Masked Autoencoders are PDE Learners

Anonymous authors
Paper under double-blind review

Abstract

Neural solvers for partial differential equations (PDEs) have great potential to generate fast and accurate physics solutions, yet their practicality is currently limited by their generalizability. PDEs evolve over broad scales and exhibit diverse behaviors; predicting these phenomena will require learning representations across a wide variety of inputs which may encompass different coefficients, boundary conditions, resolutions, or even equations. As a step towards generalizable PDE modeling, we adapt masked pretraining for physics problems. Through self-supervised learning across PDEs, masked autoencoders can consolidate heterogeneous physics to learn rich latent representations. We show that learned representations can generalize to unseen equations or parameters and are semantically meaningful by performing latent PDE arithmetic. Furthermore, we demonstrate that masked pretraining can improve PDE coefficient regression and the classification of PDE features. Lastly, conditioning neural solvers on learned latent representations can improve time-stepping and super-resolution performance across a variety of coefficients, discretizations, or boundary conditions, as well as on unseen PDEs. We hope that masked pretraining can emerge as a unifying method across large, unlabeled, and heterogeneous datasets to learn latent physics at scale.

1 Introduction

The physical world is incredibly complex; physical phenomena can be extremely diverse and span wide spatiotemporal scales—from neuron excitations to turbulent flow to even global climate. Importantly, many of these phenomena can be mathematically modeled with time-dependent partial differential equations (PDEs)(FitzHugh, 1961; Nagumo et al., 1962; Lorenz, 1963). These PDEs are generally analytically intractable and require the use of numerical solvers to obtain approximate solutions. For complex physics, these solutions can often be slow to obtain; furthermore, different PDEs often require a careful design of tailored solvers.

Advances in deep learning in the past decade have led to the design of a new class of solvers for PDEs. These neural solvers can be extremely fast and display resolution invariance; however, neural networks introduce training difficulties and a lack of theoretical guarantees. Many important advances have been made to address these challenges, with SOTA models outperforming numerical solvers on well-studied PDEs under certain setups, proposing error bounds, and being extended to solve real-world problems. (Raissi et al., 2019; Lu et al., 2019; Li et al., 2020; Cao, 2021; Brandstetter et al., 2022; Li et al., 2023; Kovachki et al., 2021; Li et al., 2024).

A current frontier in neural PDE solvers lies in generalizing solvers to different parameters, conditions, or equations, thereby avoiding the need to collect new data and retrain networks when given unseen PDE dynamics. Prior work in this space has explored many methods to achieve this, from directly conditioning on PDE coefficients (Takamoto et al., 2023; Lorsung et al., 2024; Shen et al., 2024) to pretraining foundation models across various equations (Subramanian et al., 2023; McCabe et al., 2023; Hao et al., 2024). Despite these advances, generalizable neural solvers remain a significant challenge. PDEs can be incredibly diverse, and neural solvers must adapt to different coefficients, geometries, discretizations, or boundary conditions.

As a step towards addressing these challenges, we propose adapting masked pretraining methods to the PDE domain. Specifically, we demonstrate that self-supervised masked pretraining can learn latent structure that can express different coefficients, discretizations, boundary conditions or PDEs under a common representation.

Furthermore, we show that masked autoencoders (MAEs) can learn highly structured latent spaces without the need for labels, as well as propose latent physics arithmetic in this space. MAE models can be used to improve downstream tasks such as predicting PDE features or guiding neural solvers in time-stepping or super-resolution through providing meaningful context. Our contributions suggest the possibility to transfer the scalability and flexibility of masked modeling from language and vision domains to physics—creating rich, unified representations of diverse physics through self-supervised learning.

2 Related Work

Neural PDE Solvers The field of neural PDE solvers has grown rapidly and has shown great advances in both the accuracy of solutions and the ability to adapt to PDE parameters. Infinite-dimensional neural operators (Li et al., 2020; Kovachki et al., 2023; Lu et al., 2019) have shown impressive accuracy in solving time-dependent PDEs by learning the mappings between initial conditions and solutions. However, these methods alone have shown brittleness with respect to changing PDE coefficients or boundary conditions (Gupta and Brandstetter, 2022; Lu et al., 2021), prompting recent work to allow neural solvers to adapt to different PDE conditions.

A variety of approaches have considered adding PDE dynamics information to neural solvers. Common neural solvers can support conditional prediction through architecture choices (Gupta and Brandstetter, 2022), and novel architectures can be designed to explicitly operate with PDE parameter knowledge (Brandstetter et al., 2022). Beyond directly conditioning on PDE dynamics, a class of neural PDE solvers has proposed the addition of an encoder or adaptive network to inform a forecaster network of different PDE coefficients (Wang et al., 2021; Kirchmeyer et al.; Takamoto et al., 2023; Lorsung et al., 2024). At an even higher level, meta-learning approaches have been adapted to PDE learning to maximize shared learning across different physics (Yin et al., 2021; Zhang et al., 2023).

Pretraining for PDEs As an effort to work towards more generalizable PDE neural solvers, recent work has followed the success of pretraining and foundational models in the broader deep learning community. Based on contrastive pretraining methods in computer vision problems, (Chen et al., 2020; Schroff et al., 2015; Zbontar et al., 2021; Bardes et al., 2022), contrastive PDE methods aim to leverage equation coefficients (Lorsung and Farimani, 2024), physical invariances (Zhang et al., 2023), or Lie point symmetries (Mialon et al., 2023; Brandstetter et al., 2022) to define differences in PDE dynamics that can be organized in a latent space. Another approach in PDE pretraining follows observed in-context learning and emergent behavior in LLMs (Wei et al., 2022; Brown et al., 2020; Radford et al.) to design neural PDE solvers that are capable of following prompted PDE examples to forecast unseen dynamics (Yang et al., 2023; Chen et al., 2024).

A more straightforward pretraining method focuses on directly training neural solvers to transfer to new PDE dynamics (Goswami et al., 2022; Chakraborty et al., 2022; Wang et al., 2022). This approach has also been scaled by training neural solvers with large and diverse training sets to characterize its transfer behavior (Subramanian et al., 2023), as well as shown to be generally more effective over other pretraining strategies (Zhou et al., 2024). As a step toward large-scale modeling, more principled training approaches have been proposed to learn PDE dynamics across diverse physics at scale. Recent work has proposed a combinatorial neural operator that learns different dynamics as separate modules (Tripura and Chakraborty, 2023), embedding separate PDEs to a common space to do multi-physics prediction (McCabe et al., 2023), incorporating denoising with scalable transformer architectures while training across diverse PDE datasets (Hao et al., 2024), and using a unified PDE embedding to align LLMs across PDE families (Shen et al., 2024).

Masked Pretraining Masked reconstruction is a popular technique popularized by the language processing (Devlin et al., 2018) and vision (Dosovitskiy et al., 2020; Xie et al., 2021; He et al., 2021) domains to pretrain models for downstream tasks. Masked modeling is a broad field that spans many masking strategies, architectures, and applications (Li et al., 2024); this ubiquity is attributed to scalability and architecture breakthroughs (Vaswani et al., 2023) that allow meaningful context to be learned through masked reconstruction (Cao et al., 2022). In the field of neural PDE solvers, masked pretraining has initially been explored as a method to pretrain neural solvers directly (Chen et al., 2024). However, we take a separate approach by investigating if models can understand physics through masked self-supervised learning and how

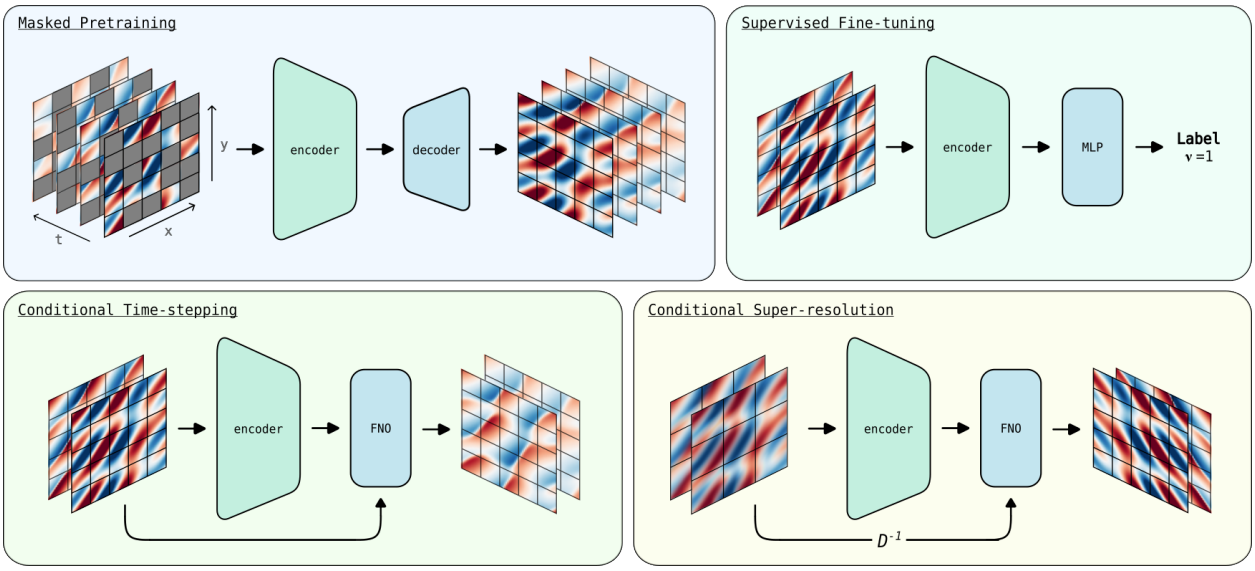

Figure 1: We investigate learning diverse PDE dynamics with masked autoencoders (MAE) and using learned representations to benefit various downstream tasks. (*Masked Pretraining*) An encoder is trained on unmasked patches of spatiotemporal PDE data, while a decoder reconstructs true data from latent encodings and learned mask tokens. (*Supervised Fine-tuning*) Pretrained encoders can be used to quickly regress equation coefficients or predict key PDE features. (*Conditional Time-stepping*) Neural solvers can achieve higher accuracy predictions through conditioning on MAE encodings. (*Conditional Super-resolution*) SR models can also benefit from conditioning on MAE encodings, using a discretization inversion (D^{-1}) and neural operator (Yang et al., 2023) to predict high-resolution physics.

these latent representations are manifested. After validating this learning method, we seek to understand if this knowledge can be used to benefit common PDE tasks.

Situating our Contribution To frame our contribution within these past works, we draw comparisons with broader deep learning efforts to pretrain large encoders through self-supervision, such as BERT (Devlin et al., 2018) or CLIP (Radford et al., 2021; Ramesh et al., 2022), to be used in downstream tasks. For many reasons (e.g. data availability, architectures, pretraining strategies, compute resources), an equivalent work does not currently exist for PDEs. However, we hope to advance this research thrust to train general-purpose physics encoders, which could be of benefit to the PDE community. Specifically, pretrained encoders are often leveraged to accelerate model development and training by outsourcing certain architecture modules to pretrained models. Furthermore, having standardized, encoded representations of diverse PDEs can help in interfacing with conventional ML pipelines, whether it be allowing LLMs to interface with PDE data or diffusion backbones to condition on PDE inputs.

3 Methods

We describe our main methods in Figure 1. We first pretrain an encoder and decoder to reconstruct masked inputs. This pretraining objective has several benefits. Firstly, the pretraining process is entirely self-supervised, so no additional effort is needed to generate pretraining labels. Furthermore, masking destroys inherent biases present in the data; especially at high masking ratios, this forces models to learn very general latent representations. Lastly, masked modeling does not assume any prior physics or leverage features specific to a particular PDE, making the approach applicable to a wide variety of physics.

After pretraining, we evaluate using the learned representations for three downstream tasks after discarding the decoder. The first task is to regress or classify PDE features, such as predicting coefficient values or boundary conditions of a PDE sample. This can be done by directly appending a linear head to the pretrained

encoder. The second task is to improve the performance of neural solvers by conditioning on an encoded representation of input physics. Both the encoded and original representations of the data are passed to the neural solver. Lastly, we consider improving super-resolution performance by passing an up-sampled input to a neural solver and conditioning on an encoded representation of the low-resolution PDE input.

Masked Pretraining for PDEs We adapt the popular Masked Autoencoder (MAE) approach (He et al., 2021; Xie et al., 2021; Feichtenhofer et al.) to train ViT models (Dosovitskiy et al., 2020; Arnab et al.) to reconstruct 1D and 2D PDE data. Data are partitioned into non-overlapping patches before a random subset of these patches is sampled to be masked. The masked patches are omitted from the encoder input; the encoder embeds only the unmasked patches through a series of transformer blocks, which allows for larger encoders and faster training at large masking ratios (He et al., 2021). The encoded patches are then recombined with mask tokens according to their position in the PDE solution. Positional embeddings are added again to preserve positional information before the latent encodings and mask tokens are decoded. The decoder can be shallower and narrower than the encoder because it is discarded in downstream tasks (He et al., 2021), which further reduces training costs. The decoded tokens are projected into the PDE space through a linear layer before reshaping the patches to the input dimension. Lastly, the output is compared to ground truth PDE data through a MSE loss. For additional details on MAE implementation and ablations, we direct readers to Appendix A.1.

Lie Point Symmetry Data Augmentation To emulate a larger pretraining dataset and aid in generalization, we investigate using Lie augmentations during pretraining, an approach that follows the use of data augmentations in vision or video pretraining (He et al., 2021; Xie et al., 2021; Feichtenhofer et al.). Given a PDE, one can derive its Lie symmetries as a set of transformations $\{g_1, \ldots, g_i\}$, each with a variable ϵ_i that modulates the magnitude of the transformation. At training time, we apply g_i sequentially, each with a randomly sampled ϵ_i to randomly augment PDE samples. This augmented PDE sample could represent a solution that has been shifted in space, time, or magnitude, among other transformations, but still propagates dynamics according to the original PDE. For a more detailed discussion of Lie point symmetries for PDEs, we refer the reader to Olver (1986) and Mialon et al. (2023). We find that the use of Lie augmentations improves MSE loss when reconstructing a validation set of PDE data; for a detailed comparison, we invite readers to see Appendix A.2.

Multi-Resolution Pretraining PDE data are often discretized on a mesh, which can be both irregular and of different resolutions. This can present a challenge for vanilla transformers, which use the same positional encodings regardless of variations in input sequence length; indeed, many approaches have been proposed to adapt ViT models to inputs of varying resolutions or patch sizes (Beyer et al., 2023; Tian et al., 2023; Dehghani et al., 2023; Fan et al., 2024). To avoid changes to the ViT backbone, we consider padding variable input sequences to a common length. We also consider an alternate strategy which interpolates positional embeddings to the desired length. Lastly, we investigate encoding the spatial coordinates of the PDE solution grid through an embedder network and incorporating this information by adding an additional token to the input. These modifications tend to improve reconstruction performance when pretraining on multi-resolution PDE data. For additional details and ablation studies, see Appendix A.3.

4 Experiments

4.1 Equations Considered

We describe a variety of PDEs used for masked pretraining and downstream evaluation. In 1D, we pretrain MAE models on the KdV-Burgers equation only, while in 2D we pretrain on the Heat, Advection, and Burgers equations simultaneously. In all PDEs, coefficients and forcing terms are randomly sampled to produce diverse dynamics within a dataset. For additional details regarding PDE parameter sampling, initial conditions, and numerical methods, and we refer readers to Appendix B.

1D KdV-Burgers Equation: The KdV-Burgers equation (KdV-B) contains the Heat, Burgers, KdV equations as corner cases modulated by coefficients (α, β, γ) (Brandstetter et al., 2022; Jeffrey and Mohamad,

1991). Furthermore, periodic boundary conditions are used with a forcing function defined by $\delta(x, t)$. For 1D multi-resolution experiments, PDE data from the KdV-Burgers equation is downsampled to variable spatial resolutions.

$$\partial_t u + \alpha u \partial_x u - \beta \partial_{xx} u + \gamma \partial_{xxx} u = \delta(t, x) \tag{KdV-B}$$

1D Heat and Burgers Equations: Although similar to the 1D KdV-Burgers equation, the 1D Heat and inviscid Burgers (Brandstetter et al., 2022) equation are not explicitly seen during pretraining and are used to evaluate generalization to PDEs resembling pretraining samples. Furthermore, to evaluate extrapolation to unseen boundary conditions (BCs), samples of the Heat equation are also generated with Dirichlet and Neumann BCs in addition to periodic BCs.

$$\partial_t u - \nu \partial_{xx} u = \delta(t, x) \tag{Heat}$$
$$\partial_t u + u \partial_x u = \delta(t, x) \tag{Burgers}$$

1D Advection, Wave, and Kuramoto-Sivashinsky Equations: The Advection (Adv), Wave, and parameter-dependent Kuramoto-Sivashinsky (KS) (Lippe et al., 2023) equations are considered to evaluate downstream performance to completely new equations; the equations contain PDE terms that are unseen during pretraining. Additionally, the Wave equation is generated with Dirichlet and Neumann BCs (Brandstetter et al., 2022) to evaluate unseen BCs on novel PDE dynamics.

$$\partial_t u + c \partial_x u = 0 \tag{Adv}$$
$$\partial_{tt} u - c^2 \partial_{xx} u = 0 \tag{Wave}$$
$$\partial_t u + u \partial_x u + \nu \partial_{xx} u + \partial_{xxxx} u = 0 \tag{KS}$$

2D Heat, Advection, and Burgers Equations: The 2D Heat, Advection (Adv), and scalar Burgers (Rosofsky et al., 2023) equations are considered for both pretraining and downstream evaluation. For 2D multi-resolution experiments, data from these equations are downsampled to variable spatial resolutions.

$$\partial_t u - \nu \nabla^2 u = 0 \tag{Heat}$$
$$\partial_t u + \mathbf{c} \cdot \nabla u = 0 \tag{Adv}$$
$$\partial_t u + u(\mathbf{c} \cdot \nabla u) - \nu \nabla^2 u = 0 \tag{Burgers}$$

2D Navier-Stokes Equations: Following the setup from Li et al. (2020), we consider the incompressible Navier-Stokes (NS) equations in vorticity form, but randomly sample the viscosity ν and forcing function $f(x)$ amplitude. To ensure consistency with the pretraining dataset, our experiments model NS dynamics as a scalar vorticity field; from this the velocity field can be derived.

$$\partial_t \omega + u \cdot \nabla \omega - \nu \nabla^2 \omega = f(x), \quad \nabla \cdot u = 0 \tag{NS}$$

4.2 MAE Pretraining

MAE models are trained on 10000 samples of 1D KdV-Burgers PDE data in 1D and 12288 samples of 2D Heat, Advection, and Burgers PDE data in 2D. We display example results from masked pretraining in Figures 2 and 3. A notable difference from vision and video domains is that physics does not follow human-recognizable structure or descriptions (e.g. backgrounds, actions, faces, shapes, etc.); furthermore, in addition to the overall meaning, the numerical accuracy of the reconstruction is important. Despite this, MAE models are able to capture underlying physics and reconstruct PDEs within the pretraining set well—both in 1D and 2D, and at high masking ratios (75% and 90%). In general, for PDEs that are similar to those seen during pretraining, such as the 1D Heat or inviscid Burgers equation, MAE models tend to extrapolate well. Furthermore, given information about the spatial discretization, MAE models can adapt to different PDE

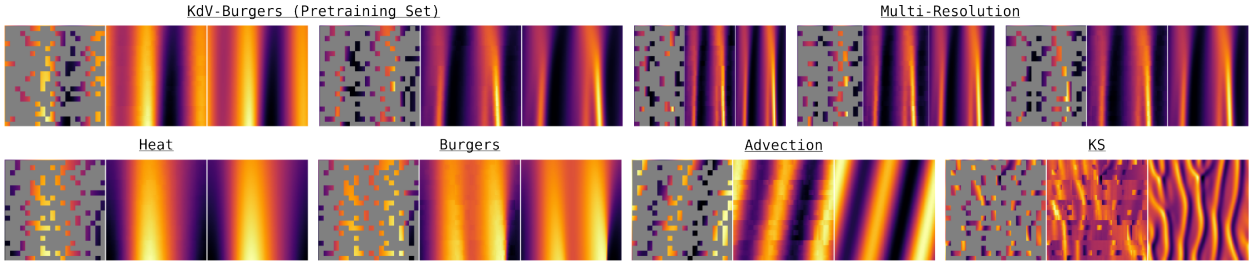

Figure 2: Example results after training on the 1D KdV-Burgers equation **only** with a masking ratio of 75%. For each triplet, we show the masked PDE (left), the MAE reconstruction (middle), and the ground-truth (right), and plot space and time on the x and y axes respectively. The MAE is able to reconstruct multiple resolutions of KdV-Burgers data, as well as extrapolate to the 1D Heat and inviscid Burgers equations, which are unseen during training. For the 1D Advection and KS equations, which contain novel PDE terms (u_x, u_{xxxx}), the extrapolation performance is limited.

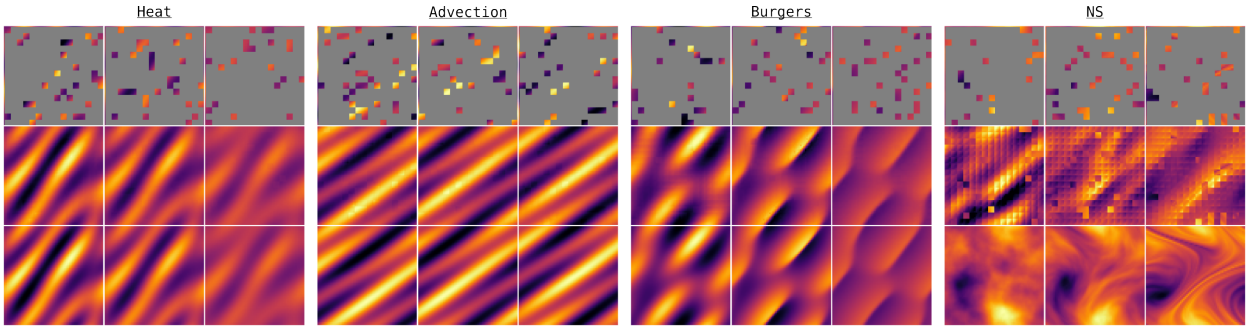

Figure 3: Example results after training on a combined set of the 2D Heat, Advection, and Burgers equations with a masking ratio of 90%. We show the masked PDE (top), the MAE reconstruction (middle), and the ground truth (bottom), plotting space on the x-y axis at multiple snapshots in time. Despite good performance within the training set, the model is unable to extrapolate to the Navier-Stokes (NS) equations, which contain novel initial conditions, forcing terms, and dynamics.

resolutions when trained to reconstruct multi-resolution inputs. This is true in 2D as well, with example results shown in Appendix C.2.

For PDEs that contain novel equation terms or boundary conditions (BCs), the MAE extrapolation performance is limited. Zero-shot reconstruction of the 1D Advection and KS equations shows mild trends, while reconstruction of the 2D Navier-Stokes equations is ineffective. To address this gap and investigate whether MAE models can perform on complex high-resolution physics with multiple variables, we train an MAE model to reconstruct pressure and velocity on 2D smoke buoyancy data with varying buoyancy factors (Gupta and Brandstetter, 2022) and qualitatively show its reconstruction in Appendix C.3. In general, MAE models can adapt to complex scenarios and multiple physical variables; however, many of the fine details (e.g. eddies, shedding) become lost at high masking ratios.

Since the 1D KdV-Burgers equation is solved with periodic BCs, we evaluate MAE extrapolation to Dirichlet and Neumann BCs when reconstructing the 1D Heat and Wave equations, shown in Appendix C.1. Overall trends remain consistent; the Heat equation, being more similar to the KdV-Burgers equation, shows limited reconstruction performance when extrapolating to novel BCs, while the Wave equation introduces a novel PDE term (u_{tt}) and initial condition (Gaussian pulse), and as a result the zero-shot reconstruction is poor.

4.3 Latent Space Evaluation

Latent Embeddings To better understand the latent representation learned by masked pretraining, we use the MAE encoder to embed various PDE validation samples and visualize embeddings with t-SNE (van der

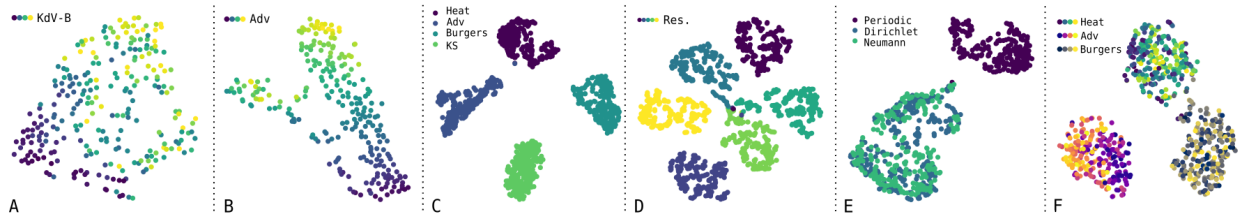

Figure 4: MAE embeddings of various PDEs after masked pretraining; the MAE latent space shows structure despite not seeing labels. **A:** 1D KdV-Burgers equation, colored by α. **B:** 1D Advection equation, colored by c. **C:** 1D Heat, Burgers, Advection, and KS equations, colored by PDE. **D:** 1D KdV-Burgers equation, colored by resolution. **E:** 1D Heat equation, colored by boundary condition. **F:** 2D Heat, Advection and Burgers equations, colored by ν and c.

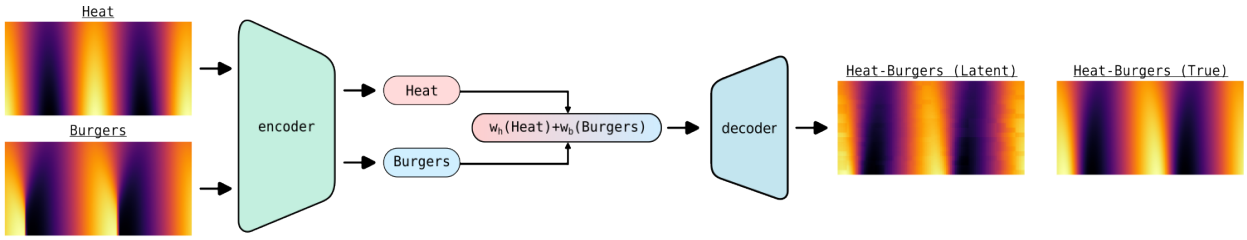

Figure 5: A proposed setup for operating on latent PDE vectors. PDE data is encoded after masked pretraining and summed in the latent space before being decoded. These reconstructions can approximate summed PDEs in physical space.

Maaten and Hinton, 2008) in Figure 4. Through self-supervised learning, MAE models can learn trends in PDEs without labeled data and in equations outside the training set. For example, through masked reconstruction of the 1D KdV-Burgers equation, MAE models can learn coefficient-dependent trends (Figure 4**A**). This can be applied to PDEs not seen during training, as the same MAE model can distinguish samples from the Advection equation based on wave speed c (Figure 4**B**). This extrapolation is also observed when embedding samples from different PDEs; representations learned from pretraining on the KdV-Burgers equations allow models to cluster PDE samples from the Heat, Burgers, Advection, and KS equations (Figure 4**C**). Lastly, MAE models are able to distinguish varying resolutions of PDE data after multi-resolution training, suggesting high model capacity across both diverse PDEs and discretizations (Figure 4**D**).

There are certainly limitations to emergent trends learned by masked pretraining. Unseen boundary conditions can be challenging; when pretrained on periodic 1D KdV-Burgers data, MAE models can only distinguish between periodic and non-periodic Heat equation samples without understanding differences between Dirichlet and Neumann BCs (Figure 4**E**). In addition, trends in 2D PDE data are more difficult to learn without labels. Despite separating between Heat, Adv, and Burgers samples, latent trends are only observed in 2D Advection samples based on wave speed **c** (Figure 4**F**).

Latent PDE Arithmetic MAE encoders have shown strong capabilities in extracting information from self-supervised PDE learning, creating opportunities to operate in this latent space. This could be beneficial since many PDEs are compositions of simpler phenomena, and recombining PDE solutions in a latent space may result in obtaining novel PDE solutions for free. After pretraining on the 1D KdV-Burgers equation, we consider an arithmetic task where samples of the Heat and Burgers equation are embedded and added in the latent space before being decoded to a novel solution (Figure 5). Concretely, we generate 1D Heat data from the PDE: $\partial_t u - \nu \partial_{xx} u = 0$, 1D inviscid Burgers data from the PDE: $\partial_t u + u \partial_x u = 0$, and 1D Heat-Burgers data from adding the two PDEs: $\partial_t u - \nu \partial_{xx} u + u \partial_x u = 0$. When given identical initial conditions and coefficients, PDE reconstructions obtained from summing latent Heat and Burgers embeddings qualitatively resemble solutions of the combined Heat-Burgers PDE. Notably, the MAE encoder/decoder has never seen the 1D Heat or Burgers equation on their own.

Table 1: 1D PDE feature prediction after MAE pretraining on the KdV-Burgers equation. Models are fine-tuned on 2000 held-out, labeled samples for each task. We consider regressing coefficients across four PDEs as well as classifying BCs of the heat/wave equation, identifying equations from a mixed set of PDEs, and sorting different spatial resolutions of a PDE. Regression errors are given as RMSE$\times 10^{-2}$ and classification errors are given as X-Ent$\times 10^{-4}$, averaged over 5 seeds.

Model	KdV-B	Heat	Adv	KS	Heat_{BC}	Wave_{BC}	PDEs	Res.
Base	3.454	0.834	**0.241**	0.354	0.123	**0.022**	0.355	64.52
MAE_{f}	1.334	0.677	0.551	0.368	1.164	1.816	0.174	63.34
MAE	**0.905**	**0.505**	0.244	**0.156**	**0.018**	0.053	**0.025**	**28.32**

Table 2: 2D PDE feature prediction after MAE pretraining on a combined set of 2D Heat, Advection, and Burgers equations. Models are fine-tuned on 1024 held-out, labeled samples for each task, or 3072 samples in the combined case. Regression errors are given as RMSE$\times 10^{-1}$ and classification errors are given as X-Ent$\times 10^{-1}$, averaged over 5 seeds.

Model	Heat	Adv	Burgers	Combined	NS	Res.
Base	0.084	**0.506**	0.682	0.320	0.748	0.694
MAE_{f}	0.232	0.540	0.606	0.384	0.709	0.636
MAE	**0.062**	0.507	**0.409**	**0.265**	**0.594**	**0.005**

In addition, if different latent embeddings can be added, the weighting of each embedding can be varied to control the resemblance of the reconstruction to make interpolated samples that have more shock formation or diffusive behavior (e.g., more resemble the Burgers or Heat equation). We anticipate that this work can be expanded and improved upon. In particular, the decoder has only seen masked data during training, and as a result latent reconstructions must be done with partially encoded information.

4.4 PDE Feature Prediction

To evaluate the latent representations learned from masked pretraining, we regress PDE coefficients and classify PDE boundary conditions, equation families, and spatial resolutions from an initial time window. Regression tasks are separated by PDE (*KdV-B, KS, Heat, Adv, Burgers, NS*). Further, classification tasks are formulated as: ($Heat_{BC}$): predicting Periodic, Dirichlet, or Neumann BCs from the Heat equation, ($Wave_{BC}$): predicting Dirichlet or Neumann BCs from the Wave equation, (*PDEs*): predicting unseen PDEs from Heat, Adv, Burgers, and KS equation samples, and (*Res.*): predicting resolutions from $n_x = \{50, 60, 70, 80, 90, 100\}$ in 1D or $(n_x, n_y) = \{(48, 48), (52, 52), (56, 56), (60, 60), (64, 64)\}$ in 2D. Lastly, several model variants are evaluated: a randomly initialized ViT baseline (Base), a pretrained, frozen MAE encoder with a linear head (MAE_{f}), and a pretrained, fine-tuned MAE encoder (MAE). Results are shown in Tables 1 & 2, with full error bars in Appendix E.1.

In 1D, the frozen MAE_{f} encoder is able to outperform a supervised baseline on the *KdV-B*, *Heat*, *PDEs*, and *Res.* tasks despite never seeing labels throughout training. Further performance gains can be realized by allowing the MAE encoder to fine-tune on labeled data, and can outperform random initialization on unseen equations and BCs. An exception to this is the Advection and Wave equations. We hypothesize that these PDEs are heavily governed by the wave speed c and boundary conditions (bouncing vs. reflecting waves) and are simple trends that supervised models can learn quickly.

In 2D, the effects of fine-tuning is more pronounced. Within the pretraining set, only in the 2D Burgers task does freezing the MAE encoder outperform a supervised baseline; nevertheless, masked pretraining serves as a good initialization for supervised fine-tuning. In addition, despite differing physics, prior knowledge from simpler 2D PDEs seems to benefit regression on the Navier-Stokes equations. When classifying 2D Heat, Advection, and Burgers data based on their discretization, MAE models greatly benefit from pretraining on multi-resolution data. We hypothesize that in 2D PDEs, variable spatial resolutions can be challenging to distinguish due to flattening the spatial dimension when patchifying inputs, whereas in 1D PDEs the data is already flattened.

Table 3: Conditional 1D PDE time-stepping and super-resolution. Models are trained on 2000 held-out fine-tuning samples to predict or upsample physics across several settings. Validation errors are reported as normalized L2 loss (time-stepping) or RMSE$\times 10^{-1}$ (SR) summed over all timesteps and averaged across five seeds.

Model	KdV-B	Heat	Burgers	Adv	KS	Heat_{BC}	Wave_{BC}	Res.
FNO	1.153	0.671	1.094	0.437	0.830	2.408	0.147	1.141
FNO-MAE_{f}	1.043	0.655	1.121	0.431	0.821	**1.747**	**0.135**	**1.018**
FNO-MAE	**1.037**	**0.643**	**0.952**	**0.294**	**0.812**	1.846	0.148	1.07
Unet	0.823	0.420	0.649	0.194	1.333	**4.249**	0.747	0.766
Unet-MAE_{f}	0.806	0.425	0.582	**0.177**	1.241	4.734	0.699	0.688
Unet-MAE	**0.758**	**0.363**	**0.546**	0.210	**1.125**	5.157	**0.659**	**0.683**
SR	0.520	0.203	0.881	0.223	2.673	0.210	0.516	—
SR-MAE_{f}	0.481	0.173	0.691	**0.169**	2.460	0.204	0.376	—
SR-MAE	**0.475**	**0.151**	**0.676**	0.194	**2.422**	**0.170**	**0.349**	—

4.5 Conditional Time-stepping and Super-resolution

1D Experiments To evaluate the use of the MAE encoder for practical tasks, we train a neural solver on various 1D PDEs to predict or upsample physics. In particular, we consider five PDEs (*KdV-B, Heat, Burgers, Adv, KS*) as well as two PDEs under varying boundary conditions ($Heat_{BC}$, $Wave_{BC}$) and predicting physics on various resolutions of the KdV-Burgers equation (*Res.*). For time-stepping, we incorporate temporal bundling and the pushforward trick to stabilize training (Brandstetter et al., 2022). Furthermore, we test on FNO (Li et al., 2020) and Unet architectures (Gupta and Brandstetter, 2022), 64, and add conditioning information to hidden states after convolutions (Ho et al., 2020; Nichol and Dhariwal, 2021). For super-resolution (SR), we implement a pipeline in which a network encodes low-resolution physics before upsampling with a discretization inversion operator D^{-1} (e.g. linear, bicubic interpolation) and mapping to an output functional space with a neural operator (Yang et al., 2023). Following this, we implement a Resnet encoder (Wang et al., 2021; Zhang et al., 2018) followed by an interpolation scheme and FNO operator; both Resnet and FNO models are provided conditioning information from MAE encodings of low-resolution physics. For additional details, see Appendix D.

After pretraining an MAE encoder on the 1D KdV-Burgers equation, we compare the base neural solvers (FNO, Unet, SR) to conditioning on a frozen MAE embedding (-MAE_{f}) and allowing the MAE encoder to fine-tune when conditioning (-MAE). Results in 1D are presented in Table 3. Within the pretraining distribution (KdV-B) and certain PDEs, MAE conditioning consistently improves time-stepping and super-resolution performance. In addition, allowing MAE encoders to fine-tune can further lower errors. However, there are various exceptions, in particular PDEs with unseen boundary conditions. Despite this, improvements are consistent across different neural solver architectures, suggesting that pretrained MAE models can be agnostic to downstream model choices.

2D Experiments Following the setup in 1D, we repeat time-stepping/super-resolution experiments on 2D PDEs (*Heat, Adv, Burgers, NS*) and a combined set of 2D Heat, Advection, and Burgers equations (*Combined*). Additionally, we evaluate time-stepping performance on the combined 2D Heat, Advection, and Burgers equations discretized at variable resolutions (*Res.*). We follow the same conditioning and training strategies as 1D experiments, but modify the architectures to support 2D inputs, and present results in Table 4. After pretraining an MAE encoder on the 2D Heat, Advection, and Burgers equations, we observe improvements in conditional physics prediction and upsampling. Improvements tend to be more pronounced in 2D; we hypothesize that the increased difficulty of the task increases the importance of MAE encoder guidance in time-stepping and super-resolution. This increased difficulty also manifests itself when extrapolating pretrained encoders to new PDEs, as the performance on the Navier-Stokes equations is limited. Nevertheless, we observe similar trends whereby MAE conditioning is agnostic to downstream architecture choices.

Table 4: Conditional 2D PDE time-stepping and super-resolution. Models are trained on 1024 held-out fine-tuning samples, or 3072 in the combined case, to predict or upsample physics. Validation errors are reported as normalized L2 loss (time-stepping) or RMSE$\times 10^{-1}$ (SR) summed over all timesteps and averaged across three seeds.

Model	Heat	Adv	Burgers	Combined	NS	Res.
FNO	0.427	2.301	0.417	0.978	**0.466**	1.006
FNO-MAE$_\text{f}$	0.233	1.179	0.252	0.607	0.59	0.701
FNO-MAE	**0.128**	**1.135**	**0.198**	**0.494**	0.477	**0.499**
Unet	0.147	1.795	0.226	0.835	0.713	0.908
Unet-MAE$_\text{f}$	0.136	1.804	0.229	0.761	**0.669**	0.861
Unet-MAE	**0.116**	**1.230**	**0.186**	**0.669**	0.692	**0.676**
SR	0.175	2.014	0.295	0.534	**0.326**	—
SR-MAE$_\text{f}$	0.159	0.659	0.264	**0.407**	0.347	—
SR-MAE	**0.152**	**0.639**	**0.253**	0.472	0.337	—

Additional Benchmarks We consider two additional benchmarks: a randomly initialized ViT encoder that embeds PDE inputs to a conditioning vector as well as a linear model that encodes ground-truth coefficient or boundary condition information to a conditioning vector. We present detailed results in Appendix E.2 and E.4, and discuss the overall results here. In general, we observe that the randomly initialized, fine-tuned encoder also improves PDE prediction and upsampling, and this improvement generally matches or outperforms the performance of the frozen MAE encoder. However, allowing the MAE encoder to fine-tune generally outperforms this random initialization and approaches the linear benchmark.

Although the linear benchmark generally performs the best in 1D, there are certain exceptions to this. Equations dominated by the coefficient response generally suffer from coefficient conditioning; we observe that the model heavily overfits to the true coefficient and does not learn the underlying PDE dynamics. Furthermore, for PDEs that do not have coefficient information, such as the 1D inviscid Burgers equation, this linear benchmark cannot be applied. In these cases, MAE encodings can still improve performance, suggesting that there are latent PDE features beyond coefficient information that neural solvers can benefit from. Lastly, in 2D the linear benchmark performs much worse, only achieving the lowest error in a few scenarios, and in some cases harming performance of the base model. This could be because PDE dynamics becomes much more complex in 2D and relies less on coefficient information, which is a low-dimensional vector and provides sparse information.

Lastly, to benchmark our method against other pretraining methods, we consider pretraining an encoder using a contrastive self-supervised technique proposed by Mialon et al. (2023), which relies on using Lie symmetries to cluster PDE data in a learned latent space. We contrastively pretrain an encoder on 1D KdV-Burgers data and the evaluate conditional timestepping performance on various downstream 1D PDEs. We present these results in Appendix E.3. To summarize, our approach is on par with Lie contrastive pretraining for PDE samples within the pretraining distribution; however, when extrapolating to unseen PDEs, masked pretraining is able to outperform contrastive methods.

Limitations Despite the work presented so far, we recognize limitations of the MAE encoder. It can be costly to fully fine-tune the pretrained encoder during time-stepping or super-resolution since it operates on all unmasked tokens, and as a result does not benefit from the speed gains during MAE pretraining. To address this, freezing the MAE encoder greatly improves the training speed but decreases performance, especially on unseen PDEs. With enough PDE data during pretraining, freezing MAE encoders can be more practical since the pretraining distribution would cover most downstream cases. Lastly, for simpler 1D dynamics when coefficient information is available, conditioning on ground-truth PDE parameters remains the best choice in most scenarios. However, these approaches are not exclusive; initial work suggests that models provided with both ground-truth information and MAE embeddings can outperform models provided with just one of either.

5 Conclusion

We have presented a new method that extends masked pretraining from vision and video domains to physics, and evaluated several PDE-specific modifications. We empirically validate MAE models through reconstructing a diverse set of 1D and 2D PDEs and show generalization behavior to different spatial discretizations and unseen equations. Furthermore, we evaluate the latent representations learned during MAE training and find structured trends despite training without labeled data. Additionally, we propose operating in this latent space to interpolate between encoded PDE vectors. In practice, MAE encoders can also be used to improve PDE feature prediction, time-stepping, and super-resolution tasks across diverse physics scenarios. A promising direction would be to scale MAE models to larger datasets, as the current approach enjoys the same scalability as the originally proposed MAE (He et al., 2021). Furthermore, future work could explore manipulating latent physics to generate new solutions or performing more sophisticated operations in this latent space. To promote future work, all code and datasets used will also be released.

References

[1] Richard FitzHugh. Impulses and physiological states in theoretical models of nerve membrane. *Biophysical Journal*, 1(6):445–466, 1961. ISSN 0006-3495. doi: https://doi.org/10.1016/S0006-3495(61)86902-6. URL https://www.sciencedirect.com/science/article/pii/S0006349561869026.

[2] J. Nagumo, S. Arimoto, and S. Yoshizawa. An active pulse transmission line simulating nerve axon. *Proceedings of the IRE*, 50(10):2061–2070, 1962. doi: 10.1109/JRPROC.1962.288235.

[3] Edward N. Lorenz. Deterministic nonperiodic flow. *Journal of Atmospheric Sciences*, 20(2):130 – 141, 1963. doi: 10.1175/1520-0469(1963)020<0130:DNF>2.0.CO;2. URL https://journals.ametsoc.org/view/journals/atsc/20/2/1520-0469_1963_020_0130_dnf_2_0_co_2.xml.

[4] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2018.10.045. URL https://www.sciencedirect.com/science/article/pii/S0021999118307125.

[5] Lu Lu, Pengzhan Jin, and George Em Karniadakis. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. 10 2019. doi: 10.1038/s42256-021-00302-5. URL http://arxiv.org/abs/1910.03193http://dx.doi.org/10.1038/s42256-021-00302-5.

[6] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. 10 2020. URL http://arxiv.org/abs/2010.08895.

[7] Shuhao Cao. Choose a transformer: Fourier or galerkin, 2021.

[8] Johannes Brandstetter, Daniel E. Worrall, and Max Welling. Message passing neural PDE solvers. *CoRR*, abs/2202.03376, 2022. URL https://arxiv.org/abs/2202.03376.

[9] Zijie Li, Kazem Meidani, and Amir Barati Farimani. Transformer for partial differential equations' operator learning, 2023.

[10] Nikola Kovachki, Samuel Lanthaler, and Siddhartha Mishra. On universal approximation and error bounds for fourier neural operators. *Journal of Machine Learning Research*, 22(290):1–76, 2021. URL http://jmlr.org/papers/v22/21-0806.html.

[11] Zijie Li, Anthony Zhou, Saurabh Patil, and Amir Barati Farimani. Cafa: Global weather forecasting with factorized attention on sphere, 2024. URL https://arxiv.org/abs/2405.07395.

[12] Makoto Takamoto, Francesco Alesiani, and Mathias Niepert. Learning neural pde solvers with parameter-guided channel attention. 4 2023. URL http://arxiv.org/abs/2304.14118.

[13] Cooper Lorsung, Zijie Li, and Amir Barati Farimani. Physics informed token transformer for solving partial differential equations, 2024.

[14] Junhong Shen, Tanya Marwah, and Ameet Talwalkar. Ups: Towards foundation models for pde solving via cross-modal adaptation, 2024.

[15] Shashank Subramanian, Peter Harrington, Kurt Keutzer, Wahid Bhimji, Dmitriy Morozov, Michael Mahoney, and Amir Gholami. Towards foundation models for scientific machine learning: Characterizing scaling and transfer behavior. 5 2023. URL http://arxiv.org/abs/2306.00258.

[16] Michael McCabe, Bruno Régaldo-Saint Blancard, Liam Holden Parker, Ruben Ohana, Miles Cranmer, Alberto Bietti, Michael Eickenberg, Siavash Golkar, Geraud Krawezik, Francois Lanusse, Mariel Pettee, Tiberiu Tesileanu, Kyunghyun Cho, and Shirley Ho. Multiple physics pretraining for physical surrogate models, 2023.

[17] Zhongkai Hao, Chang Su, Songming Liu, Julius Berner, Chengyang Ying, Hang Su, Anima Anandkumar, Jian Song, and Jun Zhu. Dpot: Auto-regressive denoising operator transformer for large-scale pde pre-training, 2024.

[18] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, Anima Anandkumar, and Lorenzo Rosasco. Neural operator: Learning maps between function spaces with applications to pdes, 2023.

[19] Jayesh K. Gupta and Johannes Brandstetter. Towards multi-spatiotemporal-scale generalized pde modeling. 9 2022. URL http://arxiv.org/abs/2209.15616.

[20] Lu Lu, Xuhui Meng, Shengze Cai, Zhiping Mao, Somdatta Goswami, Zhongqiang Zhang, and George Em Karniadakis. A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data. 11 2021. doi: 10.1016/j.cma.2022.114778. URL http://arxiv.org/abs/2111.05512http://dx.doi.org/10.1016/j.cma.2022.114778.

[21] Rui Wang, Robin Walters, and Rose Yu. Meta-learning dynamics forecasting using task inference. 2 2021. URL http://arxiv.org/abs/2102.10271.

[22] Matthieu Kirchmeyer, Yuan Yin, Jérémie Donà, Nicolas Baskiotis, Alain Rakotomamonjy, and Patrick Gallinari. Generalizing to new physical systems via context-informed dynamics model.

[23] Yuan Yin, Ibrahim Ayed, Emmanuel de Bézenac, Nicolas Baskiotis, and Patrick Gallinari. Leads: Learning dynamical systems that generalize across environments. 6 2021. URL http://arxiv.org/abs/2106.04546.

[24] Lu Zhang, Huaiqian You, Tian Gao, Mo Yu, Chung-Hao Lee, and Yue Yu. Metano: How to transfer your knowledge on learning hidden physics. 1 2023. URL http://arxiv.org/abs/2301.12095.

[25] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. *CoRR*, abs/2002.05709, 2020. URL https://arxiv.org/abs/2002.05709.

[26] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2015. doi: 10.1109/cvpr.2015.7298682. URL http://dx.doi.org/10.1109/CVPR.2015.7298682.

[27] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction, 2021.

[28] Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning, 2022.

[29] Cooper Lorsung and Amir Barati Farimani. Picl: Physics informed contrastive learning for partial differential equations, 2024.

[30] Rui Zhang, Qi Meng, and Zhi-Ming Ma. Deciphering and integrating invariants for neural operator learning with various physical mechanisms. *National Science Review*, 11(4), December 2023. ISSN 2053-714X. doi: 10.1093/nsr/nwad336. URL http://dx.doi.org/10.1093/nsr/nwad336.

[31] Grégoire Mialon, Quentin Garrido, Hannah Lawrence, Danyal Rehman, Yann LeCun, and Bobak T. Kiani. Self-supervised learning with lie symmetries for partial differential equations. 7 2023. URL http://arxiv.org/abs/2307.05432.

[32] Johannes Brandstetter, Max Welling, and Daniel E. Worrall. Lie point symmetry data augmentation for neural pde solvers. 2 2022. URL http://arxiv.org/abs/2202.07643.

[33] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models, 2022.

[34] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[35] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. URL https://github.com/codelucas/newspaper.

[36] Liu Yang, Siting Liu, Tingwei Meng, and Stanley J Osher. In-context operator learning with data prompts for differential equation problems. 2023. doi: 10.1073/pnas. URL https://doi.org/10.1073/pnas.2310142120.

[37] Wuyang Chen, Jialin Song, Pu Ren, Shashank Subramanian, Dmitriy Morozov, and Michael W. Mahoney. Data-efficient operator learning via unsupervised pretraining and in-context learning. 2 2024. URL http://arxiv.org/abs/2402.15734.

[38] Somdatta Goswami, Katiana Kontolati, Michael D. Shields, and George Em Karniadakis. Deep transfer operator learning for partial differential equations under conditional shift. *Nature Machine Intelligence*, 4(12):1155–1164, December 2022. ISSN 2522-5839. doi: 10.1038/s42256-022-00569-2. URL http://dx.doi.org/10.1038/s42256-022-00569-2.

[39] Ayan Chakraborty, Cosmin Anitescu, Xiaoying Zhuang, and Timon Rabczuk. Domain adaptation based transfer learning approach for solving pdes on complex geometries. *Engineering with Computers*, 38(5): 4569–4588, Oct 2022. ISSN 1435-5663. doi: 10.1007/s00366-022-01661-2. URL https://doi.org/10.1007/s00366-022-01661-2.

[40] Hengjie Wang, Robert Planas, Aparna Chandramowlishwaran, and Ramin Bostanabad. Mosaic flows: A transferable deep learning framework for solving pdes on unseen domains. *Computer Methods in Applied Mechanics and Engineering*, 389, 2 2022. ISSN 00457825. doi: 10.1016/j.cma.2021.114424.

[41] Anthony Zhou, Cooper Lorsung, AmirPouya Hemmasian, and Amir Barati Farimani. Strategies for pretraining neural operators, 2024. URL https://arxiv.org/abs/2406.08473.

[42] Tapas Tripura and Souvik Chakraborty. A foundational neural operator that continuously learns without forgetting, 2023.

[43] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. 10 2018. URL http://arxiv.org/abs/1810.04805.

[44] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. 10 2020. URL http://arxiv.org/abs/2010.11929.

[45] Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. Simmim: A simple framework for masked image modeling. *CoRR*, abs/2111.09886, 2021. URL https://arxiv.org/abs/2111.09886.

[46] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross B. Girshick. Masked autoencoders are scalable vision learners. *CoRR*, abs/2111.06377, 2021. URL https://arxiv.org/abs/2111.06377.

[47] Siyuan Li, Luyuan Zhang, Zedong Wang, Di Wu, Lirong Wu, Zicheng Liu, Jun Xia, Cheng Tan, Yang Liu, Baigui Sun, and Stan Z. Li. Masked modeling for self-supervised representation learning on vision and beyond, 2024.

[48] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[49] Shuhao Cao, Peng Xu, and David A. Clifton. How to understand masked autoencoders. 2 2022. URL http://arxiv.org/abs/2202.03670.

[50] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.

[51] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents, 2022.

[52] Qidong Yang, Alex Hernandez-Garcia, Paula Harder, Venkatesh Ramesh, Prasanna Sattegeri, Daniela Szwarcman, Campbell D. Watson, and David Rolnick. Fourier neural operators for arbitrary resolution climate data downscaling, 2023.

[53] Christoph Feichtenhofer, Haoqi Fan, Yanghao Li, Kaiming He, and Meta Ai. Masked autoencoders as spatiotemporal learners. URL https://github.com/facebookresearch/mae_st.

[54] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lucic, and Cordelia Schmid. Vivit: A video vision transformer.

[55] Peter Olver. *Applications of Lie Groups to Differential Equations*. Springer New York, NY, 1986.

[56] Lucas Beyer, Pavel Izmailov, Alexander Kolesnikov, Mathilde Caron, Simon Kornblith, Xiaohua Zhai, Matthias Minderer, Michael Tschannen, Ibrahim Alabdulmohsin, and Filip Pavetic. Flexivit: One model for all patch sizes, 2023.

[57] Rui Tian, Zuxuan Wu, Qi Dai, Han Hu, Yu Qiao, and Yu-Gang Jiang. Resformer: Scaling vits with multi-resolution training, 2023.

[58] Mostafa Dehghani, Basil Mustafa, Josip Djolonga, Jonathan Heek, Matthias Minderer, Mathilde Caron, Andreas Steiner, Joan Puigcerver, Robert Geirhos, Ibrahim Alabdulmohsin, Avital Oliver, Piotr Padlewski, Alexey Gritsenko, Mario Lučić, and Neil Houlsby. Patch n' pack: Navit, a vision transformer for any aspect ratio and resolution, 2023.

[59] Qihang Fan, Quanzeng You, Xiaotian Han, Yongfei Liu, Yunzhe Tao, Huaibo Huang, Ran He, and Hongxia Yang. Vitar: Vision transformer with any resolution, 2024.

[60] A. Jeffrey and M.N.B. Mohamad. Exact solutions to the kdv-burgers' equation. *Wave Motion*, 14 (4):369–375, 1991. ISSN 0165-2125. doi: https://doi.org/10.1016/0165-2125(91)90031-I. URL https://www.sciencedirect.com/science/article/pii/016521259190031I.

[61] Phillip Lippe, Bastiaan S. Veeling, Paris Perdikaris, Richard E. Turner, and Johannes Brandstetter. Pde-refiner: Achieving accurate long rollouts with neural pde solvers, 2023.

[62] Shawn G. Rosofsky, Hani Al Majed, and E A Huerta. Applications of physics informed neural operators. *Machine Learning: Science and Technology*, 4, 2023.

[63] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. URL `http://jmlr.org/papers/v9/vandermaaten08a.html`.

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

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

[66] Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models, 2021.

[67] Xintao Wang, Liangbin Xie, Chao Dong, and Ying Shan. Real-esrgan: Training real-world blind super-resolution with pure synthetic data, 2021.

[68] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution, 2018.

[69] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

[70] Nail H. Ibragimov. *CRC Handbook of Lie Group Analysis of Differential Equations: Symmetries, Exact Solutions, and Conservation Laws*. Number v. 1. Taylor & Francis, 1993. ISBN 9780849344886. URL `https://books.google.com/books?id=mpm5Yq1q6T8C`.

[71] Martin S. Alnaes, Jan Blechta, Johan Hake, August Johansson, Benjamin Kehlet, Anders Logg, Chris N. Richardson, Johannes Ring, Marie E. Rognes, and Garth N. Wells. The FEniCS project version 1.5. *Archive of Numerical Software*, 3, 2015. doi: 10.11588/ans.2015.100.20553.

[72] Anders Logg, Kent-Andre Mardal, Garth N. Wells, et al. *Automated Solution of Differential Equations by the Finite Element Method*. Springer, 2012. doi: 10.1007/978-3-642-23099-8.

Table 5: MAE Hyperparameters during pretraining.

(a) 1D PDEs

Parameters	Value
Batch Size	256
Epochs	20
Encoder Dim	256
Decoder Dim	32
Patch Size	(5, 5)
Masking Ratio	0.75
Time Window	20
Augmentation Ratio	0.5
Base LR	1e-3
Optimizer	AdamW
Scheduler	OneCycleLR

(b) 2D PDEs

Parameters	Value
Batch Size	64
Epochs	20
Encoder Dim	256
Decoder Dim	32
Patch Size	(4, 4, 4)
Masking Ratio	0.90
Time Window	16
Augmentation Ratio	0.5
Base LR	1e-3
Optimizer	AdamW
Scheduler	OneCycleLR

A MAE Implementation and Ablation Studies

A.1 Implementation

To implement the MAE encoder and decoder, we use a ViT architecture (44), which uses a self-attention layer (48) and MLP, both with LayerNorms (69). For MAE models evaluated in the paper trained on 1D and 2D PDEs, we present the pretraining parameters in Table 5. To study the effects of various hyperparameters, including model size, masking ratio, and patch size, we run ablation studies on masked reconstruction of 1D PDEs, and report reconstruction MSE errors on a validation set in Table 6. Overall, we find that increasing model size in limited data regimes—only 10000 KdV-Burgers samples were used in pretraining—tends to contribute to overfitting and increases validation errors. Predictably, increasing the masking ratio increases reconstruction errors as a result of less information being provided to MAE models. Furthermore, decreasing the patch size reduces errors but requires a higher computational cost, which is consistent with results in CV domains (56).

Following these design considerations, we use a MAE model with 5M params; pretraining with this setup takes around 3 hours on a NVIDIA GeForce RTX 2080 Ti GPU for the 1D KdV-Burgers dataset. In 2D, we use a MAE model with 10M params, which takes about 1 day to pretrain on a NVIDIA RTX A6000 GPU on the 2D Heat, Advection, and Burgers dataset.

Table 6: MAE model ablation studies on the 1D KdV-Burgers equation.

(a) Model Size

# Params	Error
1M	2.37e-03
5M	2.48e-03
25M	3.36e-03

(b) Masking Ratio

Masking Ratio	Error
0.6	8.02e-04
0.75	1.66e-03
0.90	6.68e-03

(c) Patch Size, in (p_t, p_x)

Patch Size	Error
(5, 5)	1.12e-03
(4, 4)	7.05e-04
(4, 2)	6.46e-04
(2, 4)	4.79e-04

A.2 Lie Point Symmetry Data Augmentations

We implement Lie Point Symmetry Data Augmentations according to Brandstetter et al. (32), including shifting and resampling PDE solutions with the Fourier shift theorem. Since we only augment PDE samples during pretraining, we consider symmetry groups for the 1D KdV-Burgers equation, as well as the 2D Heat, Advection, and Burgers equations. The 1D KdV-Burgers equation has the following symmetries, adapted from the generalized KdV-Burgers equation (70):

$$\partial_t u + \frac{j}{2} + \alpha u^m \partial_x u - \beta \partial_{xx} u + \gamma \partial_{xxx} u = 0 \tag{1}$$

We consider the special case where $j = 0, m = 1$, resulting in the following Lie subalgebras (70):

$$X_1 = \frac{\partial}{\partial t}, \quad X_2 = \frac{\partial}{\partial x}, \quad X_3 = \alpha t \frac{\partial}{\partial x} + \frac{\partial}{\partial u} \tag{2}$$

Taking the exponential map results in the following Lie groups (70):

$$g_1(\epsilon)(x, t, u) = (x, t + \epsilon, u), \quad \text{(Time Shift)} \tag{3}$$
$$g_2(\epsilon)(x, t, u) = (x + \epsilon, t, u), \quad \text{(Space Shift)} \tag{4}$$
$$g_3(\epsilon)(x, t, u) = (x + \epsilon t, t, u + \epsilon), \quad \text{(Galilean Boost)} \tag{5}$$

For the 2D Heat, Advection, and Burgers equations, there are many possible Lie subalgebras (70). For simplicity, we only consider a basic subset of these that apply to all three equations, however, there is ample room to implement more symmetries:

$$X_1 = \frac{\partial}{\partial t}, \quad X_2 = \frac{\partial}{\partial x}, \quad X_3 = \frac{\partial}{\partial y}, \tag{6}$$

These result in the following Lie groups in 2D:

$$g_1(\epsilon)(x, y, t, u) = (x, y, t + \epsilon, u), \quad \text{(Time Shift)} \tag{7}$$
$$g_2(\epsilon)(x, y, t, u) = (x + \epsilon, y, t, u), \quad \text{(X Shift)} \tag{8}$$
$$g_3(\epsilon)(x, y, t, u) = (x, y + \epsilon, t, u), \quad \text{(Y Shift)} \tag{9}$$

To measure the effect of augmentations on MAE reconstruction performance, we consider augmenting samples with different probabilities (augmentation ratio), with 0 corresponding to never augmenting samples and 1 corresponding to always augmenting samples. We report results on the 1D KdV-Burgers equation in Table 7. Lie augmentations consistently improve masked reconstruction on validation samples, even at low augmentation ratios. We believe that at high augmentation ratios, the samples may be too different than the validation data distribution. Additionally, the magnitude of the parameter ϵ can also modulate the strength of the augmentation; this is an additional consideration, but in practice we find that smaller values tend to work well.

Table 7: Lie augmentation ablation studies

Aug Ratio	Error
0.00	2.48e-03
0.25	1.24e-03
0.50	1.17e-03
0.75	1.27e-03
1.00	1.37e-03

A.3 Multi-resolution Details

Table 8: Multiresolution ablation studies

Strategy	Error
None	4.40e-03
Pad	3.81e-03
Interp.	3.23e-03
Token	1.89e-03

To handle PDE inputs of varying discretizations, we consider the setup where inputs of shape (n_t, n_x) or (n_t, n_x, n_y) can vary along their spatial dimension (n_x, n_y) by multiples of the patch size (p_x, p_y). During pretraining, snapshots in time, or time windows, are sampled with length n_t, which discretizes the PDE trajectories to a common time dimension. This is the same setup during fine-tuning; PDE features are predicted from an initial time window, and time-stepping/resolution experiments use temporal bundling (8).

Given a fixed patch size, varying spatial resolutions will affect the input sequence length to the ViT. This can be an issue for vanilla positional encodings, as the same position in the input sequence can now represent different spatial coordinates. This is especially pronounced in 2D; flattening two spatial dimensions results in positional information needing to wrap around inputs, and as a result, the same position can represent very different points in space when given different discretizations. To mitigate this issue, we propose strategies for embedding spatial resolution information into the ViT backbone.

The simplest strategy would be to pad the inputs to the maximum sequence length (*Pad*). However, this does not address the fact that positional information becomes overloaded and only provides information about input length. To address this, we consider interpolating positional embeddings from the maximum sequence length to variable sequence lengths (*Interp.*). This would allow tokens to receive approximate positions; however, this would not be accurate for irregular grids. Lastly, to adapt to arbitrary grids, we consider using an embedder network, which could be a 1D or 2D CNN (depending on the spatial dimension), to project the spatial coordinate grid to a token (*Token*). Validation MSE errors after training on varying resolutions of the 1D KdV-Burgers equation are reported in Table 8. We observe consistent improvement by using more sophisticated methods for embedding spatial discretizations.

In 1D, multiresolution pretraining is significantly easier to learn, so we use the interpolation strategy for simplicity. In 2D, the use of tokenized spatial grid information becomes more important, so we use this strategy for 2D multiresolution experiments. The hyperparameters and training costs are the same as those presented in Appendix A.1.

B PDE Details

In this section we detail the various PDEs considered in the paper. For convenience, we copy the equations from Section 4.1 here as well.

B.1 1D PDEs

1D KdV-Burgers equation The KdV-Burgers equation is a diverse equation that contains diffusive, shock formation, and dispersive behaviors:

$$\partial_t u + \alpha u \partial_x u - \beta \partial_{xx} u + \gamma \partial_{xxx} u = \delta(t, x) \tag{10}$$

We use a varied form that contains the Heat, Burgers, and KdV equations as corner cases modulated by the coefficient values (α, β, γ), and is solved numerically according to the setup in Brandstetter et al. (8) with periodic boundary conditions. Both initial conditions and forcing terms are generated from the periodic function $\delta(x, t)$, where we uniformly sample $A_j \in [-0.5, 0.5], \omega_j \in [-0.4, 0.4], l_j \in \{1, 2, 3\}, \phi_j \in [0, 2\pi)$ while fixing $J = 5, L = 16$.

$$\delta(t, x) = \sum_{j=1}^{J} A_j sin(\omega_j t + 2\pi l_j x / L + \phi_j), \quad u(0, x) = \delta(0, x) \tag{11}$$

To generate diverse PDE samples, the coefficients are sampled uniformly from $\alpha \in [0, 3], \beta \in [0, 0.4], \gamma \in [0, 1]$. Furthermore, samples are generated with a discretization $(n_t, n_x) = (250, 100)$ on an interval $x = [0, 16]$ from $t = 0$ to $t = 2$.

1D Heat and Burgers equations The 1D Heat and Burgers equations describe the common phenomena of diffusion and shock formation:

$$\partial_t u - \nu \partial_{xx} u = \delta(t, x), \quad \text{Heat} \tag{12}$$
$$\partial_t u + u \partial_x u = \delta(t, x), \quad \text{Burgers} \tag{13}$$

We solve the equations with the same periodic BCs, initial conditions, and forcing function setup and as the 1D KdV-Burgers equation (8), but by setting the appropriate coefficient values. Specifically, from Equation 10 we uniformly sample $\nu = \beta \in [0.1, 0.8]$ for the Heat equation and fix $\alpha = 0.5, \beta = 0$ to model the inviscid Burgers equation. These equations are also solved with a discretization $(n_t, n_x) = (250, 100)$ on an interval $x = [0, 16]$ from $t = 0$ to $t = 2$.

To generate data for the Heat equation that enforces Dirichlet or Neumann boundary conditions, we write the Heat equation in its variational form after an implicit Euler discretization:

$$\int_\Omega (uv + \Delta t \nabla u \cdot \nabla v) dx = \int_\Omega (u^n + \Delta t f^{n+1}) v dx \tag{14}$$

This formulation can be solved using FEniCS (71; 72). To simplify the boundary value problem, we set the forcing term $f = 0$. Furthermore, we set $[u(x = 0) = u(x = L) = 0]$ to enforce Dirichlet BCs, and $[\partial_x u(x = 0) = \partial_x u(x = L) = 0]$ to enforce Neumann BCs; however, in both of these cases, the initial conditions in Equation 11 need to be modified to respect the new BCs.

$$u(0, x) = \sum_{j=1}^{J} A_j sin(2\pi l_j x/L + \phi_j), \quad \text{Dirichlet ICs} \tag{15}$$

$$u(0, x) = \sum_{j=1}^{J} A_j cos(2\pi l_j x/L + \phi_j), \quad \text{Neumann ICs} \tag{16}$$

In both equations, we uniformly sample $A_j \in [-0.5, 0.5], l_j \in \{1, 2, 3\}, \phi_j \in \{0, \pi\}$ while fixing $J = 5, L = 16$.

1D Advection Equation The 1D Advection equation is a simple model of transport phenomena:

$$\partial_t u + c\partial_x u = 0 \tag{17}$$

We solve the Advection equation with periodic BCs using its analytical solution:

$$u(x, t) = u_0(x - ct) \tag{18}$$

Initial conditions are generated according to Equation 11 and the wave speed is uniformly sampled from $c \in [0.1, 5]$. The solution domain and discretization are the same as previous cases, with $(n_t, n_x) = (250, 100)$, $x = [0, 16]$, and time ranging from $t = 0$ to $t = 2$.

1D Wave Equation The 1D Wave equation is a common equation that describes wave propagation across many disciplines:

$$\partial_{tt} u - c^2 \partial_{xx} u = 0 \tag{19}$$

We solve the equation according to the setup in Brandstetter et al. (8) with Dirichlet ($u(x = 0) = u(x = L) = 0$) and Neumann ($\partial_x u(x = 0) = \partial_x u(x = L) = 0$) BCs, resulting in waves that either bounce or reflect off boundaries. The wave speed is fixed at $c = 2$, and the initial condition is a Gaussian pulse with unit amplitude and with its peak randomly sampled on the spatial domain. Lastly, the equation is solved from $t = 0$ to $t = 100$ on the interval $x = [-8, 8]$ with a discretization $(n_t, n_x) = (250, 100)$.

1D Kuramoto-Sivashinsky Equation The 1D KS equation is a chaotic fourth-order PDE with rich, complex dynamics:

$$\partial_t u + u\partial_x u + \nu \partial_{xx} u + \partial_{xxxx} u = 0 \tag{20}$$

We numerically solve this equation with periodic BCs according to Brandstetter et al. (32), however, the initial conditions are modified from the original work to match Equation 11. Following the data setup proposed by Lippe et al. (61), we additionally modify the original work to uniformly sample $\nu \in [0.75, 1.25]$ to vary the second-order term in the KS equation. Furthermore, due to the unique dynamics of the KS equation, we solve the PDE from $t = 0$ to $t = 100$ on the interval $x = [0, 64]$ with a discretization of $(n_t, n_x) = (100, 100)$.

B.2 2D PDEs

2D Heat, Advection, and Burgers Equations The 2D Heat, Advection, and Burgers are simple models of diffusion, transport, and shock phenomena in 2D:

$$\partial_t u - \nu \nabla^2 u = 0, \quad \text{Heat} \tag{21}$$

$$\partial_t u + \mathbf{c} \cdot \nabla u = 0, \quad \text{Advection} \tag{22}$$

$$\partial_t u + u(\mathbf{c} \cdot \nabla u) - \nu \nabla^2 u = 0 \quad \text{Burgers} \tag{23}$$

We use a 1st-order central differences scheme to discretize the diffusive term ($\nabla^2 u$), and a 1st-order upwinding scheme to discretize the nonlinear convective term ($u \nabla u$). Additionally, the 2D Advection equation is solved with its analytical solution:

$$u(x, y, t) = u_0(x - c_x t, y - c_y t) \tag{24}$$

For the Heat equation, we uniformly sample the $\nu \in [2 \times 10^{-3}, 2 \times 10^{-2}$; for the Advection equation, we uniformly sample $\mathbf{c} = [c_x, c_y] \in [0.1, 2.5]^2$; and for the Burgers equation, we uniformly sample $\nu \in [7.5 \times 10^{-3}, 1.5 \times 10^{-2}$, and $\mathbf{c} = [c_x, c_y] \in [0.5, 1.0]^2$. For all equations, we use periodic BCs and solve on a grid $(n_t, n_x, n_y) = (100, 64, 64)$ on a solution domain $(x, y) = [-1, 1]^2$ from $t = 0$ to $t = 2$. Lastly, initial conditions are generated from:

$$u(0, x, y) = \sum_{j=1}^{J} A_j sin(2\pi l_{xj} x / L + 2\pi l_{yj} y / L + \phi_j) \tag{25}$$

Initial condition parameters are uniformly sampled from $A_j \in [-0.5, 0.5], \omega_j \in [-0.4, 0.4], l_{xj} \in \{1, 2, 3\}, l_{yj} \in \{1, 2, 3\}, \phi_j \in [0, 2\pi)$ while fixing $J = 5, L = 2$.

2D Incompressible Navier-Stokes Equations The 2D incompressible Navier-Stokes equations are an important set of PDEs that describe viscous fluid flow. Following the setup in Li et al. (6), we formulate the NS equations in vorticity form with variable viscosity and forcing function amplitude:

$$\partial_t \omega + \mathbf{u} \cdot \nabla \omega - \nu \nabla^2 \omega = f(x, y), \quad \nabla \cdot \mathbf{u} = 0, \quad \nabla \times \mathbf{u} = \omega \tag{26}$$

$$f(x, y) = A(sin(2\pi(x + y)) + cos(2\pi(x + y))) \tag{27}$$

The solution is solved on a grid $(n_t, n_x, n_y) = (100, 64, 64)$ on a solution domain $(x, y) = [0, 1]^2$ from $t = 0$ to $t = 25$. PDE parameters are uniformly sampled from $\nu \in \{\{1, 2, 3, 4, 5, 6, 7, 8, 9\} \times 10^{-\{6,7,8,9\}}\}$ and $A \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\} \times 10^{-3}$. Lastly, initial conditions ω_0 are sampled according to Li et al. (6) from a Gaussian random field.

C Additional MAE Examples

C.1 Additional 1D MAE Predictions

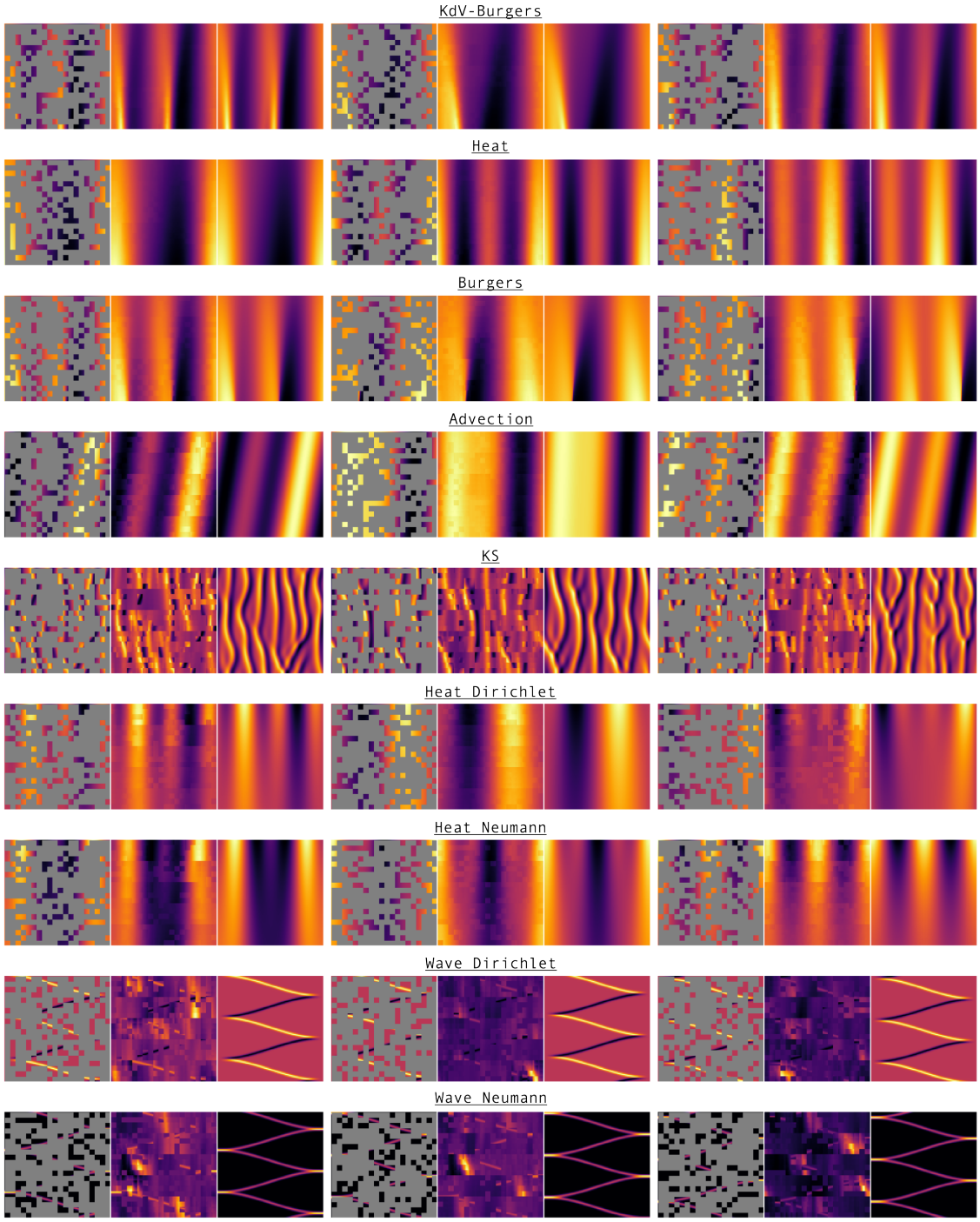

Figure 6: Additional 1D MAE Reconstruction examples after pretraining on the 1D KdV-Burgers equation. Each triplet is shown with the masked sample (Left), MAE reconstruction (Middle), and ground truth PDE (right). We include additional reconstructions of unseen boundary conditions for the Heat and Wave equations.

C.2 Additional 2D MAE Predictions

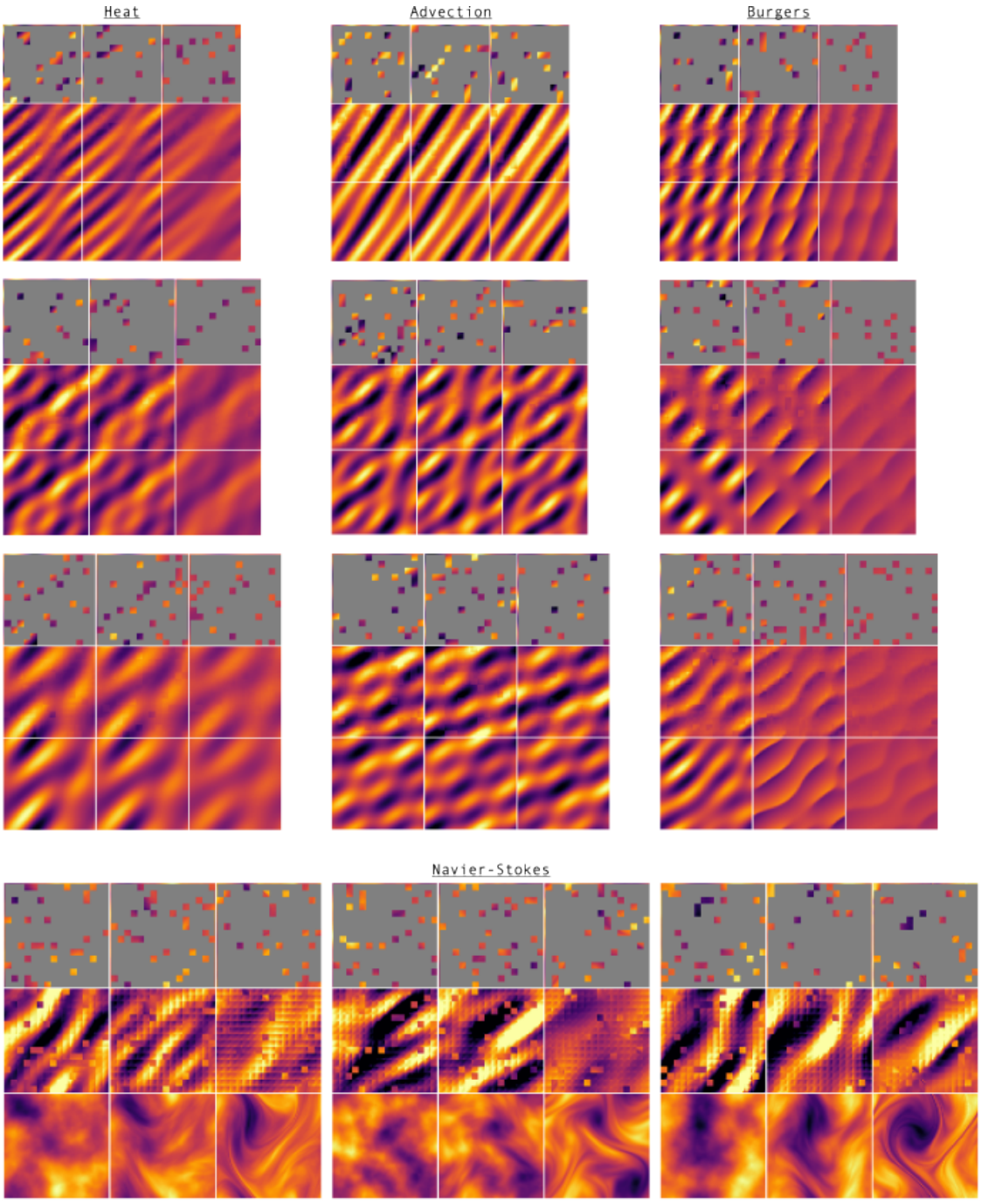

Figure 7: Additional 2D MAE Reconstruction examples after pretraining on the 2D Heat, Advection, and Burgers Equations. Each sample is shown with the masked sample (Top), MAE reconstruction (Middle), and ground truth PDE (Bottom). We include sample MAE predictions at variable resolutions for the 2D Heat, Advection, and Burgers equations; the lowest resolution (top) is $(48, 48)$, the medium resolution (middle) is $(52, 52)$, and the high resolution (bottom) is $(56, 56)$ We include additional reconstructions of the incompressible NS equations at the native resolution $(64, 64)$.

C.3 2D Smoke Buoyancy Predictions

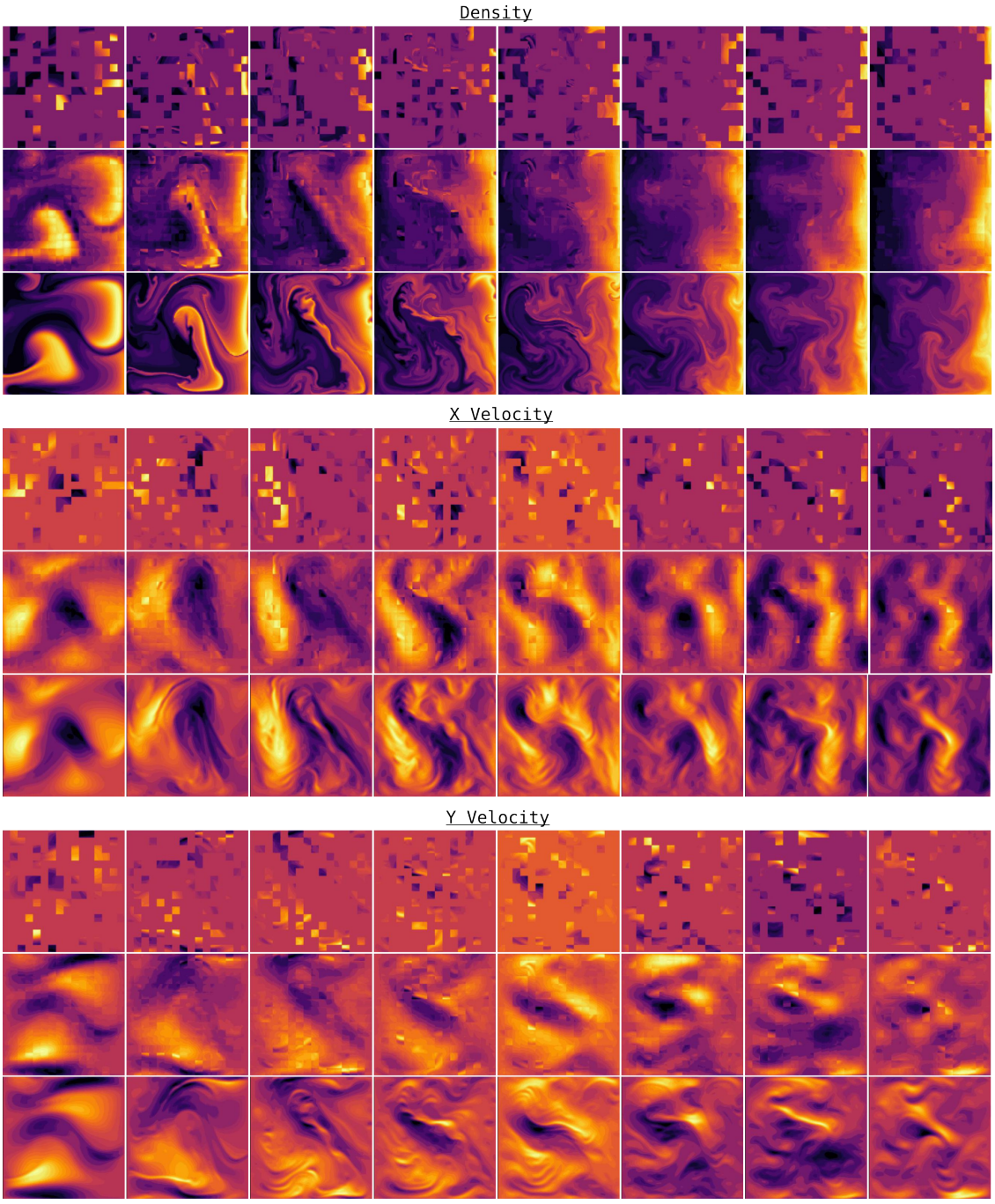

Figure 8: MAE validation reconstructions after training on 2D Navier-Stokes data with variable buoyancy factors (19). The MAE model is trained on a resolution of $(n_t, n_x, n_y) = (56, 128, 128)$ with three data channels (ρ, v_x, v_y) and a masking ratio of 0.75. Triplets are shown with the masked input (top), MAE reconstruction (middle), and ground truth (bottom), with the top, middle, and bottom triplets displaying density, X velocity, and Y velocity. The complex dynamics is challenging; indeed, many of the fine details are lost in the MAE reconstruction. We train with a larger model (45M params) and patch size $(2, 8, 8)$, which takes around 9 hours on a NVIDIA RTX A6000 GPU.

Table 9: Hyperparameters for architectures used for time-stepping and super-resolution.

(a) FNO Hyperparameters

Parameter	1D/2D
Modes	24/12
Width	64/48
# Layers	4
Cond. dim	32
Conditioning	Add
Init lr	8e-4
# Params	1M/5M

(b) Unet Hyperparameters

Parameter	1D/2D
Hidden channels	16
Channel mults.	(1, 2, 4)
Cond. dim	32
Conditioning	AdaGN
Init lr	8e-4
# Params	1M/2M

(c) Resnet Hyperparameters

Parameter	1D/2D
Hidden channels	64
# Blocks	4
Cond. dim	32
Conditioning	Add
Init lr	8e-4
# Params	1M/3M

D Training Details

D.1 PDE Feature Prediction

The baseline encoder is a ViT with the same model size and architecture as the pretrained MAE encoder. Regression and classification are performed using a CLS token and projecting the CLS embedding to the number of prediction features through a simple MLP. When the MAE encoder is frozen, only this MLP head receives gradient updates. For extremely small coefficient values ($< 1 \times 10^{-2}$), the coefficient labels are scaled to the interval $[0, 1]$. Regression and classification tasks tend to be very quick, on the order of minutes for 1D and 2D PDEs.

D.2 Time-stepping

Time-stepping is performed autoregressively by predicting multiple timesteps simultaneously to reduce error accumulation (8). Furthermore, the pushforward trick (8) is implemented. This adds model noise to inputs during training by making a prediction of a future timestep and using that prediction as a noised input the model; importantly gradients are not calculated for the initial pass. Hyperparameters used during time-stepping are presented in Table 9. For each seed, an experiment in 1D takes around 30 minutes, or 45 minutes if fine-tuning an encoder on a NVIDIA RTX 2080 Ti; an experiment in 2D takes around 15 minutes, or 2 hours if fine-tuning an encoder on a NVIDIA RTX A6000.

D.3 Super-resolution

Super-resolution is performed in multiple time steps simultaneously throughout the low-resolution PDE trajectory. We present a detailed schematic of the SR pipeline in Figure 9. Since the encoder operates on low-resolution physics, we use the multi-resolution MAE in 1D. In 2D, the lowest resolution used during multi-resolution training is too large, so a separate MAE is trained on 32×32 data. The discretization inversion operator D^{-1} used is linear interpolation in 1D and bicubic interpolation in 2D. The hyperparameters used for the Resnet encoder, which uses residual dense blocks (68), and FNO operator are reported in Table 9. Super-resolution tasks tend to be fast, on the order of minutes for 1D and 2D samples, and are generally limited by dataloading and memory considerations.

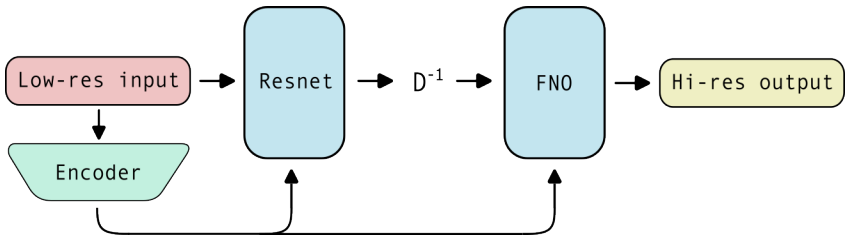

Figure 9: Conditional super-resolution pipeline.

E Additional Results and Statistical Significance

E.1 Feature Prediction

Table 10: 1D PDE coefficient regression. Validation errors are given as RMSE $\times 10^{-2}$. The mean error and standard deviation are calculated over five seeds; error bars are reported as one standard deviation above or below the mean. Lowest errors that are statistically significant are bolded.

Model	KdV-B	Heat	Adv	KS
Base	3.454 ±0.131	0.834 ±0.041	0.241 ±0.051	0.354 ±0.104
MAE_f	1.334 ±0.036	0.677 ±0.016	0.551 ±0.03	0.368 ±0.02
MAE	**0.905 ±0.059**	**0.505 ±0.065**	0.244 ±0.064	**0.156 ±0.023**

Table 11: 1D PDE feature classification. Validation Errors are given as X-Ent $\times 10^{-4}$. The mean error and standard deviation are calculated over five seeds; error bars are reported as one standard deviation above or below the mean. Lowest errors that are statistically significant are bolded.

Model	$Wave_{BC}$	$Heat_{BC}$	PDEs	Res.
Base	0.022 ±0.004	0.123 ±0.108	0.355 ±0.276	64.52 ±3.475
MAE_f	1.817 ±1.175	1.164 ±0.838	0.174 ±0.064	63.34 ±3.71
MAE	0.053 ±0.048	0.018 ±0.002	**0.025 ±0.013**	**28.322 ±23.351**

Detailed results for 1D PDE feature prediction tasks are reported in Tables 10 and 11. For 1D tasks, certain experiments have high variance; we hypothesize that this is due to the fact that each seed samples a random dataset of 2000 samples from a much larger dataset. This would make some seeds easier to regress/classify than others, but within each seed the models follow trends consistent with the mean statistics. Furthermore, the magnitude of the X-Ent error is very small, leading to high variations after the model has learned most of the relevant features.

Table 12: 2D PDE coefficient regression and feature classification. Validation errors are reported as RMSE $\times 10^{-1}$ for regression tasks and X-Ent $\times 10^{-1}$ for classification tasks. The mean error and standard deviation are calculated over five seeds; error bars are reported as one standard deviation above or below the mean. Lowest errors that are statistically significant are bolded.

Model	Heat	Adv	Burgers
Base	0.084 ±0.014	0.506 ±0.009	0.682 ±0.037
MAE_f	0.232 ±0.01	0.54 ±0.015	0.606 ±0.012
MAE	**0.062 ±0.003**	0.507 ±0.006	**0.409 ±0.008**
Model	Combined	NS	Res.
Base	0.320 ±0.007	0.748 ±0.005	0.694 ±0.174
MAE_f	0.384 ±0.009	0.709 ±0.01	0.636 ±0.061
MAE	**0.265 ±0.007**	**0.594 ±0.038**	**0.005 ±0.002**

Detailed results for 2D feature prediction tasks are reported in Table 12. The 2D results tend to be more consistent and have lower variance, since a fixed dataset was used for each seed and only the shuffling is changed.

E.2 Time-stepping

Table 13: Conditional 1D PDE time-stepping. Validation errors are reported as normalized L2 loss summed over all PDE timesteps. The mean error and standard deviation are calculated over five seeds; error bars are reported as one standard deviation above or below the mean. Lowest errors that are statistically significant are bolded.

Model	KdV-B	Heat	Burgers	Adv
FNO	1.132 ±0.037	0.671 ±0.039	1.094 ±0.060	0.437 ±0.052
FNO-Enc	1.041 ±0.029	0.644 ±0.038	1.129 ±0.062	0.347 ±0.074
FNO-MAE$_\text{f}$	1.077 ±0.060	0.655 ±0.008	1.121 ±0.051	0.431 ±0.054
FNO-MAE	1.060 ±0.032	0.643 ±0.035	**0.952 ±0.038**	0.294 ±0.033
FNO-Lin	**0.936 ±0.029**	0.75 ±0.062	N/A	**0.204 ±0.019**
Unet	0.872 ±0.069	0.420 ±0.021	0.649 ±0.052	0.194 ±0.059
Unet-Enc	0.834 ±0.043	0.395 ±0.021	0.582 ±0.032	0.224 ±0.057
Unet-MAE$_\text{f}$	0.833 ±0.038	0.425 ±0.012	0.582 ±0.017	0.177 ±0.012
Unet-MAE	0.795 ±0.040	**0.363 ±0.010**	0.546 ±0.024	0.21 ±0.047
Unet-Lin	**0.659 ±0.045**	0.445 ±0.008	N/A	0.166 ±0.032
Model	KS	Heat$_\text{BC}$	Wave$_\text{BC}$	Res.
FNO	0.83 ±0.028	0.147 ±0.015	2.408 ±0.848	1.141 ±0.021
FNO-Enc	0.82 ±0.082	0.133 ±0.019	2.012 ±1.194	1.038 ±0.037
FNO-MAE$_\text{f}$	0.821 ±0.088	0.135 ±0.013	1.747 ±0.665	1.018 ±0.13
FNO-MAE	0.812 ±0.061	0.148 ±0.015	1.846 ±1.885	1.070 ±0.011
FNO-Lin	0.757 ±0.077	0.132 ±0.020	1.454 ±0.450	**0.899 ±0.01**
Unet	1.333 ±0.068	0.747 ±0.043	4.249 ±2.296	0.766 ±0.083
Unet-Enc	1.203 ±0.102	0.691 ±0.046	4.902 ±1.935	0.739 ±0.088
Unet-MAE$_\text{f}$	1.241 ±0.055	0.699 ±0.023	4.734 ±2.135	0.688 ±0.077
Unet-MAE	1.125 ±0.029	0.659 ±0.047	5.157 ±1.760	0.683 ±0.087
Unet-Lin	1.172 ±0.039	0.717 ±0.027	4.727 ±2.093	0.573 ±0.095

Following the main paper, we introduce two conditional benchmarks. We evaluate a randomly initialized and fine-tuned ViT encoder with the same architecture as the MAE encoder (-Enc), as well as a linear encoder that embeds the ground-truth PDE parameters as the conditioning information (-Lin).

In 1D, time-stepping results tend to have high variance; however, overall trends are still consistent with those reported in the main body. The variance is likely attributed to variations in the dataset for each seed; each seed samples a different set of 2000 samples from a larger PDE dataset, and as a result, some data splits may be easier than others. This results in the variance being high across seeds, however, within a seed (i.e. within a dataset), model performance closely follows trends consistent with the mean statistics.

Table 14: Conditional 2D PDE time-stepping. Validation errors are reported as normalized L2 loss summed over all PDE timesteps. The mean error and standard deviation are calculated over three seeds; error bars are reported as one standard deviation above or below the mean. Lowest errors that are statistically significant are bolded.

Model	Heat	Adv	Burgers
FNO	0.427 ±0.006	2.301 ±0.094	0.417 ±0.063
FNO-Enc	0.152 ±0.013	1.909 ±0.399	0.241 ±0.032
FNO-MAE$_f$	0.233 ±0.028	1.179 ±0.036	0.252 ±0.012
FNO-MAE	0.128 ±0.008	1.135 ±0.121	0.198 ±0.009
FNO-Lin	0.118 ±0.005	2.531 ±0.013	**0.149 ±0.036**
Unet	0.147 ±0.031	1.795 ±0.105	0.226 ±0.018
Unet-Enc	0.132 ±0.040	1.604 ±0.164	0.218 ±0.02
Unet-MAE$_f$	0.136 ±0.009	1.804 ±0.066	0.229 ±0.017
Unet-MAE	0.116 ±0.031	**1.23 ±0.161**	**0.186 ±0.011**
Unet-Lin	0.153 ±0.043	2.571 ±0.011	0.215 ±0.006
Model	Combined	NS	Res.
FNO	0.978 ±0.055	0.466 ±0.014	1.006 ±0.02
FNO-Enc	0.767 ±0.028	0.514 ±0.123	0.709 ±0.055
FNO-MAE$_f$	0.607 ±0.019	0.59 ±0.107	0.701 ±0.051
FNO-MAE	**0.494 ±0.043**	0.477 ±0.029	**0.499 ±0.024**
FNO-Lin	0.977 ±0.021	0.445 ±0.026	0.986 ±0.015
Unet	0.835 ±0.067	0.713 ±0.005	0.908 ±0.061
Unet-Enc	0.791 ±0.061	0.695 ±0.027	0.971 ±0.023
Unet-MAE$_f$	0.761 ±0.051	0.669 ±0.031	0.861 ±0.028
Unet-MAE	**0.669 ±0.015**	0.692 ±0.039	**0.676 ±0.064**
Unet-Lin	1.013 ±0.03	**0.635 ±0.002**	1.098 ±0.026

In 2D, time-stepping results have much lower variance; this is likely due to the fact that each seed uses the same dataset, with only the shuffling changing. Furthermore, the linear benchmark is less effective; in most experiments a learned encoding can outperform ground-truth PDE parameters, especially when predicting a combined or multi-resolution dataset of PDEs.

E.3 Comparison to Contrastive Learning with Lie Augmentations

Table 15: We compare our approach to a contrastive self-supervised approach. After training a masked and contrastive encoder on the KdV-B pretraining set, we compare conditioning an FNO backbone to time-step different downstream 1D PDEs. The MAE encoder shows comparable performance within the pretraining set (KdV-B), but has better generalization behavior to unseen PDEs. Validation errors are reported as normalized L2 loss summed over all PDE timesteps

Model	KdV-B	Heat	Burgers	Adv
FNO	1.506	0.827	1.386	0.567
FNO-Contrastive	**1.171**	0.918	0.916	0.555
FNO-MAE	1.183	**0.721**	**0.831**	**0.299**

E.4 Super-resolution

Table 16: Conditional 1D super-resolution. Validation errors are reported as RMSE $\times 10^{-1}$ summed over all PDE timesteps. The mean error and standard deviation are calculated over five seeds; error bars are reported as one standard deviation above or below the mean. Lowest errors that are statistically significant are bolded.

Model	KdV-B	Heat	Burgers	Adv
SR	0.520 ±0.021	0.203 ±0.011	0.881 ±0.062	0.223 ±0.004
SR-Enc	0.489 ±0.022	0.166 ±0.021	0.642 ±0.074	0.202 ±0.003
SR-MAE$_f$	0.481 ±0.039	0.173 ±0.015	0.691 ±0.027	0.169 ±0.032
SR-MAE	0.475 ±0.018	0.151 ±0.028	0.676 ±0.060	0.194 ±0.016
SR-Lin	0.484 ±0.017	0.131 ±0.017	N/A	**0.133 ±0.013**
Model	KS	Heat$_{BC}$	Wave$_{BC}$	
SR	2.673 ±0.101	0.210 ±0.016	0.516 ±0.015	
SR-Enc	2.585 ±0.062	0.174 ±0.01	0.373 ±0.022	
SR-MAE$_f$	2.460 ±0.056	0.204 ±0.015	0.376 ±0.032	
SR-MAE	2.422 ±0.052	0.170 ±0.019	0.349 ±0.022	
SR-Lin	2.517 ±0.038	0.177 ±0.027	0.451 ±0.014	

Differences between benchmarks for 1D super-resolution tend to be incremental. Despite this, using a frozen MAE encoding remains a simple method to improve performance with a negligible training cost. In general, super-resolution for 1D PDEs is a relatively easy task, and changes in model architecture do not significantly affect results.

Table 17: Conditional 2D super-resolution. Validation errors are reported as RMSE $\times 10^{-1}$ summed over all PDE timesteps. he mean error and standard deviation are calculated over three seeds; error bars are reported as one standard deviation above or below the mean. Lowest errors that are statistically significant are bolded.

Model	Heat	Adv	Burgers
SR	0.175 ±0.023	2.014 ±0.205	0.295 ±0.018
SR-Enc	0.152 ±0.01	0.804 ±0.052	0.252 ±0.047
SR-MAE$_f$	0.159 ±0.007	0.659 ±0.089	0.264 ±0.05
SR-MAE	0.152 ±0.004	0.639 ±0.125	0.253 ±0.017
SR-Lin	0.167 ±0.015	2.016 ±0.015	0.263 ±0.021
Model	Combined	NS	
SR	0.534 ±0.037	0.326 ±0.039	
SR-Enc	0.49 ±0.016	0.364 ±0.006	
SR-MAE$_f$	0.407 ±0.002	0.347 ±0.007	
SR-MAE	0.472 ±0.005	0.337 ±0.012	
SR-Lin	1.235 ±0.032	0.366 ±0.003	

In 2D, the general trend remains similar to 1D results; clear model choices for super-resolution are not apparent. Despite this, using a frozen MAE encoding often outperforms the linear benchmark; this can be a good way to boost model performance without additional training cost when PDE samples are within the pretraining distribution.