LATE FUSION NEURAL OPERATORS FOR PARAMETERIZED PARTIAL DIFFERENTIAL EQUATIONS

Eva van Tegelen[1,2], **Taniya Kapoor**[1], **George A. K. van Voorn**[2], **Ioannis N. Athanasiadis**[1]
[1]Artifical Intelligence Group, [2]Biometris
Wageningen University and Research
Wageningen, The Netherlands

ABSTRACT

Developing models that can accurately predict PDE behaviour for unseen parameter values is crucial for achieving robust and reliable generalization in scientific and engineering applications. In practice, variations in physical parameters induce distribution shifts between training and prediction regimes, making extrapolation a central challenge. As a result, the way parameters are incorporated into neural operator models plays a key role in their ability to generalize, particularly when state and parameter representations are entangled. In this work, we introduce a novel architecture for parameterized PDEs, that employs a late fusion neural operator architecture to improve performance inside and outside the training distribution. Our approach combines neural operators for learning latent state representations with sparse regression to incorporate parameter structure in an interpretable manner. By explicitly separating the learning of state dynamics from parameter dependence, the proposed Late Fusion Operator enables controlled and interpretable incorporation of parameter effects. Across a range of benchmark PDEs, the proposed Late Fusion Operator consistently outperforms existing approaches (best-performing in approximately 75% of experiments, with 32–86% improvement compared to the second-best method), demonstrating strong generalization across both in-domain and out-of-domain parameter regimes.

1 INTRODUCTION

Many physical systems can be represented by partial differential equations (PDEs) that model their spatial and temporal evolution, and whose dynamics depend on underlying parameters such as material properties (Evans, 2022; Pletcher et al., 2012; Gurtin et al., 2010) and environmental conditions (Vallis, 2017; Borgogno et al., 2009). In practice, parameters vary across operating regimes or environments, while data are typically available only for a limited subset of the parameter space due to physical, logistical, or economic constraints. From a machine-learning perspective, variations in physical parameters effectively change the distribution of the input data between training and prediction, a phenomenon commonly referred to as covariate shift, which poses a major challenge for data-driven models (Tamang et al., 2025; Bickel et al., 2009). Therefore, developing models that can accurately predict system behaviour for unseen parameter values is crucial for robust and reliable generalization in scientific and engineering applications (Li et al., 2025a).

Recent advances in machine learning have led to powerful frameworks for learning mappings between function spaces from data, enabling efficient data-driven solutions of PDEs(Azizzadenesheli et al., 2024; Kovachki et al., 2023; Fanaskov & Oseledets, 2023). Neural operator architectures, such as FNO (Li et al.) and DeepONet (Lu et al., 2019), learn solution operators that generalize across different initial and boundary conditions, while methods like PDE-Refiner (Lippe et al., 2023) incorporate iterative refinement strategies for improved long-term accuracy. These approaches can, in principle, be extended to parameterized PDEs by incorporating system parameters as additional input dimensions. However, this naive concatenation can encourage entangled parameter–state representations, which we hypothesize might hinder extrapolation performance.

Several recent works have focused specifically on improving parameter extrapolation in PDE learning. Li et al. (2025b) propose a physics-guided invariant learning framework that combines a

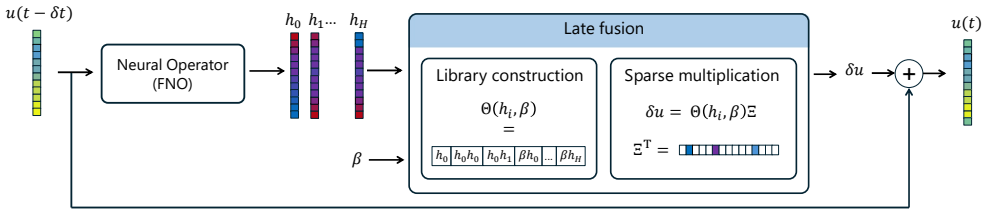

Figure 1: Late Fusion Operator model structure. The current state $\boldsymbol{u}(t - \delta t)$ is given as input to the neural operator, which outputs a set of hidden states h_i. These hidden states are taken together with the input parameters $\boldsymbol{\beta}$ to construct a library. This library vector Θ is multiplied with a sparse matrix Ξ to give the rate of change $\boldsymbol{\delta u}$ which is added to the current state to obtain the next state.

mixture of operator experts with parameter fusion to enhance out-of-distribution generalization. Kassaï Koupaï et al. (2024) introduce an adaptive conditioning approach that enables neural solvers to adapt to unseen parameters through fine-tuning on short time series in new environments. Similarly, Takamoto et al. (2023) leverage parameter-guided channel attention to integrate parameter information directly into the model and improve temporal prediction. Together, these approaches highlight the growing research interest in designing architectures and conditioning mechanisms that explicitly target parameterized PDEs. However, extrapolating to unseen parameter regimes is still a challenging and open problem (Li et al., 2024; Gulrajani & Lopez-Paz, 2020), leaving room for methods that better disentangle parameter and state dynamics.

In this work, we focus specifically on the challenge of in- and out-of-distribution parameter extrapolation in a strictly one-shot framework, which means predictions must be made from a single initial condition without access to past trajectories. We assume no prior knowledge of the governing equations, including their functional form or differential operators, and learn the dynamics entirely from data. When predicting across new parameter regimes, past trajectories may be limited or unavailable, making this setting a practical approach for extrapolation. While previous approaches often rely on temporal history, such as sliding windows with multiple time step inputs (Li et al.), or require test-time fine-tuning to adapt to new environments (Kassaï Koupaï et al., 2024), here we focus on predicting system dynamics solely from the initial condition and known PDE parameters.

To achieve this goal, we introduce a novel late fusion architecture that processes the system parameters separately from the state variables and only combines them at a later stage in the network. This design draws inspiration from numerical PDE solvers, where state variables are treated separately from physical coefficients and are later merged(Ames, 2014).

2 Late Fusion Operator for parameterized PDEs

2.1 Problem set-up

Following the notation of Brandstetter et al. (2022), we consider PDEs defined over a time interval $t \in [0, T]$ and over spatial dimensions $\boldsymbol{x} = [x_1, ..., x_D]^T \in \mathbb{X}$. Furthermore we introduce a parameter vector $\boldsymbol{\beta} \in \mathbb{P} \subseteq \mathbb{R}^P$ which characterizes the system dynamics. We assume the system state $\boldsymbol{u} : [0, T] \times \mathbb{X} \to \mathbb{R}^V$ is governed by a parameterized PDE of the form:

$$\partial_t \boldsymbol{u}(t, \boldsymbol{x}) = F\left(t, \boldsymbol{x}, \boldsymbol{u}, \partial_{\boldsymbol{x}} \boldsymbol{u}, \partial_{\boldsymbol{xx}} \boldsymbol{u}, \ldots; \boldsymbol{\beta}\right), \tag{1}$$

$$\boldsymbol{u}(0, \boldsymbol{x}) = \boldsymbol{u}^0(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{X}, \tag{2}$$

$$B[\boldsymbol{u}](t, \boldsymbol{x}) = 0, \quad \boldsymbol{x} \in \partial\mathbb{X}, \tag{3}$$

where $F(\cdot; \boldsymbol{\beta})$ is a non-linear differential operator parameterized by $\boldsymbol{\beta}$. Here, $\partial_t \boldsymbol{u}$ is the partial derivative of the solution u with respect to time and $\partial_{\boldsymbol{x}} \boldsymbol{u}, \partial_{\boldsymbol{xx}} \boldsymbol{u}$,... denote the partial derivatives of $\boldsymbol{u}$ with respect to space. $\boldsymbol{u}^0(\boldsymbol{x})$ denotes the initial condition, and $B[\boldsymbol{u}](t, \boldsymbol{x})$ define the boundary conditions on the domain boundary $\partial\mathbb{X}$.

We assume access to a dataset consisting of multiple solution trajectories of the parameterized PDE corresponding to different initial conditions and parameter values, where the parameters are assumed to remain constant over time within each trajectory. Specifically, for a finite set of parameters $\{\boldsymbol{\beta}^{(i)}\}_{i=1}^{N} \subset \mathbb{P}$ and initial conditions $\{\boldsymbol{u}^{0,(i)}\}_{i=1}^{N}$, we observe N trajectories

$$\boldsymbol{u}^{(i)}(t, \boldsymbol{x}) \in \mathbb{R}^{V}, \quad t \in [0, T],\ \boldsymbol{x} \in \mathbb{X},$$

which are discretized in time and space, yielding sequences of spatio-temporal data. The resulting dataset can thus be written as

$$\mathcal{D} = \left\{ \left(\boldsymbol{u}^{0,(i)}, \boldsymbol{\beta}^{(i)}, \boldsymbol{u}^{(i)}(t, \boldsymbol{x}) \right) \right\}_{i=1}^{N}, \tag{4}$$

where each sample represents a full discretized solution trajectory conditioned on its corresponding initial condition and system parameters.

2.2 Late Fusion Operator

For our method, we were inspired by classical numerical solvers for PDEs, which typically decompose time integration into two conceptual steps(Ames, 2014). First, relevant spatial derivatives are approximated numerically from the current state of the system. Second, these numerical derivatives are combined, together with known parameters of the governing equation, within a time-stepping rule to compute the temporal update of the state. In the present setting however, the governing equation is assumed to be unknown, such that neither the functional form of this update rule nor the specific derivatives to be approximated are available.

To address the lack of access to the governing equations, we propose a data-driven framework that mirrors the two-stage structure of classical PDE solvers while remaining fully equation-agnostic. Given the current system state $\boldsymbol{u}(t - \delta t)$, where δt denotes the temporal discretization step, we first apply a neural operator $\mathcal{N}_\theta$, parameterized by trainable weights θ, to obtain a set of latent representations:

$$\{h_j\}_{j=1}^{H} = \mathcal{N}_\theta(\boldsymbol{u}(t - \delta t)), \tag{5}$$

where the hidden states h_j are learned directly from data and can be viewed as latent analogues of intermediate quantities that arise in classical PDE discretizations, such as derivatives or non-linear interaction terms, without imposing any explicit physical interpretation. In a second step, the hidden states are combined with known parameters values $\boldsymbol{\beta}$ in a library of candidate functions, similar to the Sparse Identification of Nonlinear Dynamics (SINDy) framework (Rudy et al., 2017; Brunton et al., 2016), in which governing equations are expressed as linear combinations of functions taken from a predefined library evaluated on the system state. The state increment (residual) $\boldsymbol{\delta u}$ is modelled using the candidate library $\Theta(\boldsymbol{h}, \boldsymbol{\beta})$:

$$\boldsymbol{\delta u} = \Theta(\boldsymbol{h}, \boldsymbol{\beta})\, \Xi, \tag{6}$$

where Ξ is a trainable coefficient matrix whose entries weigh the contributions of the candidate functions in the library $\Theta(\boldsymbol{h}, \boldsymbol{\beta})$. Sparsity is encouraged through regularization during training, allowing the model to select a subset of candidate functions without imposing hard thresholding. The state is then advanced in time according to

$$\boldsymbol{u}(t) = \boldsymbol{u}(t - \delta t) + \boldsymbol{\delta u}. \tag{7}$$

The model is trained using a composite loss function that balances predictive accuracy with sparsity of the late fusion. The first component of the loss calculates the MSE between the predicted state and the observed data, and is defined as

$$\mathcal{L}_{\text{data}} = \frac{1}{N} \sum_{i=1}^{N} \left\| \boldsymbol{u}_{\text{pred}}^{(i)} - \boldsymbol{u}_{\text{true}}^{(i)} \right\|_2^2, \tag{8}$$

where $\boldsymbol{u}_{\text{pred}}^{(i)}$ denotes the predicted state for sample i and $\boldsymbol{u}_{\text{true}}^{(i)}$ the corresponding ground-truth observation. A second loss term enforces sparsity on the coefficient matrix Ξ. Specifically, we employ ℓ_1 regularization,

$$\mathcal{L}_{\text{sparse}} = \|\Xi\|_1, \tag{9}$$

which encourages the model to select only a small subset of candidate functions from the library. The total training objective is then given by:

$$\mathcal{L} = \mathcal{L}_{\text{data}} + \lambda_{\text{sparse}} \mathcal{L}_{\text{sparse}}, \tag{10}$$

where the sparsity coefficient λ_{sparse} controls the trade-off between predictive accuracy and model simplicity. This coefficient is selected via validation (see Appendix B).

3 EXPERIMENTS

To validate the proposed approach and evaluate its in- and out-of-distribution prediction capabilities, we generate synthetic datasets from benchmark PDEs in one and two dimensions: 1D Advection, 1D Burgers, and 2D Gray–Scott. These benchmark PDEs exhibit diverse dynamical behaviours, such as shocks and pattern formation, making them suitable for assessing generalization. For the 1D cases we employed the PDEBench framework (Takamoto et al., 2022) to generate the datasets. As the original PDEBench datasets are not designed to be suitable for systematic evaluation across a wide range of parameter values, we adapted the original generation scripts to sample for a wider range of parameter values and create separate training and test distributions. We will briefly discuss the test-cases below. Table 1 describes the ranges of the parameter values that were used to generate the training and test datasets.

1D Advection: We consider the one-dimensional advection equation:

$$\partial_t u(t,x) = -\beta \partial_x u(t,x), \quad x \in (0,1), t \in (0,2], \tag{11}$$

$$u(0,x) = u_0(x), \quad x \in (0,1), \tag{12}$$

where β is a constant advection speed. We use periodic boundary conditions and initial conditions as described in Takamoto et al. (2022). We generate data for different values of the parameter β. For training and in-domain extrapolation, we randomly sample β from the interval $[0, 0.5]$. Out-of-domain data are generated by sampling β from the interval $[0.5, 1]$. We select the training and test intervals to assess out-of-distribution performance: the test set includes higher advection speeds than seen during training, requiring the model to predict more extreme cases. All trajectories are sampled over the time interval $[0, 0.5]$ with a fixed time step of $\Delta t = 0.05$.

1D Burgers: We consider the one-dimensional Burgers equation:

$$\partial_t u(t,x) = -\partial_x (u^2(t,x)/2) + \nu\pi \partial_{xx} u(t,x), \quad x \in (0,1), t \in (0,2], \tag{13}$$

$$u(0,x) = u_0(x), \quad x \in (0,1), \tag{14}$$

where ν is the diffusion coefficient, which we vary. We use periodic boundary conditions and initial conditions following the Burgers setup in Takamoto et al. (2022). For training and in-domain extrapolation, we randomly sample ν from the interval $[0.01, 0.02]$. Out-of-domain data are generated by sampling ν from the interval $[0, 0.01]$, corresponding to weaker diffusion and therefore sharper shock formation. All trajectories are sampled over the time interval $[0, 0.5]$ with a fixed time step of $\Delta t = 0.005$.

2D Gray-Scott: For the Gray-Scott model we consider the following system of equations (McGough & Riley, 2004):

$$\partial_t u = D_u \Delta u + F(1-u) - uv^2, \qquad x \in (0,1), t \in (0, 2500)], \tag{15}$$

$$\partial_t v = D_v \Delta v - (F+k)v + uv^2, \qquad y \in (0,1), \tag{16}$$

where u and v denote the concentrations of two interacting chemical species. D_u and D_v are the diffusion constants for u and v respectively, F is the feeding rate of u and k is the removal (killing) rate of v. For our simulations we set $D_u = 1.0$, $D_v = 0.5$ and $k = 0.061$. Numerical solutions were generated using a finite-difference scheme with time step $\delta t = 1$, and the resulting trajectories were subsampled by taking every 25th time step. The parameter that we will vary during the simulations is F. All trajectories are sampled over the time interval $[0, 2500]$ with an effective time step of $\delta t = 25$.

We benchmark our method against a diverse set of established neural operators and architectures designed for parameterized generalization. As standard baselines, we employ the Fourier Neural

Table 1: Parameter ranges used for training, in-domain testing, and out-domain testing. For the one-dimensional Advection and Burgers equations, 100 parameter values are sampled from a uniform distribution for each split. For the two-dimensional Gray-Scott equations, parameters are sampled at evenly spaced values. N_{traj} is the number of trajectories in the corresponding dataset split.

	Train	N_{traj}	**Test In-domain**	N_{traj}	**Test Out-domain**	N_{traj}
1D Advection (β)	$U(0, 0.5)$	100	$U(0, 0.5)$	100	$U(0.5, 1)$	100
1D Burgers (ν)	$U(0.01, 0.02)$	100	$U(0.01, 0.02)$	100	$U(0, 0.01)$	100
2D Gray-Scott (F)	$[0.038, \ldots, 0.046]$	10	$[0.0385, \ldots, 0.0455]$	9	$[0.05]$	1

Operator (FNO) (Li et al.) and the robust Convolutional Neural Operator (CNO) (Raoni'c et al., 2023). In both cases, the parameter vector is concatenated as additional input channels alongside the state variables. We also compare against a method that was explicitly designed for parameter extrapolation: CAPE-FNO (Takamoto et al., 2023), which integrates parameter information via a channel-guided attention mechanism. Hyperparameter tuning is discussed in appendix B.

To strictly adhere to the one-shot prediction goal, all models are trained on single-step transitions ($t \to t + \delta t$) and applied auto-regressively during inference to generate the full trajectory from the initial condition.

4 RESULTS

We evaluate all methods on the 1D Advection, 1D Burgers, and 2D Gray-Scott test datasets using the normalized root mean squared error (nRMSE). Table 2 reports results for both in-domain and out-domain evaluations, across short-term and long-term prediction horizons. The in-domain evaluation tests accuracy under conditions similar to training, while out-of-domain evaluation assesses the ability of models to generalize to unseen parameter regimes, which is central to our study. Short-term and long-term horizons capture different aspects of predictive performance: short-term errors reflect immediate prediction accuracy, while long-term errors assess trajectory stability and the accumulation of errors over time. All results are averaged over three random seeds to account for variability in training, with the standard deviation reported.

Across all three equations, prediction error generally increases with the rollout horizon, reflecting the accumulation of errors over longer-term forecasts. This trend is consistently observed for both in-domain and out-of-domain evaluations. Overall error levels are substantially higher in the out-of-domain setting, indicating that generalization to unseen conditions is more difficult.

For the 1D Advection equation, Table 2 shows that our Late Fusion Operator architecture achieves the lowest nRMSE on both the in-domain and out-domain experiments and for both prediction horizons. This observation is consistent with Figure 2, which reports the nRMSE per trajectory on the y-axis with the corresponding advection parameter β on the x-axis. Across the full parameter range, both in-domain and out-domain, Late Fusion Operator exhibits consistently lower errors compared to the other models. In contrast, CNO and FNO show a pronounced degradation in performance when evaluated outside the training regime. While CAPE-FNO is less sensitive to the domain shift, Late Fusion Operator remains more consistent overall. Moreover, the Late Fusion Operator shows very little variation across random seeds, indicating robustness to initialization.

For the 1D Burgers equation, Table 2 shows that while the Late Fusion Operator achieves the best performance on the in-domain test dataset, it is outperformed by CAPE-FNO in the out-domain setting. Figure 3 provides insight into the origin of this performance gap. In particular, Late Fusion Operator exhibits markedly high errors for three specific parameter values, which dominate its overall out-domain nRMSE. Outside of these cases, its performance is comparable to that of CAPE-FNO across the remaining parameter range.

This is also illustrated in Figure 4, which shows the predicted solution for the out-domain Burgers trajectory with $\nu = 0.0052$. The corresponding error plots indicate that both CAPE-FNO and Late Fusion Operator substantially outperform CNO and FNO, in particular by more accurately capturing

Table 2: nRMSE for 1D Advection, 1D Burgers, and 2D Gray-Scott test datasets, evaluated in-domain and out-of-domain. Results are reported as mean nRMSE over all prediction steps up to horizon T steps, averaged over three random seeds with standard deviation also reported. The * indicates that one seed was excluded due to NaN values.

Model	**In-domain**		**Out-domain**	
	10 steps	20 steps	10 steps	20 steps
1D Advection equation				
CNO	3.75e-2±5.96e-3	6.92e-2±1.17e-2	3.12e-1±1.14e-1	5.51e-1±2.12e-1
FNO	3.90e-2±1.76e-2	4.87e-1±4.25e-1	1.53e-1±8.45e-2	6.71e-1±3.68e-1
CAPE-FNO	3.83e-2±4.67e-3	8.16e-2±1.46e-2	1.35e-1±1.04e-2	2.37e-1±2.54e-2
Late Fusion	**6.93e-3±1.58e-3**	**1.14e-2±3.61e-3**	**1.86e-2±2.38e-3**	**3.36e-2±4.34e-3**
1D Burgers equation				
CNO	5.35e-2±5.14e-3	1.10e-1±2.39e-2	1.97e-1±3.86e-2	3.97e-1±1.19e-1
FNO	1.37e-2±1.66e-4	2.30e-2±9.90e-4	9.25e-2±3.72e-2	1.84e-1±8.04e-2
CAPE-FNO	1.42e-2±1.10e-3	2.05e-2±1.91e-3	**5.42e-2±4.62e-3**	**8.37e-2±9.84e-3**
Late Fusion	**9.20e-3±6.63e-4**	**1.35e-2±1.60e-3**	6.78e-2±3.76e-2	2.59e-1±2.27e-1*
2D Gray-Scott equation				
	50 steps	100 steps	50 steps	100 steps
CNO	8.57e-2±1.26e-2*	2.46e-1±3.34e-2*	7.95e-2±9.51e-4*	1.77e-1±6.40e-3*
FNO	3.23e-2±5.20e-3	1.59e-1±2.20e-2	**4.48e-2±1.23e-2**	1.53e-1±3.08e-3
CAPE-FNO	5.25e-2±5.34e-3	1.71e-1±1.76e-2	7.45e-2±1.42e-2	2.13e-1±1.32e-2
Late Fusion	**1.56e-2±2.37e-3**	**8.50e-2±8.98e-3**	4.82e-2±1.30e-3	**1.04e-1±1.10e-2**

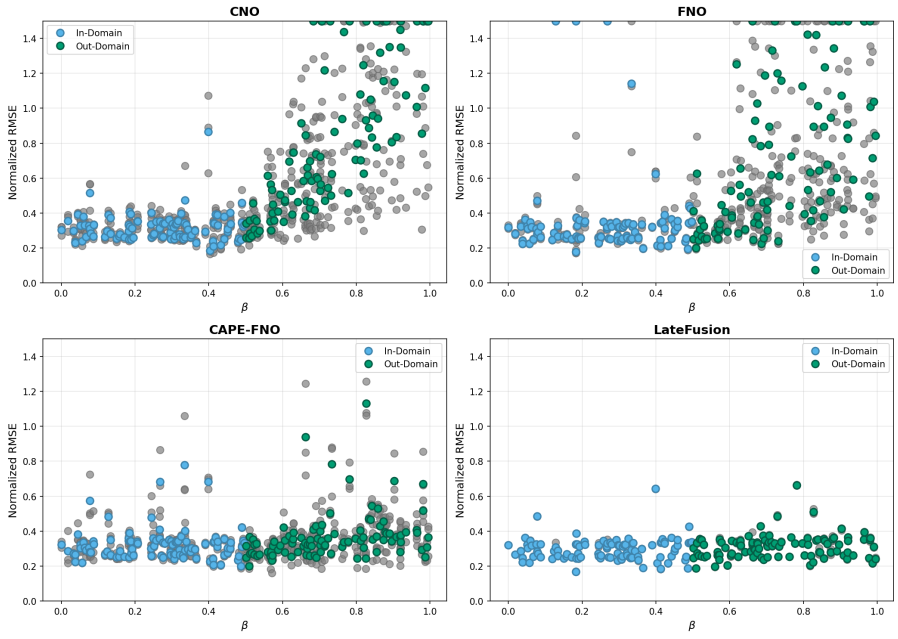

Figure 2: nRMSE per trajectory for the 1D Advection equation as a function of the advection parameter β, shown separately for each model. Each point corresponds to a single trajectory, with grey markers indicating results from different random seeds. The blue and green markers denote the mean nRMSE across seeds for the in-domain and out-domain test sets, respectively. Errors that fall outside the plotting range are clipped and shown as dots at the top of the panel.

the sharp shock structures. For all models, the error increases over time, reflecting the compounding effect of autoregressive prediction in this setting.

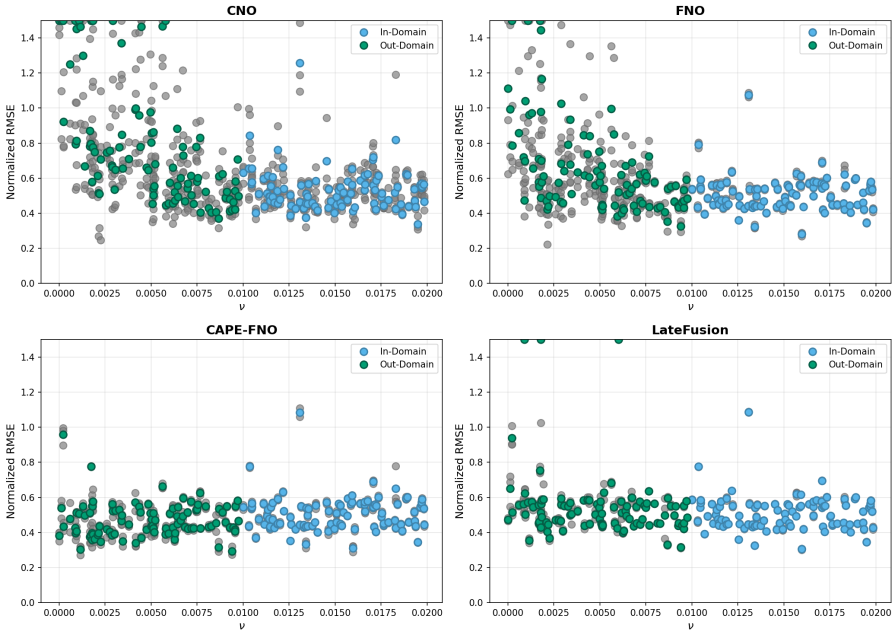

Figure 3: nRMSE per trajectory for 1D Burgers equation as a function of the parameter ν, shown separately for each method. Each point corresponds to a single trajectory, with grey markers indicating results from different random seeds. The blue and green markers denote the mean nRMSE across seeds for the in-domain and out-domain test sets, respectively. Errors that fall outside the plotting range are clipped and shown as dots at the top of the panel.

For the Gray–Scott system, the performance across architectures is more comparable. As shown in Table 2, Late Fusion Operator again achieves the best results on the in-domain test set, while in the out-domain setting it is outperformed by FNO for short-term predictions and by CAPE-FNO for long-term predictions.

However, a visual inspection of the predicted solutions reveals a notable discrepancy between quantitative error and qualitative behavior. Figure 5 shows model predictions at different time instances, where Late Fusion Operator most closely reproduces the characteristic patterns of the true solution, both in-domain and out-of-domain, despite not achieving the lowest nRMSE. This can likely be attributed to the sensitivity of the Gray–Scott system, where small parameter deviations can lead to qualitatively different patterns, resulting in relatively large pointwise errors even when the overall structure is well captured.

5 Discussion

In this work, we focused on the challenging problem of learning parameterized partial differential equations in a strictly one-shot prediction setting, with a particular focus on in- and out-of-distribution parameter extrapolation. Within this setting, we introduced an elegant and flexible framework: Late Fusion Operator for learning parameterized PDEs that decouples the evolution of the state variables from the system parameters. Specifically, our method leverages neural operators to obtain expressive latent representations of the system dynamics that are independent of the parameters. Parameter information is then incorporated through a structured library construction over the learned latent states and parameter inputs, with sparse regression used to identify the resulting dynamics. The design allows the Late Fusion Operator to be readily integrated with existing operator-learning approaches while remaining applicable across a broad range of parameterized PDEs.

A central finding across all experiments is that explicitly separating parameter representations from state dynamics leads to improved generalization in parameterized PDE learning, particularly in the

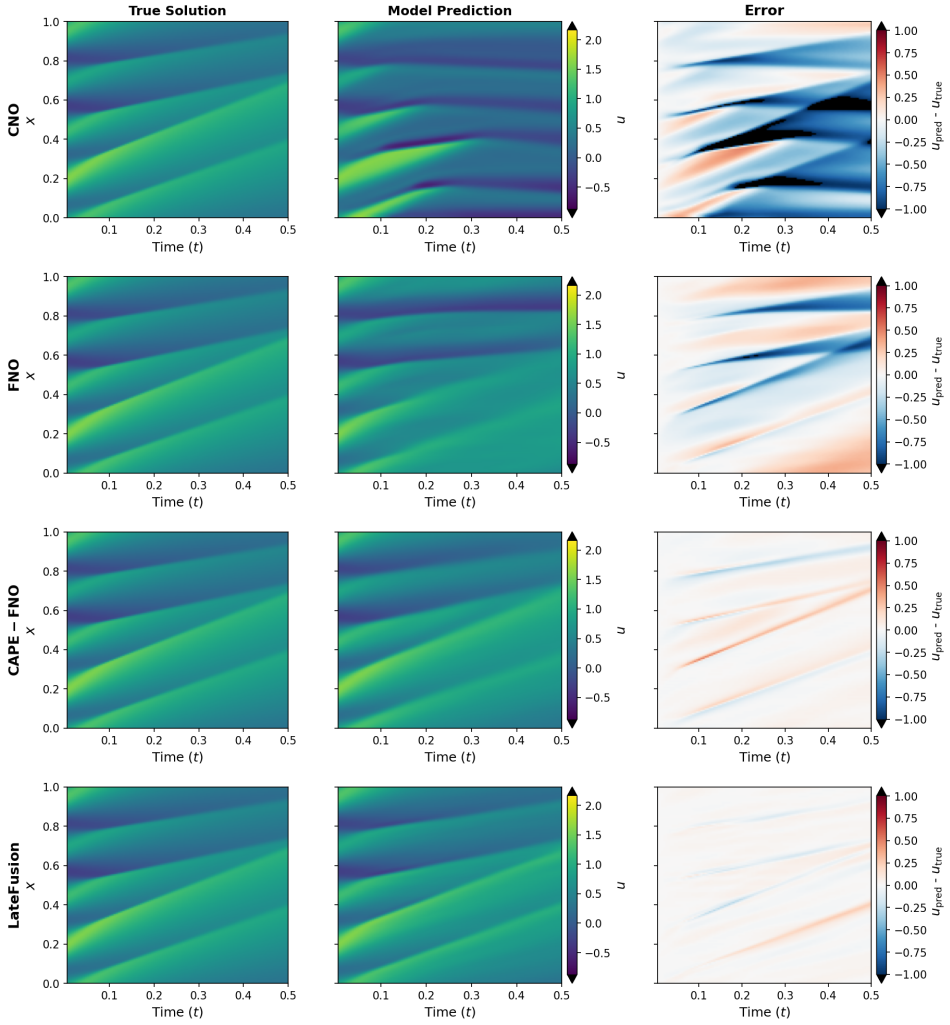

Figure 4: Results 1D Burgers equation for $\nu = 0.0052$ (out-domain). Left column true solution, middle column model predictions and right column error $u_{\text{pred}} - u_{\text{true}}$.

challenging regime of parameter extrapolation. Models that treat parameters as additional input channels, as is done in FNO and CNO, consistently perform poorly when evaluated outside the training distribution. In contrast, approaches that incorporate parameter information through dedicated mechanisms, either via parameter attention, as in CAPE-FNO, or through explicit late fusion, are substantially more robust to domain shifts.

We hypothesize that this improved generalization arises from how parameter–state interactions are modelled. When parameters are provided directly as inputs to a neural operator, the resulting parameter dependence is learned implicitly and can become highly non-linear and entangled with the state representation. This flexibility can lead to unnecessary non-linear correlations, making extrapolation to unseen parameter regimes difficult. By contrast, introducing parameter effects through a structured library and sparse regression restricts the functional form of the parameter dependence and limits unnecessary model complexity. While CAPE-FNO also improves robustness by separating parameter information via attention, it still processes the combined parameter-state representation through a neural operator, which may explain why Late Fusion Operator exhibits greater consistency in certain extrapolation settings.

A key limitation shared by all methods is the growth of error over long prediction horizons, a well-known and actively studied challenge in PDE forecasting. In autoregressive settings, small stepwise

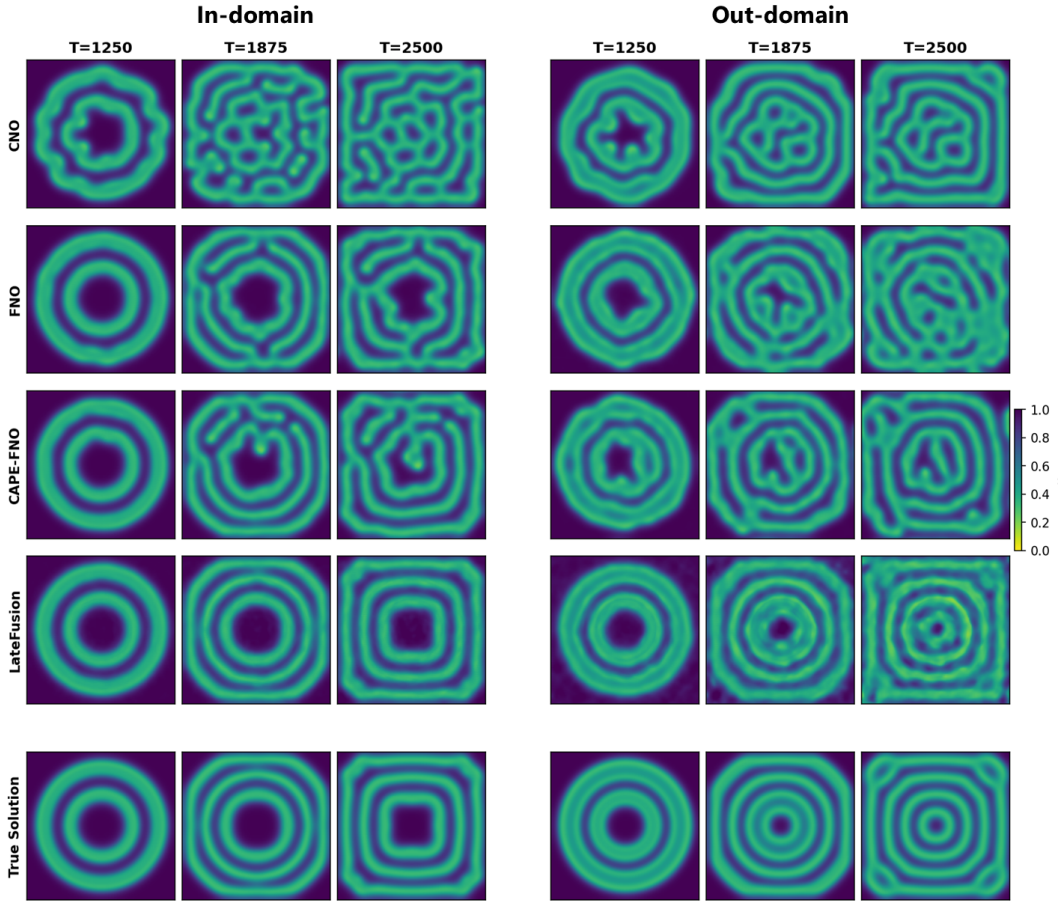

Figure 5: Result Gray-Scott equation for $F = 0.0455$ (in-domain) and $F = 0.050$ (out-domain).

errors accumulate over time, and even methods that perform well in the short term can diverge for extended rollouts. This issue is further amplified when extrapolating to unseen parameter regimes, where small inaccuracies in parameter dependence are compounded across successive steps. Improving long-horizon stability has been a massive topic of interest in recent years (McGreivy & Hakim, 2023; Lippe et al., 2023; Brandstetter et al., 2022; Li et al., 2021). Importantly, many of these strategies, such as noise injection (Wikner et al., 2024) or PDE-refiner (Lippe et al., 2023), can be integrated with the late fusion framework with minimal modification, offering a straightforward path to improved long-term stability.

Looking forward, the proposed Late Fusion Operator opens several promising avenues for further directions. While we focused on demonstrating the benefit of explicit parameter-state separation, the influence of architectural choices, such as the number of latent dimensions, operator design, and the structure of the library, remains largely unexplored and may yield additional gains in extrapolation performance. In particular, developing more principled methods for constructing the library could both enhance accuracy and reduce reliance on manual design. Furthermore, the sparse regression component provides a natural pathway toward interpretability: by identifying a small number of active terms, it may be possible to recover explicit parameter-dependent forms of the learned dynamics, offering insights into the underlying physics.

ACKNOWLEDGMENTS

The authors acknowledge Wageningen University and Research for their investment program Data Science and Artificial Intelligence.

REFERENCES

William F Ames. *Numerical methods for partial differential equations*. Academic press, 2014.

Kamyar Azizzadenesheli, Nikola Kovachki, Zongyi Li, Miguel Liu-Schiaffini, Jean Kossaifi, and Anima Anandkumar. Neural operators for accelerating scientific simulations and design. *Nat. Rev. Phys.*, 6(5):320–328, 2024.

Steffen Bickel, Michael Brückner, and Tobias Scheffer. Discriminative learning under covariate shift. *Journal of Machine Learning Research*, 10(9), 2009.

Fabio Borgogno, P D'odorico, Francesco Laio, and Luca Ridolfi. Mathematical models of vegetation pattern formation in ecohydrology. *Reviews of geophysics*, 47(1), 2009.

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

Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proc. Natl. Acad. Sci. U. S. A.*, 113(15): 3932–3937, 12 April 2016.

L C Evans. *Partial differential equations*, volume 19. American mathematical society, 2022.

Vladimir Sergeevich Fanaskov and Ivan V Oseledets. Spectral neural operators. In *Doklady Mathematics*, volume 108, pp. S226–S232. Springer, 2023.

Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. *arXiv preprint arXiv:2007.01434*, 2020.

Morton E Gurtin, Eliot Fried, and Lallit Anand. *The mechanics and thermodynamics of continua*. Cambridge university press, 2010.

Armand Kassaï Koupaï, Jorge Mifsut Benet, Yuan Yin, Jean-Noël Vittaut, and Patrick Gallinari. Boosting generalization in parametric PDE neural solvers through adaptive conditioning. In *Advances in Neural Information Processing Systems*, volume 37, pp. 70659–70692, 2024.

Nikola B Kovachki, Zong-Yi Li, Burigede Liu, K Azizzadenesheli, K Bhattacharya, A Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to PDEs. *J. Mach. Learn. Res.*, 24(89):89:1–89:97, 2023.

Kangming Li, Andre Niyongabo Rubungo, Xiangyun Lei, Daniel Persaud, Kamal Choudhary, Brian DeCost, Adji Bousso Dieng, and Jason Hattrick-Simpers. Probing out-of-distribution generalization in machine learning for materials. *Commun. Mater.*, 6(1):9, 11 January 2025a.

Siyang Li, Yize Chen, Yan Guo, Ming Huang, and Hui Xiong. Towards generalizable PDE dynamics forecasting via physics-guided invariant learning. *arXiv preprint arXiv:2509.24332*, 2025b.

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

Zongyi Li, Miguel Liu-Schiaffini, Nikola Kovachki, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Learning dissipative dynamics in chaotic systems. *arXiv preprint arXiv:2106.06898*, 2021.

Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *ACM / IMS J. Data Sci.*, 1(3):1–27, 31 July 2024.

Phillip Lippe, Bastiaan S Veeling, Paris Perdikaris, Richard E Turner, and Johannes Brandstetter. PDE-refiner: Achieving accurate long rollouts with neural PDE solvers. *arXiv preprint arXiv:2308.05732*, 2023.

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

Jeff S McGough and Kyle Riley. Pattern formation in the Gray–Scott model. *Nonlinear Anal. Real World Appl.*, 5(1):105–121, February 2004.

Nick McGreivy and Ammar Hakim. Invariant preservation in machine learned PDE solvers via error correction. *arXiv preprint arXiv:2303.16110*, 2023.

Richard H Pletcher, John C Tannehill, and Dale Anderson. *Computational fluid mechanics and heat transfer*. CRC press, 2012.

Bogdan Raoni'c, R Molinaro, T D Ryck, Tobias Rohner, Francesca Bartolucci, Rima Alaifari, Siddhartha Mishra, and Emmanuel de B'ezenac. Convolutional neural operators for robust and accurate learning of PDEs. *Neural Inf Process Syst*, 36:77187–77200, 2 February 2023.

Samuel H Rudy, Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Data-driven discovery of partial differential equations. *Sci. Adv.*, 3(4):e1602614, April 2017.

M Takamoto, T Praditia, Raphael Leiteritz, Dan MacKinlay, F Alesiani, D Pflüger, and Mathias Niepert. PDEBENCH: An extensive benchmark for scientific machine learning. *Neural Inf Process Syst*, abs/2210.07182:1596–1611, 13 October 2022.

M Takamoto, F Alesiani, and Mathias Niepert. Learning neural PDE solvers with parameter-guided channel attention. *ICML*, abs/2304.14118:33448–33467, 27 April 2023.

Lakpa Tamang, Mohamed Reda Bouadjenek, Richard Dazeley, and Sunil Aryal. Handling out-of-distribution data: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 2025.

Geoffrey K Vallis. *Atmospheric and oceanic fluid dynamics*. Cambridge University Press, 2017.

Alexander Wikner, Joseph Harvey, Michelle Girvan, Brian R Hunt, Andrew Pomerance, Thomas Antonsen, and Edward Ott. Stabilizing machine learning prediction of dynamics: Novel noise-inspired regularization tested with reservoir computing. *Neural Netw.*, 170:94–110, February 2024.

A DATA VISUALISATION

In this section, we provide visualisations of the different datasets that were used for our experiments. Figure 6, 7 and 8 display the solution trajectories for different values of the parameters for 1D Advection, 1D Burgers and 2D Gray-Scott respectively.

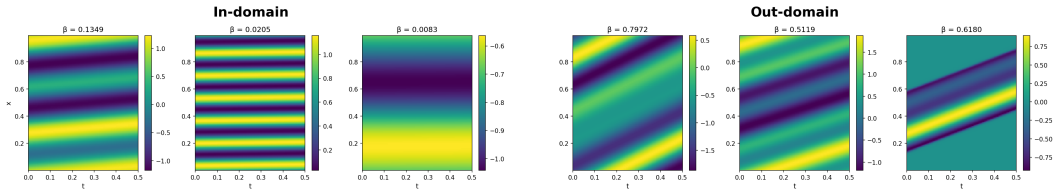

Figure 6: Visualisation of the 1D Advection equation. Trajectories corresponding to different parameter values of β are shown for in-domain and out-domain datasets.

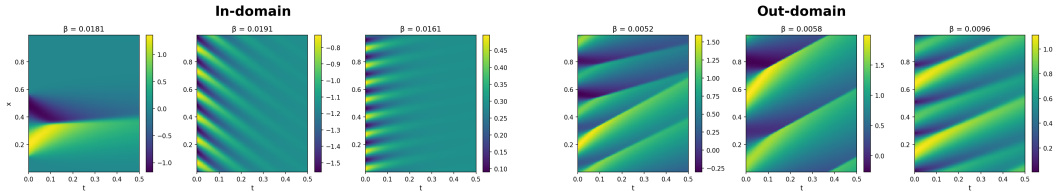

Figure 7: Visualisation of the 1D Burgers equation. Trajectories corresponding to different parameter values of ν are shown for in-domain and out-domain datasets.

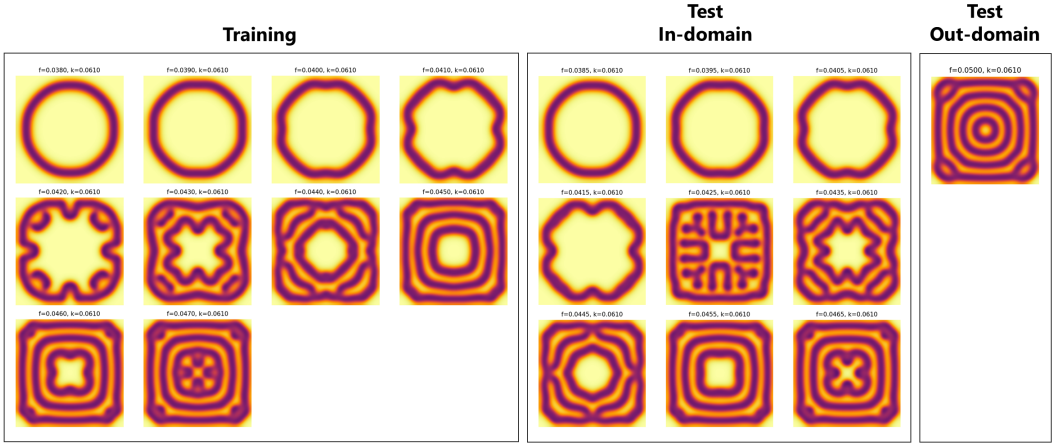

Figure 8: Visualisation of the 2D Gray-Scott equation. Solutions at $T = 2500$ corresponding to different parameter values of F are shown for Training and Test in-domain and Test out-domain datasets.

B HYPERPARAMETERS AND VALIDATION

For fair comparison, all models are trained using the same fixed training schedules and operator architectures, with additional hyperparameter tuning performed only where required.

Training schedules. For the 1D test cases, all models are trained for 500 epochs using an initial learning rate of 10^{-3}, which is reduced by a factor of two every 100 epochs. For the 2D test case, models are trained for 150 epochs with the same initial learning rate, halved every 30 epochs.

Operator architectures. All models that rely on a neural operator backbone, namely FNO, CAPE-FNO, and Late Fusion, employ the same underlying 4-level FNO architecture to ensure a fair and consistent comparison across methods. Specifically, we use:

- 16 Fourier modes and width 64 for the 1D test cases,
- 12 Fourier modes and width 32 for the 2D test cases.

For CNO, which is based on a different operator design, we use the default hyperparameters provided in the original implementation.

Additional hyperparameter tuning. For the CAPE-FNO and Late Fusion models, we perform additional hyperparameter selection for loss-related coefficients, with tuning carried out separately for each equation. In the case of CAPE-FNO, the weight associated with the CAPE loss term λ_{CAPE} is tuned. For the Late Fusion model, the sparsity coefficient $\lambda_{\mathrm{sparse}}$ is selected using validation. In addition, the Late Fusion model employs 4 latent states together with a polynomial feature library of order 3.

Table 3: Hyperparameter search space and selected values for CAPE-FNO and Late Fusion for each equation.

Model	**Hyperparameter**	**Search Space**	**Adv.**	**Burg.**	**Gray-Scott**
CAPE-FNO	λ_{CAPE}	$\{10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$	10^{-5}	10^{-6}	10^{-5}
Late Fusion	$\lambda_{\mathrm{Sparse}}$	$\{10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$	10^{-5}	10^{-6}	10^{-5}

Validation procedure. Hyperparameter selection is performed by splitting the available training data into training and validation sets. For the 1D Advection and Burgers equations, 10% of trajectories are randomly selected as validation data. For the Gray-Scott equation, we use a single trajectory at the boundary of the training parameter range to assess generalization near the edge of the training distribution. For each candidate set of hyperparameters, models are evaluated using the mean squared error (MSE) over a prediction horizon of 20 time steps on the validation set. The best-performing hyperparameters are then used to retrain the corresponding model on the full training dataset.

C Extra results

In this section we will present additional visual results for our 1D test cases. These additional visual results are included for completeness and to provide further qualitative support for the main findings. Figure 9 and 10 show the predicted solution of the different models for the 1D Advection equation for the in-domain and out-domain datasets respectively. Notably, the FNO suffers from a very large rollout error. Looking at the error plots, one can see that the Late Fusion Operator both for the in-domain and out-domain solutions has the lowest overall error.

Figure 11 shows the solution trajectory for a representative value of ν from the in-domain test dataset. A visual comparison between the true solution and the model predictions indicates that all methods capture the overall dynamics reasonably well. However, the corresponding error plots reveal clear performance differences, with the Late Fusion Operator achieving the lowest errors across the trajectory.

Figure 12 and 13 visualizes the learned latent states for a representative input. The hidden states exhibit clear spatial variation and non-trivial structure, indicating that the neural operator learns expressive internal representations rather than collapsing to trivial features. While these latent variables are not directly interpretable, their variability suggests that they encode meaningful information about the underlying system dynamics that is subsequently exploited by the downstream library construction.

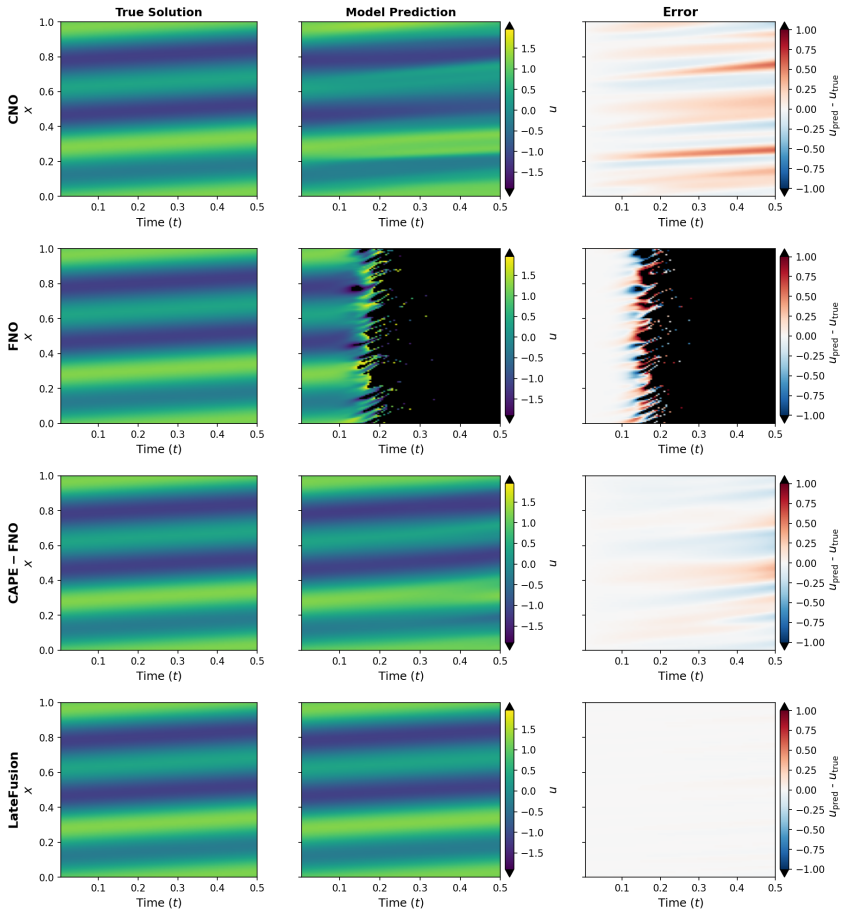

Figure 9: Results 1D Advection equation for $\beta = 0.123$ (in-domain). Left column true solution, middle column model predictions and right column error $u_{\text{pred}} - u_{\text{true}}$.

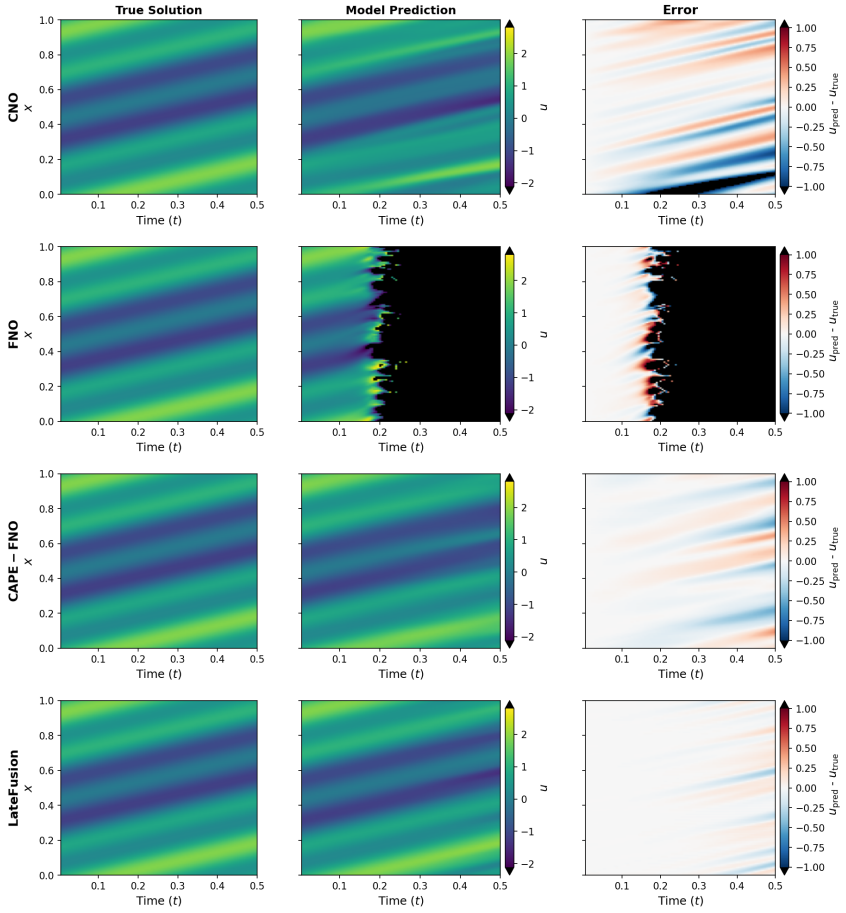

Figure 10: Results 1D Advection equation for $\beta = 0.512$ (out-domain). Left column true solution, middle column model predictions and right column error $u_{\text{pred}} - u_{\text{true}}$.

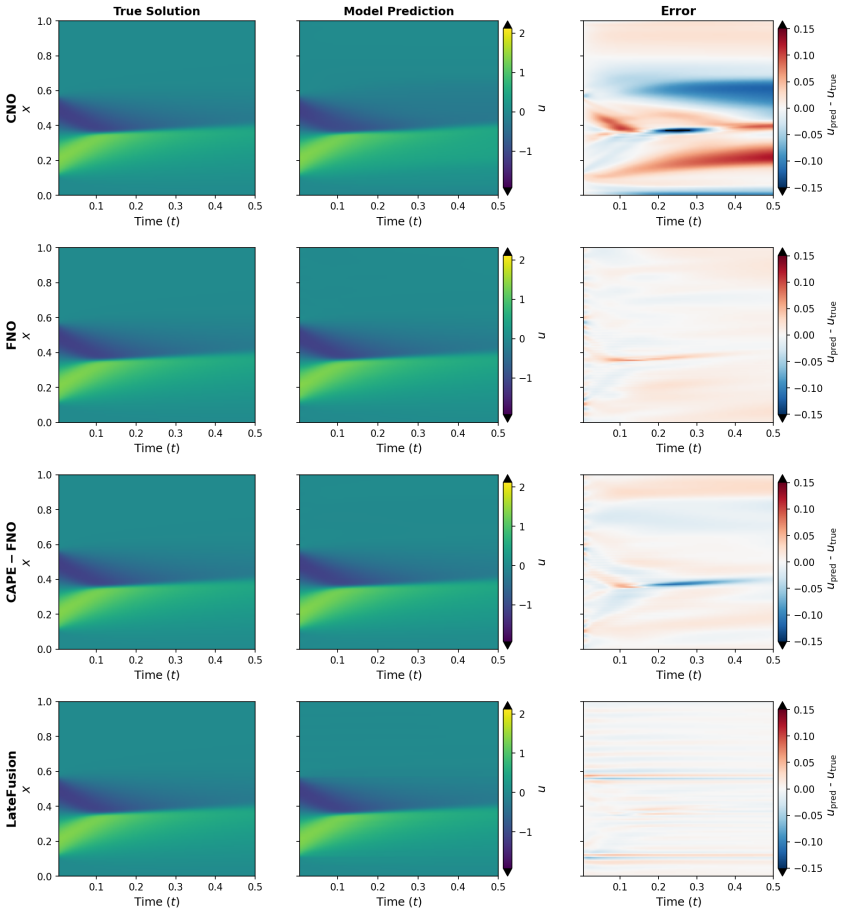

Figure 11: Results Burgers equation for $\nu = 0.0175$ (in-domain). Left column true solution, middle column model predictions and right column error $u_{\text{pred}} - u_{\text{true}}$.

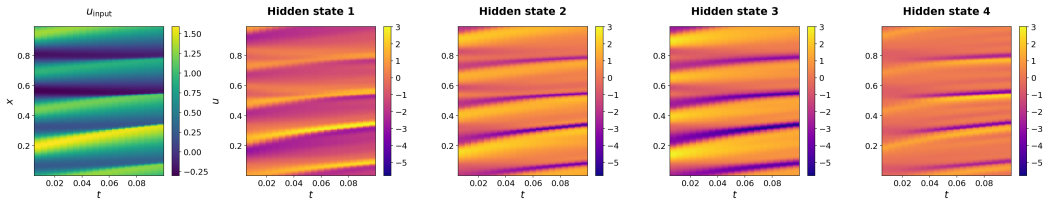

Figure 12: Visualisation of hidden states for Burgers

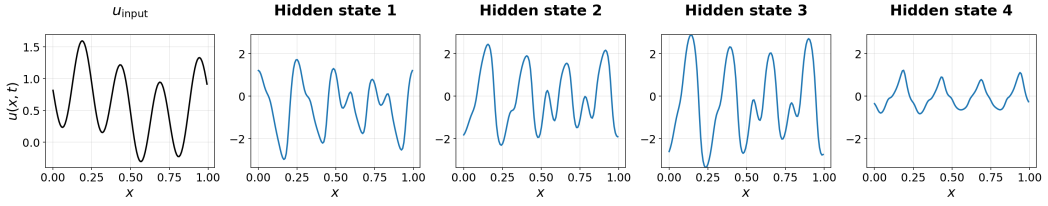

Figure 13: Visualisation of hidden states for Burgers for $t = 0$