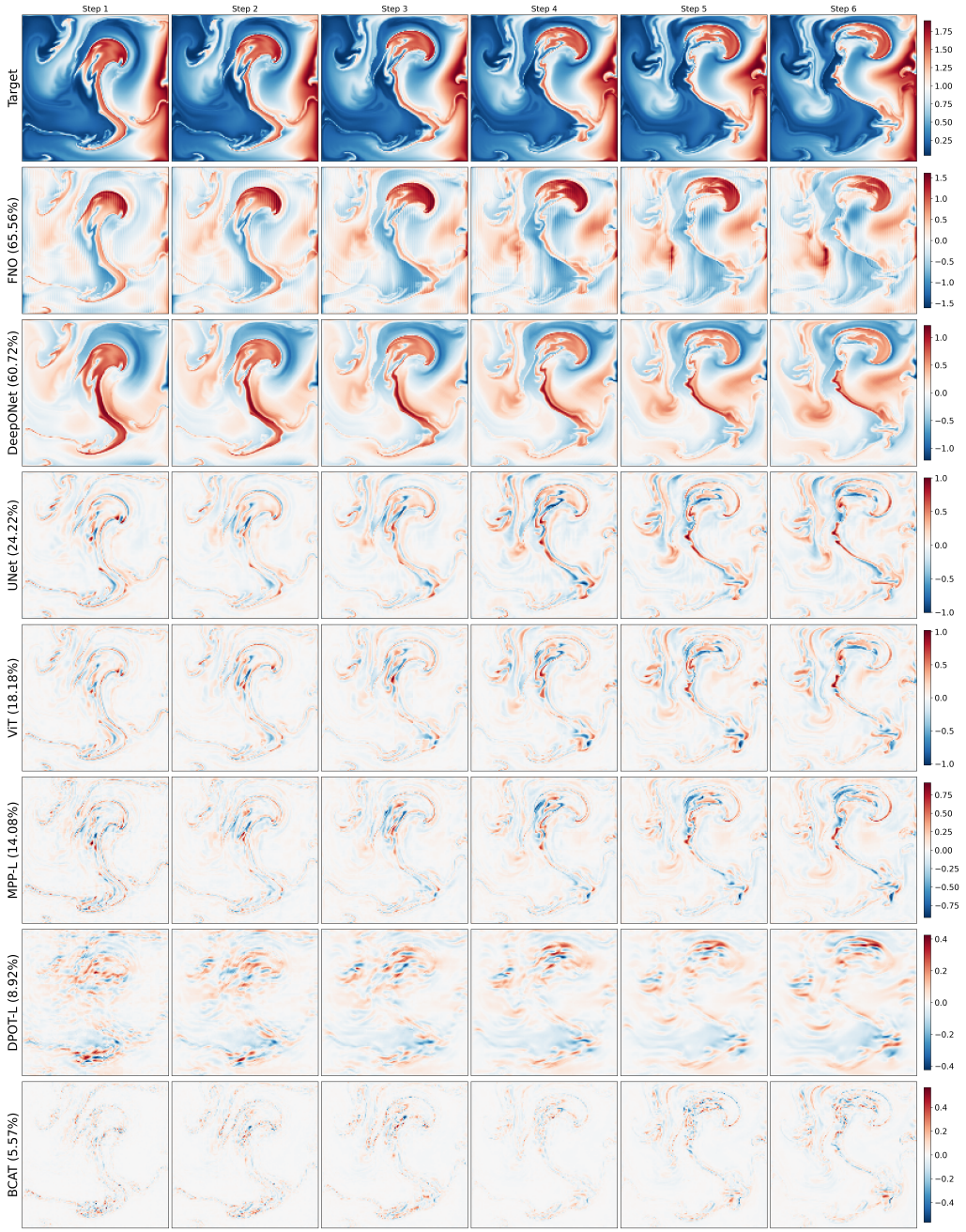



Fig. 9. **Comparing outputs from different models.** The target (first row) is the first 6 output steps from the PDEArena Navier-Stokes (conditioned) dataset (particle density channel). For each model (each row), we display the difference between the target and model output. Relative $L^2$ errors for the full trajectories are listed after the model names.

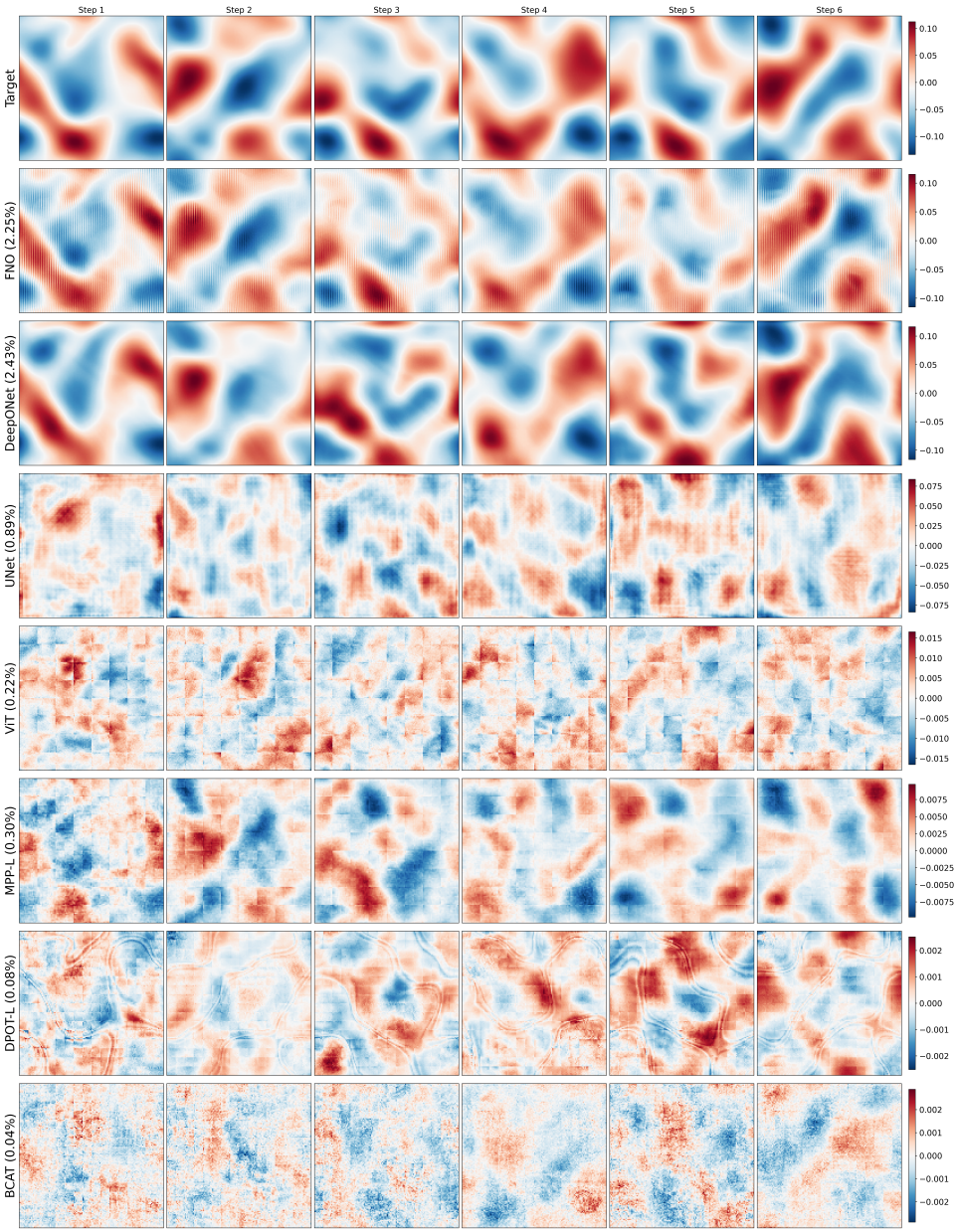



Fig. 10. **Comparing outputs from different models.** The target is the first 6 output steps from the PDEBench Compressible Navier-Stokes dataset ($y$-velocity channel). For each model (each row), we display the difference between target and model output. Relative $L^2$ errors for the full trajectory are listed after the model names.

Received 27 February 2026; accepted 12 October 2026