A ROBUST IN-CONTEXT MODEL FOR CONSERVATION LAWS: INJECTING CONTEXT INTO FLUX NEURAL OPERATORS VIA RECURRENT VISION TRANSFORMERS

Anonymous authors
Paper under double-blind review

ABSTRACT

PDE foundation models adapt to new systems from context but do so with generic backbones that ignore the numerical structure of the equations they solve. We study the opposite design for one class: families of hyperbolic conservation laws, whose finite-volume structure guarantees discrete conservation. HFluxNO infers a numerical flux operator from a short observed trajectory, with no access to the governing equation or its coefficients: a temporally recurrent vision transformer encodes the trajectory, a hypernetwork maps the encoding to the parameters of a Flux Neural Operator, and the generated flux advances the state through the conservative finite-volume update, so that conservation holds exactly for every generated solver. Across one- and two-dimensional scalar and vector-valued conservation laws, including the compressible Euler equations, HFluxNO improves rollout accuracy over recent in-context solvers, remains the most accurate model under zero-shot shifts to a flux family never seen in training and, in all but one setting, to shock-dominated initial conditions, and conserves the state exactly in every case. Code: `https://anonymous.4open.science/r/in_context_flux_no/`.

1 INTRODUCTION

Neural networks for solving partial differential equations (PDEs) have moved through three stages in quick succession. Physics-informed neural networks fit one network to one PDE instance (Raissi et al., 2019). Neural operators learn the solution map of a fixed PDE across inputs, so that a single trained model can output solutions for different initial or boundary data (Li et al., 2020; Lu et al., 2021; Kovachki et al., 2023). More recently, the success of large pretrained models in language and vision (Brown et al., 2020; Dosovitskiy et al., 2020) has inspired *PDE foundation models*: single models trained across many PDEs, that adapt to a new system from contextual information such as a short observed trajectory or the PDE coefficients (Subramanian et al., 2023; Hao et al., 2024; Herde et al., 2024; Yang & Osher, 2024; McCabe et al., 2024; Morel et al., 2025; Bischof et al., 2025).

While the ambition of this last stage is to treat all of physics as a single domain, the experience of other fields that have embraced foundation models suggests caution. Across genomics, satellite imagery, and time series forecasting, supervised models tuned on the target task matched or exceeded foundation models pretrained on far larger corpora (Xu et al., 2025). In time series forecasting, the pretrained language models inside LLM-based forecasters were found to contribute little (Tan et al., 2024). The trajectory of chaos prediction is the most telling: generic time series foundation models could forecast unseen chaotic systems zero-shot (Zhang & Gilpin, 2025), but by parroting the context rather than inferring the dynamics. The field has since moved to models pretrained on, and designed for, dynamical systems, with markedly better generalization (Hemmer & Durstewitz, 2025; Lai et al., 2026).

Physics is no exception. A recent benchmark of pretrained PDE foundation models found their generalization to be uneven: accuracy varies widely across equation types, degrades most on wave-like and high-variation dynamics, and is not consistently improved by pretraining or model scale (Chu et al., 2026). This is not incidental: sharp fronts, finite propagation speeds, and weak solutions impose demands on a learned solver that are incompatible with those of diffusive or oscillatory

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

dynamics, and a single dense architecture serving all of them must compromise on each. Classical numerical analysis reflects the same tension. No universal solver exists for arbitrary PDEs; a method commits to a class of equations, and its stability and accuracy follow from the mathematical properties of that class (LeVeque, 2002; Larsson & Thomée, 2009).

We read this evidence as a central design principle for PDE foundation models: *broad adaptability should first be developed within a mathematically coherent class, by incorporating the numerical structure of the class into the model, before being extended to unrestricted multiphysics settings.* Hyperbolic conservation laws are the natural class with which to test this principle. Their solutions develop discontinuities from smooth data, and operator-learning architectures with a linear reconstruction step provably cannot approximate the resulting solution maps efficiently (Lanthaler et al., 2023). Architectures that escape this bound still require dedicated modifications to handle shocks and contact discontinuities reliably (Peyvan et al., 2024). At the same time, the class offers a mature numerical structure to inherit: the finite-volume method, which evolves cell averages of the solution through differences of numerical fluxes and underlies modern shock-capturing schemes (LeVeque, 2002).

In this work we therefore study generalization across a parametrized *family* of hyperbolic conservation laws: from a short observed trajectory alone, the model must infer a numerical flux consistent with the unknown governing equation and produce accurate rollouts. Existing in-context neural operators address such tasks with generic sequence or attention backbones that predict the next state directly, and so do not structurally enforce conservation (Yang & Osher, 2024; Hao et al., 2024; Morel et al., 2025). Conversely, the recent line of finite-volume-inspired neural solvers learns a numerical flux for a single fixed equation (Chen et al., 2024; Kim et al., 2025; Lichtlé et al., 2026). To our knowledge, no existing method infers a conservative flux operator from observed trajectories alone, without knowledge of the governing equations.

Our main contributions are as follows:

- We formulate in-context flux learning for parametrized conservation laws: a short observed trajectory is used to infer a numerical flux operator without access to the equation, its coefficients, or its flux function.
- We introduce HFluxNO, in which a recurrent ViT encoder and a hypernetwork instantiate a conservative finite-volume target network whose update conserves the state exactly by construction, and show that generic substitutes for any of its three components degrade accuracy.
- On one- and two-dimensional scalar and vector-valued conservation laws, HFluxNO improves rollout accuracy over recent in-context PDE solvers and generalizes better under controlled distribution shift to unseen flux functions and, in all but one setting, to shock-dominated initial conditions.

2 BACKGROUND

Hyperbolic conservation laws. We consider systems of conservation laws

$$\partial_t \boldsymbol{u} + \nabla \cdot \boldsymbol{F}(\boldsymbol{u}; \boldsymbol{p}) = \boldsymbol{0}, \quad \boldsymbol{u} \in \mathbb{R}^d, \ \boldsymbol{F}(\cdot\,; \boldsymbol{p}) : \mathbb{R}^d \to \mathbb{R}^{d \times N} \tag{1}$$

where $\boldsymbol{u}$ is the vector of d conserved variables and $\boldsymbol{F}$ is the physical flux in N spatial dimensions, parameterized by coefficients $\boldsymbol{p}$. We refer to a *family* of conservation laws as the set of equations obtained as $\boldsymbol{p}$ varies across a prescribed distribution, or more generally, as $\boldsymbol{F}$ varies across a prescribed class of flux functions. Eq. (1) is hyperbolic when, for every direction $\boldsymbol{n}$, the Jacobian $\partial(\boldsymbol{F} \cdot \boldsymbol{n})/\partial \boldsymbol{u}$ has real eigenvalues, corresponding to the characteristic speeds at which information propagates in the system. Two properties of hyperbolic PDEs shape every numerical method for them: (i) solutions with smooth initial data can develop discontinuities, after which Eq. (1) holds only in the weak sense, and (ii) weak solutions are not unique, so the physically admissible one must be selected by an entropy condition (LeVeque, 2002; Toro, 2009).

Finite volume methods. Finite volume methods discretize the integral form of Eq. (1), which states that the amount of $\boldsymbol{u}$ in any region changes only through the flux across its boundary. Space is partitioned into cells, and the cell average of $\boldsymbol{u}$ is advanced in time by the net flux through the cell

faces (LeVeque, 2002). Consider a Cartesian grid in N spatial dimensions with cell widths Δx_ℓ and time levels $t^n = n\Delta t$, and let $\boldsymbol{u}_{\boldsymbol{i}}^n$ denote the cell average in cell $\boldsymbol{i} \in \mathbb{Z}^N$. The finite volume update is then

$$\boldsymbol{u}_{\boldsymbol{i}}^{n+1} = \boldsymbol{u}_{\boldsymbol{i}}^n - \Delta t \sum_{\ell=1}^{N} \frac{1}{\Delta x_\ell} \left(\hat{\boldsymbol{f}}^{(\ell),n}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_\ell} - \hat{\boldsymbol{f}}^{(\ell),n}_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}_\ell} \right), \tag{2}$$

where $\boldsymbol{e}_\ell$ is the unit vector in direction ℓ. Since every interior flux enters the update of two neighboring cells with opposite signs, summing Eq. (2) over $\boldsymbol{i}$ telescopes. The total $\sum_{\boldsymbol{i}} \boldsymbol{u}_{\boldsymbol{i}}^n$ therefore changes only through the fluxes at the domain boundary, and under periodic boundary conditions it is conserved exactly for any choice of $\hat{\boldsymbol{f}}$.

The *numerical flux* $\hat{\boldsymbol{f}}^{(\ell)}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_\ell}$ approximates $\boldsymbol{F}^{(\ell)}$, the ℓ-th column of $\boldsymbol{F}$, on the face between cells $\boldsymbol{i}$ and $\boldsymbol{i} + \boldsymbol{e}_\ell$. It is computed from the cell averages in a stencil $\mathcal{S}_\ell(\boldsymbol{i})$ around the face,

$$\hat{\boldsymbol{f}}^{(\ell),n}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_\ell} = \hat{\boldsymbol{f}}^{(\ell)}\left(\{\boldsymbol{u}_{\boldsymbol{j}}^n\}_{\boldsymbol{j}\in\mathcal{S}_\ell(\boldsymbol{i})} \right), \tag{3}$$

and is called consistent if it returns $\boldsymbol{F}^{(\ell)}(\boldsymbol{u})$ whenever all stencil values equal $\boldsymbol{u}$. The smallest stencil consists of the two cells adjacent to the face; the first-order Rusanov flux (Rusanov, 1962; Toro, 2009), for instance, averages the physical flux of the two states and adds a dissipation proportional to the largest local wave speed (Section B). Higher-order schemes use larger stencils in the normal (Shu, 2009) and transverse (LeVeque, 1997; Balsara, 2010) directions. In every case, the numerical flux is derived from the physical flux $\boldsymbol{F}$ and hence requires knowledge of the governing equation.

3 Related work

PDE foundation models and in-context operator learning. Foundation models for PDEs aim to solve many equations with a single network, and what separates them is how much the model must be told about the system it is solving. The largest are pretrained on a corpus of simulations and reach a new system through fine-tuning (Subramanian et al., 2023; McCabe et al., 2024; Herde et al., 2024; Hao et al., 2024; Rahman et al., 2024; McCabe et al., 2026). Others are told the system outright, through its coefficients, a symbolic equation, or a full specification passed as auxiliary input (Takamoto et al., 2023; Hagnberger et al., 2024; Bischof et al., 2025; Zhou et al., 2025; Ye et al., 2025). The setting of this paper is the third: nothing about the equation is supplied, and the model must identify it at inference time from contextual data, such as example solutions of the same system (Yang et al., 2023; Yang & Osher, 2024; Serrano et al., 2025; Cao et al., 2026) or a short history of the trajectory being predicted (Morel et al., 2025; Koupaï et al., 2025). Whatever the adaptation route, the backbone is a generic architecture that outputs the solution field directly, and none constrains the update to the conservative form (Eq. (2)).

Numerical structure in neural solvers. A separate tradition keeps a classical discretization of a known equation and hands part of it to a network. At one end, a classical solver still produces the solution and the network supplies one local ingredient, such as a stencil coefficient or a correction after each step. The ingredient is learned from high-resolution reference simulations so that a coarse grid recovers fine-grid accuracy (Bar-Sinai et al., 2019; Kochkov et al., 2021; Um et al., 2020). In shock-capturing schemes, the ingredient pertains to discontinuity treatment: WENO reconstruction weights (Stevens & Colonius, 2020; Bezgin et al., 2022; Charles & Ray, 2025), troubled-cell detection and artificial viscosity in discontinuous Galerkin schemes (Ray & Hesthaven, 2018; Discacciati et al., 2020), or the Riemann solver at a tracked interface (Magiera et al., 2020). At the other end, the scheme disappears into the architecture: message-passing that generalizes stencil updates (Brandstetter et al., 2022), outputs that are divergence-free by construction (Liu et al., 2024), or predictions projected onto the integral form of a conservation law (Hansen et al., 2023). Across this range the learned component travels with the scheme, and with it the requirement that the equation be known. Where a single model serves several equations, it does so because the host scheme is given the physical flux (Stevens & Colonius, 2020; Charles & Ray, 2025) or the coefficients are given as inputs (Brandstetter et al., 2022), never because the system is inferred from context.

Flux learning in finite-volume form. If conservation is the structure to keep, the natural design is to keep the finite-volume update (Eq. (2)) and let a network stand in for its flux, so that discrete

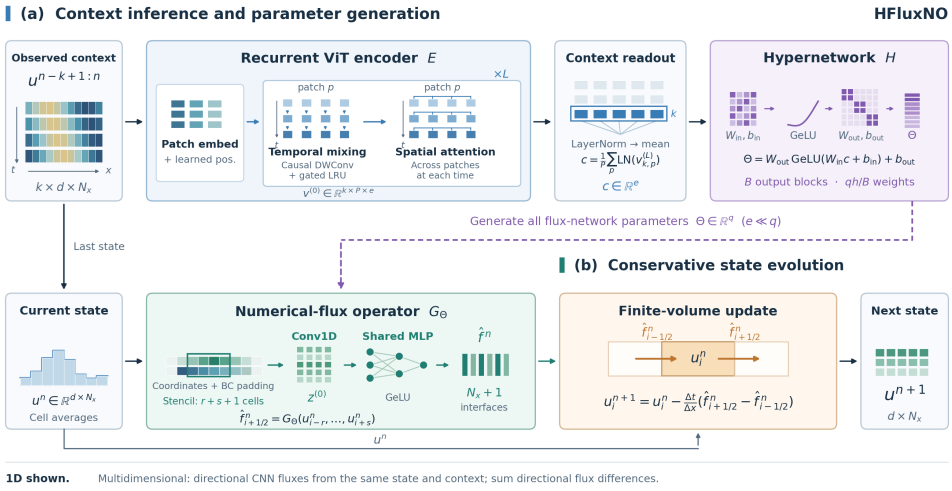


Figure 1: Overview of HFluxNO. (a) A temporally recurrent vision transformer encodes the context window into a context vector, which a hypernetwork maps to the parameters of the target network. (b) The generated target network predicts numerical fluxes at every cell face and advances the state through the conservative finite-volume update.

conservation holds whatever the network computes. Two things can be meant by the flux. Learning the *physical* flux of the governing equation, or a constitutive closure within it, leaves a classical numerical flux in place and inherits its admissibility guarantees, such as Rankine-Hugoniot or entropy conditions (Karlbauer et al., 2022; Patsatzis et al., 2025; Liu et al., 2026). Learning the *numerical* flux (Eq. (3)) itself, as a network over the local stencil (Chen et al., 2024; Lichtlé et al., 2026; Kim & Kang, 2025; Kim et al., 2025) or as the message function of a graph solver (Horie & Mitsume, 2024), keeps conservation as the only structure and removes the need for a flux formula. Either way, one flux is trained per equation and its parameters are fixed once training ends. In this work, we take the second route but generate the flux parameters from an observed trajectory - allowing a single conservative solver to span a family of equations (Eq. (1)).

4 HFluxNO: An in-context flux neural operator

Given a family of conservation laws (Eq. (1)) and a context window $\boldsymbol{U}^n := (\boldsymbol{u}^{n-k+1}, \dots, \boldsymbol{u}^n)$ of k consecutive states of one member of the family, sampled on a periodic Cartesian grid with $N_1 \times \cdots \times N_N$ cells, our goal is to predict the next state $\boldsymbol{u}^{n+1}$. Neither the flux $\boldsymbol{F}$, its coefficients $\boldsymbol{p}$, nor any identifying label for the equation is supplied; the system must be inferred from $\boldsymbol{U}^n$ alone.

We propose HFluxNO, which learns the map $\boldsymbol{U}^n \mapsto \boldsymbol{u}^{n+1}$ not as an unconstrained predictor but as the composition of three components (Fig. 1): (i) a ***context encoder*** E that compresses $\boldsymbol{U}^n$ into a context vector $\boldsymbol{c} = E(\boldsymbol{U}^n) \in \mathbb{R}^e$, implicitly solving an inverse problem for the governing dynamics; (ii) a ***hypernetwork*** H that maps $\boldsymbol{c}$ to the parameters $\boldsymbol{\Theta} = H(\boldsymbol{c}) \in \mathbb{R}^q$ of the target network; and (iii) a ***target network*** $G_{\boldsymbol{\Theta}}$, a neural numerical flux that advances $\boldsymbol{u}^n$ to $\boldsymbol{u}^{n+1}$ through the finite-volume update (Eq. (2)). This composition imposes an information bottleneck $\mathbb{R}^{k\times d\times N_1\times\cdots\times N_N} \to \mathbb{R}^e \to \mathbb{R}^q$ with $e \ll q$: the window is first distilled into a low-dimensional summary of the dynamics, which then determines the solver. During autoregressive rollout, the model's own prediction is appended to the window and the three components are re-evaluated at every step.

4.1 Context encoder

From a short trajectory, the encoder must extract identifying information for the underlying equation: wave speeds, dissipation, and the nonlinearity of the flux. This information pertains to how the solution *evolves*, so the encoder should respect the temporal ordering of the window rather than treating time as another spatial axis. We therefore adopt a temporally recurrent vision transformer

in the spirit of TRecViT (Patraucean et al., 2025), in which temporal mixing is handled by gated linear recurrent units (De et al., 2024; Botev et al., 2024) and spatial mixing by transformer blocks (Dosovitskiy et al., 2020).

Each state in the window is tokenized into P spatial patches by a linear patch embedding with learned positional encodings, giving $\boldsymbol{v}^{(0)} \in \mathbb{R}^{k\times P\times e}$, indexed by window time $\tau = 1,\dots,k$ and patch $j = 1,\dots,P$. Each of L encoder layers then alternates a recurrent block applied along time to every patch and a self-attention block applied across patches at every time,

$$\widehat{\boldsymbol{v}}^{(m)}_{:,j} = \mathrm{Rec}^{(m)}\big(\boldsymbol{v}^{(m)}_{:,j}\big), \qquad \boldsymbol{v}^{(m+1)}_{\tau,:} = \mathrm{Attn}^{(m)}\big(\widehat{\boldsymbol{v}}^{(m)}_{\tau,:}\big), \qquad m = 0,\dots,L-1, \tag{4}$$

where Rec is a residual block around a causal gated recurrent unit and Attn a standard transformer encoder block. The recurrent block is causal, so the representation at window time τ depends only on states up to τ, and the attention block mixes patches within a single time, so causality is preserved through the whole encoder. The final time step therefore carries the accumulated information of the window, and the context vector is read out there by normalizing and averaging over patches,

$$\boldsymbol{c} = \frac{1}{P}\sum_{j=1}^{P} \mathrm{LayerNorm}\big(\boldsymbol{v}^{(L)}_{k,j}\big) \in \mathbb{R}^e. \tag{5}$$

Implementation details of the recurrent block, and the spatio-temporal ViViT encoder (Arnab et al., 2021) against which we ablate this design in Section 4.4, are given in Section A.1.

4.2 Hypernetwork

The hypernetwork maps the context vector to the full parameter vector of the target network through a multilayer perceptron (MLP) trunk $\mathbb{R}^e \to \mathbb{R}^e$ and a linear head,

$$\boldsymbol{\Theta} = H(\boldsymbol{c}) = W_{\mathrm{out}}\,\mathrm{MLP}(\boldsymbol{c}) + \boldsymbol{b}_{\mathrm{out}}, \qquad W_{\mathrm{out}} \in \mathbb{R}^{q\times e},\ \boldsymbol{b}_{\mathrm{out}} \in \mathbb{R}^q. \tag{6}$$

Two choices keep this design feasible at $q \sim 10^6$. First, W_{out} is block-diagonal with B blocks, which reduces its size from qe to qe/B.[1] Second, a standard initialization of W_{out} does not place the generated target network at a good initialization of its own (Chang et al., 2019; Ortiz et al., 2023; Beck et al., 2023). We therefore use Bias-HyperInit (Beck et al., 2023), which sets $W_{\mathrm{out}} = 0$ and $\boldsymbol{b}_{\mathrm{out}}$ to the default initialization of the target network. Then $H(\boldsymbol{c}) = \boldsymbol{b}_{\mathrm{out}}$ for every $\boldsymbol{c}$ at the start of training, and the dependence on the context is learned as a perturbation of a well-initialized solver (Section A.2).

4.3 Target network: Flux Neural Operator

The target network is a finite-volume solver whose numerical flux is a neural network with parameters $\boldsymbol{\Theta}$. Following the stencil form of the numerical flux (Eq. (3)), we define

$$\hat{\boldsymbol{f}}^{(\ell),n}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_\ell} = G^{(\ell)}_{\boldsymbol{\Theta}}\left(\{\boldsymbol{u}^n_{\boldsymbol{i}'}\}_{\boldsymbol{i}'\in\mathcal{S}_\ell(\boldsymbol{i})}\right), \qquad \ell = 1,\dots,N, \tag{7}$$

where the stencil $\mathcal{S}_\ell(\boldsymbol{i})$ spans $r+1$ cells behind and s cells ahead of the face along the normal direction ℓ, and $s_\perp$ cells to either side along every transverse direction. The state is then advanced in time by Eq. (2). We call this target a Flux Neural Operator after Kim & Kang (2025), but we do not parametrize the flux by a Fourier neural operator, which has a global domain of dependence (Li et al., 2020). Instead, $G_{\boldsymbol{\Theta}}$ is a convolution over the stencil followed by a channel-wise MLP, so that the learned flux has the finite domain of dependence of a classical numerical flux and q stays small enough for a hypernetwork to generate. The state is padded periodically, in the manner of ghost cells, by $(r+1, s)$ cells along ℓ and $s_\perp$ cells along the other axes. A convolution with kernel $(r+s+1)\times(2s_\perp+1)^{N-1}$ produces one feature vector per face, which a shared MLP then maps to the flux. A single kernel and a single MLP serve all N directions. The kernel is oriented along ℓ by transposing that axis to the front, and the MLP has N output heads, of which the ℓ-th gives $G^{(\ell)}_{\boldsymbol{\Theta}}$. All directional fluxes are evaluated at the same state $\boldsymbol{u}^n$, and Δt and Δx_ℓ enter the update explicitly rather than being absorbed into the network. Further details, including the optional coordinate channels, are in Section A.3.

[1] The same device is used in adaptive Fourier neural operators (Guibas et al., 2021) and in the gates of the recurrent unit (De et al., 2024).

Table 1: Necessity of each component, on the 1D cubic family: 20-step autoregressive rollout error, mean $\pm$ std over three training runs, and ratio to the full model. Upper block (Q1): is context-based adaptation needed? Lower block (Q2): given adaptation, does the choice of each component matter?

Varied	Alternative	Rel. ℓ^2 ($\times 10^{-1}$)	Rel. ℓ^∞ ($\times 10^{-1}$)	Ratio
—	**HFluxNO (full)**	$\mathbf{0.521 \pm 0.022}$	$\mathbf{3.68 \pm 0.16}$	**1.0**$\times$
All (Q1)	Fixed Flux NO, $\boldsymbol{u}^n \to \boldsymbol{u}^{n+1}$	3.58 ± 0.001	9.92 ± 0.00	6.9$\times$
All (Q1)	FNO on window, $\boldsymbol{U}^n \to \boldsymbol{u}^{n+1}$	2.75 ± 0.003	9.79 ± 0.11	5.3$\times$
Target (Q2)	FNO	1.53 ± 0.050	7.21 ± 0.12	2.9$\times$
Encoder (Q2)	ViViT (spatio-temporal patches)	1.36 ± 0.031	4.28 ± 0.16	2.6$\times$
Target (Q2)	U-Net	0.830 ± 0.070	5.13 ± 0.22	1.6$\times$
Cond. (Q2)	Concatenation	0.544 ± 0.018	3.82 ± 0.12	1.04$\times$

Conservation. With periodic padding, the faces at either end of each axis are computed from identical stencils, so summing Eq. (2) over $\boldsymbol{i}$ telescopes and $\sum_{\boldsymbol{i}} \boldsymbol{u}_{\boldsymbol{i}}^n$ is conserved exactly. This holds for *every* parameter vector $\boldsymbol{\Theta}$ the hypernetwork can produce and at every step of an autoregressive rollout; it is a property of the architecture, not of training. Section E confirms this to floating-point precision.

Scale normalization. For the Euler system, whose conserved variables differ in magnitude, the target's input and output are rescaled by the per-variable root-mean-square $\boldsymbol{\sigma}$ of the context window and $\log \boldsymbol{\sigma}$ is passed to the hypernetwork; as $\boldsymbol{\sigma}$ is constant over the grid, conservation is unaffected (Section A.2).

4.4 Necessity of each component

Before comparing HFluxNO with other methods, we answer two questions about its design. **(Q1) Is context-based adaptation needed at all?** We compare against two models that receive the same information but do not adapt: a fixed Flux NO mapping $\boldsymbol{u}^n$ to $\boldsymbol{u}^{n+1}$ with parameters shared across the family, and a Fourier neural operator (FNO) mapping the entire window $\boldsymbol{U}^n$ to $\boldsymbol{u}^{n+1}$ directly. **(Q2) Can the components be replaced?** We substitute a spatio-temporal transformer (Arnab et al., 2021) for the recurrent encoder, concatenation of $\boldsymbol{c}$ to the target's input for parameter generation, and an FNO (Li et al., 2020) or a U-Net (Ronneberger et al., 2015) for the flux network. We test on the one-dimensional cubic family with comparable parameter count for every variant and report the 20-step rollout error for each.

The results in Table 1 clearly answer both questions. **(A1) Adaptation is necessary and just providing context is not enough.** Both the fixed Flux NO and the FNO that sees the whole window show more than five-fold error increase, signifying that adapting across a PDE family requires a dedicated mechanism for extracting context from solution history. **(A2) No component is interchangeable.** Replacing the conservative target by an FNO costs a factor of three and by a U-Net a factor of 1.6, consistent with the remaining gap being due to the absence of finite-volume structure. Replacing the causal encoder by a spatio-temporal transformer costs a factor of 2.6, consistent with the reading of Section 4.1 that identifying the equation is an inverse problem over the time evolution. Generating the parameters rather than concatenating the context is the smallest effect, 4%; we keep the hypernetwork because it delivers a solver with explicit parameters that can be run as a standalone finite-volume scheme. The two largest effects, causal adaptation and the conservative target, are thus both necessary for strong performance.

5 Experimental setup

Benchmark families. We evaluate on the five families of Table 2, generated for this work so that each generalization claim can be tied to one controlled axis of variation. The axes are: coefficients within a flux family, the flux family itself, the initial-condition type, the spatial dimension, scalar versus vector-valued states, and the presence of diffusion (viscous Burgers, the one member outside the strictly hyperbolic class). A model sees trajectories from many coefficient draws during train-

ing and is tested on unseen draws; the out-of-distribution sets in the last column shift one further axis, the initial-condition type or the flux family, and are evaluated zero-shot. Equations, sampling distributions, reference solvers, and split sizes are given in Section B.

Table 2: Benchmark families. All problems are posed on the periodic domain $[0, 1]^N$ and saved with $\Delta t = 0.005$ up to $T = 0.4$; reference solutions are computed with classical finite-volume solvers (Section B). GRF: Gaussian random field; IC: initial condition.

Family	Equation and sampled parameters	Training IC	Grid	OOD test sets
1D cubic	$u_t + (c_1 u + c_2 u^2 + c_3 u^3)_x = 0$ $\boldsymbol{c} \sim \mathcal{U}[-1, 1]^3$	periodic GRF	100	shock IC; sine flux, GRF or shock IC
1D shallow water	(h, m) system, Section B $\alpha, \gamma \sim \mathcal{U}[0.5, 1.5]$, $\beta \sim \mathcal{U}[8, 12]$	GRF m, lognormal h	100	shock IC in h
1D viscous Burgers	$u_t + (au^2)_x = \nu u_{xx}$ $a \sim \mathcal{U}[0.5, 1.5]$, $\nu \sim \mathcal{U}[0.005, 0.015]$	periodic GRF	100	shock IC
2D cubic	$u_t + f(u)_x + f(u)_y = 0$, cubic f $\boldsymbol{c} \sim \mathcal{U}[-1, 1]^3$	2D spectral GRF	50^2	shock IC
2D Euler	ideal gas, Section B $\gamma \sim \mathcal{U}[1.2, 1.5]$	transonic GRF primitives	128^2	shock IC

Baselines. We compare against two recent in-context PDE solvers that cover the generic routes to context conditioning: **DPOT** (Hao et al., 2024), which aggregates the context by patch embedding and a learned temporal weighting and conditions a Fourier-attention operator (Guibas et al., 2021) on the result, and **DISCO** (Morel et al., 2025), which, like HFluxNO, uses a hypernetwork, here mapping the context to the parameters of a U-Net vector field (Ronneberger et al., 2015) that is integrated in time as a neural ODE (Chen et al., 2018). All models are implemented in JAX (Bradbury et al., 2018) from the original code where available and configured to comparable parameter counts (Section C). **ICON** (Yang et al., 2023) predicts a target snapshot from example pairs rather than advancing the state autoregressively, so it cannot be placed on the same rollout footing; we report it separately in Section D.

Training and evaluation. All models are trained on the one-step prediction error with a context length of $k = 20$, for 50,000 AdamW steps with linear warm-up and cosine decay, and every number we report is the mean and standard deviation over three training runs (Section C). Each family is evaluated on a held-out test set with unseen coefficient draws and initial conditions in four settings: (i) single-step prediction from a true context; (ii) a 20-step autoregressive rollout, generated with the model's single step outputs fed back as context; (iii) autoregressive rollout over the full trajectory of 80 steps to $T = 0.4$; and (iv) zero-shot evaluation on the out-of-distribution sets of Table 2. Accuracy is the relative ℓ^2 and ℓ^∞ error, $\|\hat{\boldsymbol{u}} - \boldsymbol{u}\|_p / \|\boldsymbol{u}\|_p$ with $p \in \{2, \infty\}$, taken over all cells and time steps of the evaluated horizon; the ℓ^∞ error is included for its sensitivity to shock-front locations.

6 Results

6.1 Accuracy across families and distribution shifts

Figure 2 summarizes the in-distribution and out-of-distribution accuracy after a 20-step rollout, i.e., one context length. In distribution, HFluxNO has the lowest error on every family in both norms. Its margin over the stronger baseline in relative ℓ^2 ranges from 1.4 (shallow water) to 3.2 (2D Euler). The margin is largest on the two-dimensional problems. There DPOT's error is 17 times (2D cubic) and 5.7 times (Euler) that of HFluxNO, and DISCO's hypernetwork narrows the gap to a factor of about three without closing it. The ℓ^∞ errors of all models are several times their ℓ^2 errors. The rollouts in Section G show why. The error concentrates at moving fronts, and a small error in a front's position incurs a large pointwise error.

Under distribution shift HFluxNO remains the most accurate model on every family but shallow water. Its error grows more than the baselines' in relative terms. On the 1D cubic and viscous

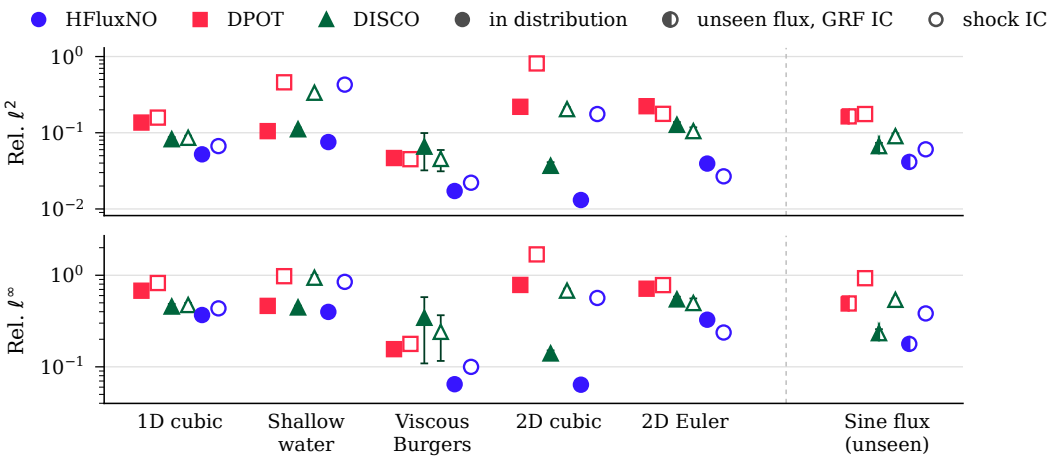


Figure 2: Accuracy after a 20-step rollout, one context length, on every family of Table 2. Marker fill encodes the test set: filled, in distribution; hollow, shock-dominated initial conditions; half-filled, the unseen sine flux with training-like initial conditions (right-hand group, cubic-trained models). Whiskers are one standard deviation over three training runs, drawn behind the markers; for most entries the standard deviation is below 5% of the mean and the whisker is hidden by the marker. Single-step errors and all numbers are tabulated in Section F.

Burgers families it rises by about 30%, whereas the baselines change by -30 to $+16\%$. On 2D cubic it rises by an order of magnitude, whereas DISCO's grows five-fold. The ordering of the models is nonetheless unchanged on these families, because HFluxNO starts from a lower level. The Euler system is the exception in the other direction. Every model is more accurate on the Voronoi-cell initial conditions than in distribution, HFluxNO by a third, which we conjecture is because the piecewise-constant states contain fewer interacting waves than the random smooth fields.

The unseen-flux group is the more demanding test. The model has only seen cubic fluxes in training and must infer a sine flux from twenty snapshots. Its rollout error there is no larger than its error on the cubic family under the same initial conditions, and the margin over the baselines is preserved. The one setting in which HFluxNO is not the most accurate is the shallow-water system under shock-dominated height profiles. There DISCO is better after 20 steps in ℓ^2, and both DISCO and DPOT are better at a single step in both norms. We have not diagnosed this case. We note in Section 7 that the shock-dominated height profiles of this set lie outside the range of states seen in training.

6.2 Error growth beyond the context length

Figure 3(a) follows the error over rollouts of four context lengths on all five families. The curves differ in level and in growth rate. The baselines' error is above HFluxNO's from the first step on every family but shallow water. Beyond one context length it also grows faster on viscous Burgers and 2D cubic, and on Euler up to two context lengths. The growth is carried by high-frequency error that accumulates from step to step, the failure mode of one-step neural surrogates analyzed by Lippe et al. (2023). In our rollouts it appears as oscillations that spread outward from the sharpest features of the solution (Section G). The HFluxNO error grows slowly and without such artifacts over the whole horizon on those three families, and on Euler up to about two context lengths, beyond which it grows faster than the baselines' and overtakes them by $t = 0.4$. Nevertheless, for the majority of the rollouts, we find that the HFluxNO error is primarily concentrated at the moving fronts, such as the thin doublet-shaped error bands in Fig. 9. This reaffirms that HFluxNO's errors primarily stem the accumulation of small front-speed errors, instead of global distortions of the field shapes as in the case of the baselines Section G).

Over long rollouts, the Euler system (Fig. 3(b)) is the most demanding family. It has four coupled conserved variables on a 128^2 grid in the transonic regime, with the ratio of specific heats to be inferred from the context. By one context length, DPOT's prediction has broken into blocks aligned with its patch grid and DISCO's is covered by ringing along every front, while HFluxNO reproduces

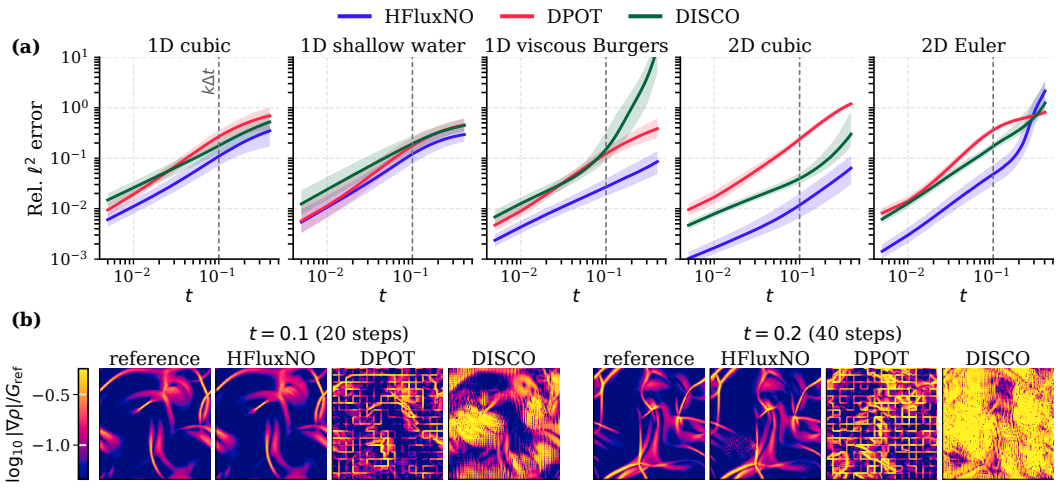


Figure 3: (a) Relative ℓ^2 error per time step, $\|\hat{\boldsymbol{u}}^n - \boldsymbol{u}^n\|_2 / \|\boldsymbol{u}^n\|_2$, over rollouts of 80 steps on the in-distribution test sets. Line: median over three training runs and all test trajectories; band: 25th to 75th percentile. The dashed line marks one context length ($k\Delta t = 0.1$). (b) Numerical schlieren, in the density-gradient convention (Luthman et al., 2019). The colors show $\log_{10} |\nabla\rho|$ normalized by the reference maximum over both times, of one in-distribution 2D Euler rollout at one and two context lengths (20 and 40 steps). Shared color scale for all panels; shock fronts appear bright.

the reference's shock and compression structure. At two context lengths the baselines no longer resemble the solution. HFluxNO still does, but a bright patch of grid-scale texture has appeared behind its strongest front (Fig. 3(b)). Beyond this horizon the patch grows into an oscillation, which Section G documents and relates to the odd-even decoupling of low-dissipation classical fluxes.

7 Limitations

Our experiments are confined to periodic Cartesian grids. Non-periodic boundaries would enter through the padding of the flux stencil, as in classical ghost-cell methods, but we have not tested them. The encoder's positional encoding is tied to the training grid, so the model does not transfer across resolutions, and the time step is fixed at its training value. Beyond conservation, stability or admissibility must be learned through training and are not guaranteed by construction. On the Euler system it under-dissipates at strong compressions and starts developing grid-scale oscillations at longer rollouts (Section G); and on the shallow-water system it loses to the baselines when the height profiles are shock-dominated; both settings contain states outside the range seen in training, and an entropy-stable or monotone parametrization of the learned flux is the natural remedy. Finally, the 2D cubic reference solutions are first order in space, and the families studied, while spanning scalar and vector-valued laws in one and two dimensions, are free of source terms.

8 Conclusion

We have posed in-context flux learning for families of hyperbolic conservation laws and introduced HFluxNO, which infers a conservative numerical flux operator from a short observed trajectory. Each of its three components contributes, and none can be replaced by a generic alternative without loss. The model outperforms recent in-context solvers in and out of distribution at the horizon of multiple context lengths, including on a flux family never seen in training and on the challenging two-dimensional Euler system. We take this as evidence for the principle stated in the introduction: broad adaptability is best built within a mathematically coherent class whose numerical structure the model can inherit. The natural next steps are a learned flux with built-in dissipation and admissibility guarantees, extensions to source terms, non-periodic boundaries and resolution transfer, and the assembly of class-scoped models of this kind into broader ones.

AI USE STATEMENT

We used generative AI tools in a limited role for implementing methods: most of the code was written by the authors, and AI-generated code was reviewed and verified by the authors. We have not used generative AI tools for any other task requiring disclosure. Additionally, we used generative AI tools for debugging code and for improving the readability of author-written drafts. We have reviewed all AI-assisted work and take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Code is available at `https://anonymous.4open.science/r/in_context_flux_no/`. The architecture is described in Sections A and 4, dataset generation in Section B (Table 3), model configurations, baselines, and training details in Section C, and the evaluation protocol in Section 5. All tabulated results are mean $\pm$ std over three seeds, with full numbers in Section F.

REFERENCES

Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. ViViT: A Video Vision Transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6836–6846, 2021.

Dinshaw S. Balsara. Multidimensional HLLE Riemann solver: Application to Euler and magnetohydrodynamic flows. *J. Comput. Phys.*, 229(6):1970–1993, 2010. ISSN 0021-9991. doi: 10.1016/j.jcp.2009.11.018.

Yohai Bar-Sinai, Stephan Hoyer, Jason Hickey, and Michael P. Brenner. Learning data-driven discretizations for partial differential equations. *Proc. Natl. Acad. Sci.*, 116(31):15344–15349, 2019. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.1814058116.

Jacob Beck, Matthew Thomas Jackson, Risto Vuorio, and Shimon Whiteson. Hypernetworks in Meta-Reinforcement Learning. In *Proceedings of The 6th Conference on Robot Learning*, pp. 1478–1487. PMLR, March 2023.

Deniz A. Bezgin, Steffen J. Schmidt, and Nikolaus A. Adams. WENO3-NN: A maximum-order three-point data-driven weighted essentially non-oscillatory scheme. *J. Comput. Phys.*, 452: 110920, 2022. ISSN 0021-9991. doi: 10.1016/j.jcp.2021.110920.

Rafael Bischof, Michal Piovarci, Michael Anton Kraus, Siddhartha Mishra, and Bernd Bickel. HyPINO: Multi-Physics Neural Operators via HyperPINNs and the Method of Manufactured Solutions. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, October 2025.

Aleksandar Botev, Soham De, Samuel L. Smith, Anushan Fernando, George-Cristian Muraru, Ruba Haroun, Leonard Berrada, Razvan Pascanu, Pier Giuseppe Sessa, Robert Dadashi, Léonard Hussenot, Johan Ferret, Sertan Girgin, Olivier Bachem, Alek Andreev, Kathleen Kenealy, Thomas Mesnard, Cassidy Hardin, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Armand Joulin, Noah Fiedel, Evan Senter, Yutian Chen, Srivatsan Srinivasan, Guillaume Desjardins, David Budden, Arnaud Doucet, Sharad Vikram, Adam Paszke, Trevor Gale, Sebastian Borgeaud, Charlie Chen, Andy Brock, Antonia Paterson, Jenny Brennan, Meg Risdal, Raj Gundluru, Nesh Devanathan, Paul Mooney, Nilay Chauhan, Phil Culliton, Luiz Gustavo Martins, Elisa Bandy, David Huntsperger, Glenn Cameron, Arthur Zucker, Tris Warkentin, Ludovic Peran, Minh Giang, Zoubin Ghahramani, Clément Farabet, Koray Kavukcuoglu, Demis Hassabis, Raia Hadsell, Yee Whye Teh, and Nando de Frietas. RecurrentGemma: Moving Past Transformers for Efficient Open Language Models, 2024.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/jax-ml/jax`.

Johannes Brandstetter, Daniel E. Worrall, and Max Welling. Message Passing Neural PDE Solvers. In *International Conference on Learning Representations*, 2022.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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 Ziegler, Jeffrey Wu, Clemens Winter, Chris 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. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020.

Yadi Cao, Yuxuan Liu, Liu Yang, Rose Yu, Hayden Schaeffer, and Stanley Osher. VICON: Vision In-Context Operator Networks for Multi-Physics Fluid Dynamics Prediction. *Trans. Mach. Learn. Res.*, 2026. ISSN 2835-8856.

Oscar Chang, Lampros Flokas, and Hod Lipson. Principled Weight Initialization for Hypernetworks. In *International Conference on Learning Representations*, September 2019.

Philip Charles and Deep Ray. Learning WENO for Entropy Stable Schemes to Solve Conservation Laws. *SIAM J. Sci. Comput.*, 47(6):C1196–C1222, 2025. ISSN 1064-8275. doi: 10.1137/24M1697566.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural Ordinary Differential Equations. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Zhen Chen, Anne Gelb, and Yoonsang Lee. Learning the Dynamics for Unknown Hyperbolic Conservation Laws Using Deep Neural Networks. *SIAM Journal on Scientific Computing*, 46(2): A825–A850, April 2024. ISSN 1064-8275. doi: 10.1137/22M1537333.

Mengdi Chu, Yang Liu, Ayan Biswas, and Han-Wei Shen. Do Physics Foundation Models Learn Generalizable Physics? A Bias-Aware Benchmark Across Physical Regimes and Distribution Shifts, July 2026.

Soham De, Samuel L. Smith, Anushan Fernando, Aleksandar Botev, George Cristian-Muraru, Albert Gu, Ruba Haroun, Leonard Berrada, Yutian Chen, Srivatsan Srinivasan, Guillaume Desjardins, Arnaud Doucet, David Budden, Yee Whye Teh, Razvan Pascanu, Nando De Freitas, and Caglar Gulcehre. Griffin: Mixing Gated Linear Recurrences with Local Attention for Efficient Language Models, 2024.

Niccolò Discacciati, Jan S. Hesthaven, and Deep Ray. Controlling oscillations in high-order Discontinuous Galerkin schemes using artificial viscosity tuned by neural networks. *J. Comput. Phys.*, 409:109304, 2020. ISSN 0021-9991. doi: 10.1016/j.jcp.2020.109304.

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. In *International Conference on Learning Representations*, 2020.

Michael Dumbser, Jean-Marc Moschetta, and Jérémie Gressier. A matrix stability analysis of the carbuncle phenomenon. *J. Comput. Phys.*, 197(2):647–670, 2004. ISSN 0021-9991. doi: 10.1016/j.jcp.2003.12.013.

John Guibas, Morteza Mardani, Zongyi Li, Andrew Tao, Anima Anandkumar, and Bryan Catanzaro. Efficient Token Mixing for Transformers via Adaptive Fourier Neural Operators. In *International Conference on Learning Representations*, 2021.

Jan Hagnberger, Marimuthu Kalimuthu, Daniel Musekamp, and Mathias Niepert. Vectorized Conditional Neural Fields: A Framework for Solving Time-dependent Parametric Partial Differential Equations. In *Proc. 41st Int. Conf. Mach. Learn.*, pp. 17189–17223. PMLR, 2024.

Derek Hansen, Danielle C. Maddix, Shima Alizadeh, Gaurav Gupta, and Michael W. Mahoney. Learning Physical Models that Can Respect Conservation Laws. In *Proc. 40th Int. Conf. Mach. Learn.*, pp. 12469–12510. PMLR, 2023.

Zhongkai Hao, Chang Su, Songming Liu, Julius Berner, Chengyang Ying, Hang Su, Anima Anandkumar, Jian Song, and Jun Zhu. DPOT: Auto-Regressive Denoising Operator Transformer for Large-Scale PDE Pre-Training. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 17616–17635. PMLR, 2024.

Christoph Jürgen Hemmer and Daniel Durstewitz. True Zero-Shot Inference of Dynamical Systems Preserving Long-Term Statistics. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 13497–13540. Curran Associates, Inc., 2025. doi: 10.52202/085713-0455.

Maximilian Herde, Bogdan Raonić, Tobias Rohner, Roger Käppeli, Roberto Molinaro, Emmanuel de Bézenac, and Siddhartha Mishra. Poseidon: Efficient Foundation Models for PDEs. In *Advances in Neural Information Processing Systems*, volume 37, pp. 72525–72624, 2024.

Masanobu Horie and Naoto Mitsume. Graph Neural PDE Solvers with Conservation and Similarity-Equivariance. In *Proc. 41st Int. Conf. Mach. Learn.*, pp. 18785–18814. PMLR, 2024.

Matthias Karlbauer, Timothy Praditia, Sebastian Otte, Sergey Oladyshkin, Wolfgang Nowak, and Martin V. Butz. Composing Partial Differential Equations with Physics-Aware Neural Networks. In *Proc. 39th Int. Conf. Mach. Learn.*, pp. 10773–10801. PMLR, 2022.

David I. Ketcheson, Kyle T. Mandli, Aron J. Ahmadia, Amal Alghamdi, Manuel Quezada de Luna, Matteo Parsani, Matthew G. Knepley, and Matthew Emmett. PyClaw: Accessible, Extensible, Scalable Tools for Wave Propagation Problems. *SIAM Journal on Scientific Computing*, 34(4): C210–C231, November 2012.

Patrick Kidger, James Morrill, James Foster, and Terry Lyons. Neural Controlled Differential Equations for Irregular Time Series. In *Adv. Neural Inf. Process. Syst.*, volume 33, pp. 6696–6707. Curran Associates, Inc., 2020.

Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. Reversible Instance Normalization for Accurate Time-Series Forecasting against Distribution Shift. In *International Conference on Learning Representations*, 2022.

Taeyoung Kim and Myungjoo Kang. Approximating Numerical Fluxes Using Fourier Neural Operators for Hyperbolic Conservation Laws. *Commun. Comput. Phys.*, 37(2):420–456, 2025. ISSN 1991-7120, 1815-2406. doi: 10.4208/cicp.OA-2024-0123.

Taeyoung Kim, Youngsoo Ha, and Myungjoo Kang. Neural operators learn the local physics of magnetohydrodynamics. *Computers & Fluids*, 297:106661, 2025. ISSN 0045-7930. doi: 10.1016/j.compfluid.2025.106661.

Dmitrii Kochkov, Jamie A. Smith, Ayya Alieva, Qing Wang, Michael P. Brenner, and Stephan Hoyer. Machine learning–accelerated computational fluid dynamics. *Proc. Natl. Acad. Sci.*, 118(21): e2101784118, 2021. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.2101784118.

Armand Kassaï Koupaï, Lise Le Boudec, Louis Serrano, and Patrick Gallinari. ENMA: Tokenwise Autoregression for Continuous Neural PDE Operators. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural Operator: Learning Maps Between Function Spaces With Applications to PDEs. *Journal of Machine Learning Research*, 24(89):1–97, 2023. ISSN 1533-7928.

Jeffrey Lai, Anthony Bao, and William Gilpin. Panda: A pretrained forecast model for chaotic dynamics. In *International Conference on Learning Representations*, volume 2026, pp. 104651–104695, April 2026.

Samuel Lanthaler, Roberto Molinaro, Patrik Hadorn, and Siddhartha Mishra. Nonlinear Reconstruction for Operator Learning of PDEs with Discontinuities. In *The Eleventh International Conference on Learning Representations*, 2023.

Stig Larsson and Vidar Thomée. *Partial Differential Equations with Numerical Methods*. Number 45 in Texts in Applied Mathematics. Springer, Berlin Heidelberg, first softcover printing edition, 2009. ISBN 978-3-540-88706-5 978-3-540-88705-8.

Randall J. LeVeque. Wave Propagation Algorithms for Multidimensional Hyperbolic Systems. *J. Comput. Phys.*, 131(2):327–353, 1997. ISSN 0021-9991. doi: 10.1006/jcph.1996.5603.

Randall J. LeVeque. *Finite Volume Methods for Hyperbolic Problems*. Cambridge University Press, Cambridge, 2002. ISBN 978-0-521-00924-9.

Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier Neural Operator for Parametric Partial Differential Equations. In *International Conference on Learning Representations*, October 2020.

Nathan Lichtlé, Alexi Canesse, Zhe Fu, Hossein Nick Zinat Matin, Maria Laura Delle Monache, and Alexandre M. Bayen. (U)NFV: (Un)Supervised Neural Finite Volume Methods for Solving Hyperbolic PDEs. In *The Fourteenth International Conference on Learning Representations*, 2026.

Phillip Lippe, Bas Veeling, Paris Perdikaris, Richard Turner, and Johannes Brandstetter. PDE-Refiner: Achieving Accurate Long Rollouts with Neural PDE Solvers. In *Advances in Neural Information Processing Systems*, volume 36, pp. 67398–67433, 2023.

Lizuo Liu, Lu Zhang, and Anne Gelb. Parametric Hyperbolic Conservation Laws: A Unified Framework for Conservation, Entropy Stability, and Hyperbolicity, 2026.

Ning Liu, Yiming Fan, Xianyi Zeng, Milan Klöwer, Lu Zhang, and Yue Yu. Harnessing the Power of Neural Operators with Automatically Encoded Conservation Laws. In *Proc. 41st Int. Conf. Mach. Learn.*, pp. 30965–30997. PMLR, 2024.

Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*, 2019.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, March 2021. ISSN 2522-5839. doi: 10.1038/s42256-021-00302-5.

Elizabeth Luthman, Niccolo Cymbalist, Daniel Lang, Graham Candler, and Paul Dimotakis. Simulating schlieren and shadowgraph images from LES data. *Exp. Fluids*, 60(8):134, 2019. ISSN 1432-1114. doi: 10.1007/s00348-019-2774-6.

Jim Magiera, Deep Ray, Jan S. Hesthaven, and Christian Rohde. Constraint-aware neural networks for Riemann problems. *J. Comput. Phys.*, 409:109345, 2020. ISSN 0021-9991. doi: 10.1016/j.jcp.2020.109345.

Michael McCabe, Bruno Régaldo-Saint Blancard, Liam Parker, Ruben Ohana, Miles Cranmer, Alberto Bietti, Michael Eickenberg, Siavash Golkar, Geraud Krawezik, Francois Lanusse, Mariel Pettee, Tiberiu Tesileanu, Kyunghyun Cho, and Shirley Ho. Multiple Physics Pretraining for Spatiotemporal Surrogate Models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 119301–119335, 2024.

Michael McCabe, Payel Mukhopadhyay, Tanya Marwah, Bruno Régaldo-Saint Blancard, François Rozet, Cristiana Diaconu, Lucas Thibaut Meyer, Kaze W. K. Wong, Mohammad-Hadi Sotoudeh, Alberto Bietti, Irina Espejo Morales, Rio Alexa Fear, Siavash Golkar, Tom Hehir, Keiya Hirashima, Geraud Krawezik, Francois Lanusse, Rudy Morel, Ruben Ohana, Liam Holden Parker, Mariel Pettee, Jeff Shen, Kyunghyun Cho, Miles Cranmer, and Shirley Ho. Walrus: A Cross-domain Foundation Model for Continuum Dynamics. In *Forty-Third International Conference on Machine Learning*, 2026.

Rudy Morel, Jiequn Han, and Edouard Oyallon. DISCO: Learning to DISCover an evolution Operator for multi-physics-agnostic prediction. In *Forty-Second International Conference on Machine Learning*, 2025.

Sebastian Müller, Lennart Schüler, Alraune Zech, and Falk Heße. GSTools v1.3: A toolbox for geostatistical modelling in Python. *Geosci. Model Dev.*, 15(7):3161–3182, 2022. ISSN 1991-9603. doi: 10.5194/gmd-15-3161-2022.

Jose Javier Gonzalez Ortiz, John Guttag, and Adrian V. Dalca. Magnitude Invariant Parametrizations Improve Hypernetwork Learning. In *The Twelfth International Conference on Learning Representations*, October 2023.

Viorica Patraucean, Xu Owen He, Joseph Heyward, Chuhan Zhang, Mehdi S. M. Sajjadi, George-Cristian Muraru, Artem Zholus, Mahdi Karami, Ross Goroshin, Yutian Chen, Simon Osindero, Joao Carreira, and Razvan Pascanu. TRecViT: A Recurrent Video Transformer. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856.

Dimitrios G. Patsatzis, Mario di Bernardo, Lucia Russo, and Constantinos Siettos. GoRINNs: Godunov-Riemann informed neural networks for learning hyperbolic conservation laws. *J. Comput. Phys.*, 534:114002, 2025. ISSN 0021-9991. doi: 10.1016/j.jcp.2025.114002.

Ahmad Peyvan, Vivek Oommen, Ameya D. Jagtap, and George Em Karniadakis. RiemannONets: Interpretable neural operators for Riemann problems. *Computer Methods in Applied Mechanics and Engineering*, 426:116996, June 2024. ISSN 0045-7825. doi: 10.1016/j.cma.2024.116996.

James J. Quirk. A contribution to the great Riemann solver debate. *Int. J. Numer. Methods Fluids*, 18(6):555–574, 1994. ISSN 1097-0363. doi: 10.1002/fld.1650180603.

Ashiqur Rahman, Robert J. George, Mogab Elleithy, Daniel Leibovici, Zongyi Li, Boris Bonev, Colin White, Julius Berner, Raymond A. Yeh, Jean Kossaifi, Kamyar Azizzadenesheli, and Anima Anandkumar. Pretraining Codomain Attention Neural Operators for Solving Multiphysics PDEs. In *Adv. Neural Inf. Process. Syst.*, volume 37, pp. 104035–104064, 2024. doi: 10.52202/079017-3306.

M. Raissi, P. Perdikaris, and G. E. 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, February 2019. ISSN 0021-9991. doi: 10.1016/j.jcp.2018.10.045.

Deep Ray and Jan S. Hesthaven. An artificial neural network as a troubled-cell indicator. *J. Comput. Phys.*, 367:166–191, 2018. ISSN 0021-9991. doi: 10.1016/j.jcp.2018.04.029.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi (eds.), *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pp. 234–241, Cham, 2015. Springer International Publishing. ISBN 978-3-319-24574-4. doi: 10.1007/978-3-319-24574-4_28.

V.V Rusanov. The calculation of the interaction of non-stationary shock waves and obstacles. *USSR Comput. Math. Math. Phys.*, 1(2):304–320, 1962. ISSN 00415553. doi: 10.1016/0041-5553(62)90062-9.

Louis Serrano, Armand Kassaï Koupaï, Thomas X. Wang, Pierre Erbacher, and Patrick Gallinari. Zebra: In-Context Generative Pretraining for Solving Parametric PDEs. In *Forty-Second International Conference on Machine Learning*, 2025.

Chi-Wang Shu. High Order Weighted Essentially Nonoscillatory Schemes for Convection Dominated Problems. *SIAM Rev.*, 51(1):82–126, 2009. ISSN 0036-1445, 1095-7200. doi: 10.1137/070679065.

Ben Stevens and Tim Colonius. Enhancement of shock-capturing methods via machine learning. *Theor. Comput. Fluid Dyn.*, 34(4):483–496, 2020. ISSN 1432-2250. doi: 10.1007/s00162-020-00531-1.

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. In *Advances in Neural Information Processing Systems*, volume 36, pp. 71242–71262. Curran Associates, Inc., 2023.

Makoto Takamoto, Francesco Alesiani, and Mathias Niepert. Learning Neural PDE Solvers with Parameter-Guided Channel Attention. In *Proc. 40th Int. Conf. Mach. Learn.*, pp. 33448–33467. PMLR, 2023.

Mingtian Tan, Mike A. Merrill, Vinayak Gupta, Tim Althoff, and Thomas Hartvigsen. Are Language Models Actually Useful for Time Series Forecasting? In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, November 2024.

Eleuterio F. Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction*. Springer, Berlin, Heidelberg, 3 edition, 2009. ISBN 978-3-540-25202-3. doi: 10.1007/b79761.

Kiwon Um, Robert Brand, Yun (Raymond) Fei, Philipp Holl, and Nils Thuerey. Solver-in-the-Loop: Learning from Differentiable Physics to Interact with Iterative PDE-Solvers. In *Adv. Neural Inf. Process. Syst.*, volume 33, pp. 6111–6122. Curran Associates, Inc., 2020.

Zongzhe Xu, Ritvik Gupta, Wenduo Cheng, Alexander Shen, Junhong Shen, Ameet Talwalkar, and Mikhail Khodak. Specialized Foundation Models Struggle to Beat Supervised Baselines. In *The Thirteenth International Conference on Learning Representations*, October 2025.

Liu Yang and Stanley J. Osher. PDE generalization of in-context operator networks: A study on 1D scalar nonlinear conservation laws. *Journal of Computational Physics*, 519:113379, 2024. ISSN 0021-9991. doi: 10.1016/j.jcp.2024.113379.

Liu Yang, Siting Liu, Tingwei Meng, and Stanley J. Osher. In-context operator learning with data prompts for differential equation problems. *Proc. Natl. Acad. Sci.*, 120(39):e2310142120, 2023. doi: 10.1073/pnas.2310142120.

Zhanhong Ye, Zining Liu, Bingyang Wu, Hongjie Jiang, Leheng Chen, Minyan Zhang, Xiang Huang, Qinghe Meng Jingyuan Zou, Hongsheng Liu, and Bin Dong. PDEformer-2: A Versatile Foundation Model for Two-Dimensional Partial Differential Equations, 2025.

Yuanzhao Zhang and William Gilpin. Zero-shot forecasting of chaotic systems. In *The Thirteenth International Conference on Learning Representations*, 2025.

Hang Zhou, Yuezhou Ma, Haixu Wu, Haowen Wang, and Mingsheng Long. Unisolver: PDE-Conditional Transformers Towards Universal Neural PDE Solvers. In *Proc. 42nd Int. Conf. Mach. Learn.*, pp. 79061–79088. PMLR, 2025.

A ARCHITECTURE DETAILS

Here, we provide additional information to augment the high-level architecture description given in Section 4. Hyperparameter values used for the experiments are collected in Section C.

A.1 CONTEXT ENCODER

Input and tokenization. The encoder receives the context window $\boldsymbol{U}^n \in \mathbb{R}^{k\times d\times N_1\times\cdots\times N_N}$. For all benchmarks except Euler, the cell-center coordinates are appended to each state as N additional channels, so that the encoder input has $d+N$ channels (see Section A.3 for the rationale). Each state is partitioned into non-overlapping patches of size $\kappa_1\times\cdots\times\kappa_N$, and each patch is mapped by a linear layer to an e-dimensional token, to which a learned positional encoding is added. This yields $P=\prod_\ell\lceil N_\ell/\kappa_\ell\rceil$ tokens per state and an initial representation $\boldsymbol{v}^{(0)}\in\mathbb{R}^{k\times P\times e}$, indexed by window time $\tau=1,\dots,k$ and patch $j=1,\dots,P$. The positional encoding is defined for a fixed grid; the encoder is therefore grid-specific, and we leave generalization across resolutions to a future study (Section 7).

Temporal block. For each patch j, the token sequence $\boldsymbol{z}=\boldsymbol{v}^{(m)}_{:,j}\in\mathbb{R}^{k\times e}$ is processed by the residual gated recurrent block $\mathrm{Rec}^{(m)}$ of Eq. (4),

$$\mathrm{Rec}(\boldsymbol{z})=\boldsymbol{z}+W_{\mathrm{o}}\Big[\mathrm{LRU}\big(\mathrm{Conv}_{\mathrm{c}}(W_{\mathrm{x}}\bar{\boldsymbol{z}})\big)\odot\mathrm{GeLU}(W_{\mathrm{y}}\bar{\boldsymbol{z}})\Big],\qquad \bar{\boldsymbol{z}}=\mathrm{RMSNorm}(\boldsymbol{z}),\tag{8}$$

applied row-wise in time. Here $W_{\mathrm{x}},W_{\mathrm{y}}\in\mathbb{R}^{w\times e}$ map tokens to a recurrent width w. $\mathrm{Conv}_{\mathrm{c}}$ is a depthwise one-dimensional convolution along time with kernel size 4 and left padding only, so that it is causal. LRU is the real-gated linear recurrent unit of De et al. (2024), which maps an input sequence $\boldsymbol{\eta}_1,\dots,\boldsymbol{\eta}_k\in\mathbb{R}^w$ to hidden states

$$\begin{aligned}&\boldsymbol{h}_\tau=\boldsymbol{a}_\tau\odot\boldsymbol{h}_{\tau-1}+\sqrt{1-\boldsymbol{a}_\tau^2}\odot\big(\boldsymbol{g}^{\mathrm{in}}_\tau\odot\boldsymbol{\eta}_\tau\big),\quad \boldsymbol{a}_\tau=\boldsymbol{a}^{\,\lambda\,\boldsymbol{g}^{\mathrm{rec}}_\tau},\\&\boldsymbol{g}^{\mathrm{rec}}_\tau=\mathrm{sigmoid}(W_{\mathrm{rec}}\boldsymbol{\eta}_\tau+\boldsymbol{b}_{\mathrm{rec}}),\quad \boldsymbol{g}^{\mathrm{in}}_\tau=\mathrm{sigmoid}(W_{\mathrm{in}}\boldsymbol{\eta}_\tau+\boldsymbol{b}_{\mathrm{in}}),\end{aligned}\tag{9}$$

where all operations, including the power, act elementwise. The decay $\boldsymbol{a}=\mathrm{sigmoid}(\boldsymbol{\Lambda})\in(0,1)^w$ is learnable through $\boldsymbol{\Lambda}$, the constant $\lambda=8$ follows De et al. (2024), and the gate matrices $W_{\mathrm{rec}},W_{\mathrm{in}}$ are block-diagonal as in Griffin. Finally, $W_{\mathrm{o}}\in\mathbb{R}^{e\times w}$ projects back to the token dimension. The recurrence is causal by construction: $\boldsymbol{h}_\tau$ depends only on $\boldsymbol{\eta}_1,\dots,\boldsymbol{\eta}_\tau$. Following TRecViT (Patraucean et al., 2025), the temporal block contains no feed-forward layer and feature mixing happens in the MLP of the spatial block.

Spatial block. For each window time τ, the P tokens $\widehat{\boldsymbol{v}}^{(m)}_{\tau,:}\in\mathbb{R}^{P\times e}$ are processed by the block $\mathrm{Attn}^{(m)}$ of Eq. (4), a standard pre-norm transformer encoder block (Dosovitskiy et al., 2020) with multi-head self-attention over the P tokens and a two-layer feed-forward network with GeLU activation. This mixes information across space at a single time and does not couple different times.

Readout. After L alternating temporal and spatial blocks, the tokens at the last window time are normalized by a LayerNorm and averaged over the P patches (Eq. (5)). Reading out at $\tau=k$ rather than pooling over time treats the terminal recurrent state as the summary of the observed trajectory, in the spirit of neural controlled differential equations (Kidger et al., 2020).

ViViT alternative (ablation). The spatio-temporal encoder ablated in Section 4.4 follows ViViT (Arnab et al., 2021) with tubelet embedding and joint spatio-temporal attention: the window is partitioned into space–time patches of size $\kappa_\tau\times\kappa_1\times\cdots\times\kappa_N$, each mapped to a token, and a standard transformer encoder attends over all $\lceil k/\kappa_\tau\rceil\cdot P$ tokens at once. The context vector is the LayerNorm-then-mean over all tokens. Tubelet embedding and joint self-attention treat time like a spatial axis, so this encoder does not respect the temporal order of the window. Both encoders are configured to matched parameter counts (Section C).

A.2 HYPERNETWORK

Trunk and head. The trunk in Eq. (6) is a one-hidden-layer MLP $\mathbb{R}^e\to\mathbb{R}^{d_{\mathrm{hyp}}}\to\mathbb{R}^e$ with GeLU activation. The head is linear, $\boldsymbol{\Theta}=W_{\mathrm{out}}\tilde{\boldsymbol{c}}+\boldsymbol{b}_{\mathrm{out}}$, where $\tilde{\boldsymbol{c}}=\mathrm{MLP}(\boldsymbol{c})$ is the trunk output. Since q

is of order 10^6 in our experiments, a dense $W_{\text{out}} \in \mathbb{R}^{q \times e}$ would dominate the parameter count. We therefore partition $\boldsymbol{\Theta}$ into B contiguous blocks and $\tilde{\boldsymbol{c}}$ into B segments of size e/B, and let W_{out} be block-diagonal, mapping the b-th segment to the b-th block. This reduces the parameter count from qe to qe/B; the same device is used in adaptive Fourier neural operators (Guibas et al., 2021) and in the gates of Eq. (9) (De et al., 2024).

Initialization. A hypernetwork's parameters must be initialized so that the target network it produces is itself well initialized; a standard scheme applied to W_{out} does not achieve this and can make the target unstable to train (Chang et al., 2019; Ortiz et al., 2023; Beck et al., 2023). We use Bias-HyperInit (Beck et al., 2023):

- the trunk is initialized with the framework default;
- W_{out} is initialized to zero;
- $\boldsymbol{b}_{\text{out}}$ is set to the flattened parameters of a target network drawn from its own default initialization.

Consequently $H(\boldsymbol{c}) = \boldsymbol{b}_{\text{out}}$ for every $\boldsymbol{c}$ at initialization: training starts from a conventionally initialized Flux NO that is independent of the context, and the dependence on $\boldsymbol{c}$ is learned as a perturbation around it.

Scale normalization (Euler only). For the two-dimensional Euler benchmark the conserved variables $(\rho, \rho u, \rho v, E)$ differ in magnitude, and we found it beneficial to normalize them, in the manner of reversible instance normalization (Kim et al., 2022). Let $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_d)$ with $\sigma_\mu = \left(\langle u_\mu^2 \rangle_{\tau, \boldsymbol{i}} + \epsilon\right)^{1/2}$ be the root-mean-square of each conserved variable over the cells and times of the context window, with $\epsilon = 10^{-6}$. Two changes are made: the trunk input becomes $[\boldsymbol{c}; \log \boldsymbol{\sigma}] \in \mathbb{R}^{e+d}$, and the target network operates on $\boldsymbol{u}^n/\boldsymbol{\sigma}$ with its output multiplied by $\boldsymbol{\sigma}$, so that the numerical flux it produces is $\boldsymbol{\sigma} \odot G_{\boldsymbol{\Theta}}(\cdot/\boldsymbol{\sigma})$. Since $\boldsymbol{\sigma}$ is constant over the grid, the update remains a flux difference and the telescoping argument of Section 4.3 applies unchanged. The coordinate channels are omitted for this benchmark.

A.3 Target network

Stencil and padding. The neural flux Eq. (7) is parametrized by three stencil widths: $r + 1$ cells behind and s cells ahead of the face along the normal direction, as in Eq. (3), and $s_\perp$ cells to either side along each transverse direction, so that $|\mathcal{S}_\ell(\boldsymbol{i})| = (r + s + 1)(2s_\perp + 1)^{N-1}$. To evaluate the fluxes through all faces normal to ℓ, the state is padded periodically by $(r + 1, s)$ cells along axis ℓ, which yields $N_\ell + 1$ faces, and by $(s_\perp, s_\perp)$ cells along every other axis. Periodic padding is the only boundary treatment used in this work; other boundary conditions would be imposed by changing the padding, as in classical ghost-cell methods, without modifying the network.

Convolution and MLP. A single convolution with kernel size $r + s + 1$ along the first axis and $2s_\perp + 1$ along the remaining $N - 1$ axes, with d_{conv} output channels, is applied to the padded state after transposing axis ℓ to the front; transposing back gives one d_{conv}-dimensional feature per face. A channel-wise MLP with D hidden layers of width d_{mlp} and GeLU activations maps each feature to $N \cdot d$ outputs, of which the ℓ-th block of d is $G_{\boldsymbol{\Theta}}^{(\ell)}$, the flux $\hat{\boldsymbol{f}}^{(\ell)}$. Thus one kernel and one MLP trunk are shared by all directions, with direction-specific output heads. All N directional fluxes are computed from the same state $\boldsymbol{u}^n$, and the state is advanced by Eq. (2) with Δt and Δx_ℓ supplied as inputs. In one dimension this reduces to a Conv1D with kernel $r + s + 1$ producing $N_1 + 1$ face features, followed by the MLP.

Coordinate channels. For all benchmarks except Euler, the cell-center coordinates $\boldsymbol{x}_{\boldsymbol{i}}$ are appended to the state as N additional input channels before the convolution, so that the flux may depend on position. This follows the convention of Fourier neural operators, whose lifting layer takes $(\boldsymbol{u}(\boldsymbol{x}), \boldsymbol{x})$ as input (Li et al., 2020), which the Flux Neural Operator of Kim & Kang (2025) inherits. For the periodic, spatially homogeneous problems studied here the channels are not required; we retain them for compatibility with spatially varying fluxes or non-periodic boundaries, which we do not study. The coordinate channels are padded periodically together with the state,

so that the faces at either end of an axis receive identical inputs and the telescoping argument of Section 4.3 holds exactly; the measured drift in Section E is at floating-point precision.

B BENCHMARK DATASETS

This appendix is the single source of the generation protocol for the families of Table 2. Every trajectory is defined by one coefficient draw and one initial condition; the coefficients are stored as metadata and never supplied to any model. All problems are posed on the periodic domain $[0, 1]^N$ with uniformly spaced cell centers and $\Delta x_\ell = 1/N_\ell$.

B.1 COMMON PROTOCOL

Table 3 lists, per family, the grid, the number of saved snapshots, the split sizes, and the size of each out-of-distribution set. Coefficients are drawn independently for the training, validation, and test splits, so no test coefficient appears in training. Reference solutions are computed with PyClaw (Ketcheson et al., 2012) for the 1D cubic, shallow-water, and Euler families, and with our own explicit finite-volume solvers, written in JAX, for the viscous Burgers and 2D cubic families. Random fields are generated with GSTools (Müller et al., 2022) or by spectral filtering, as stated below. Trajectories containing non-finite values are identified at generation time and removed from every split; the number removed is reported in the last column of Table 3. column of Table 3. Only the shallow-water family is affected, where about 10% of the generated trajectories are removed due to height positivity violations during the solve stemming from problematic ill-sampled initial conditions.

Table 3: Generation protocol per family. Splits are numbers of coefficient draws; each draw is paired with N_{init} independent initial conditions, so a split of 1000 draws with $N_{\text{init}} = 100$ contains 10^5 trajectories. Every out-of-distribution (OOD) set uses 100 draws $\times$ 100 initial conditions, except for 2D Euler, which uses 10 draws $\times$ 100 initial conditions. *: 2D Cubic uses 500 initial conditions for the training set only.

Family	Grid	Δt	T	N_t	Train / val / test draws	N_{init}	Removed
1D cubic	100	0.005	0.4	80	1000 / 100 / 100	100	0
1D shallow water	100	0.005	0.4	80	1000 / 100 / 100	100	12316
1D viscous Burgers	100	0.005	0.4	80	1000 / 100 / 100	100	0
2D cubic	50^2	0.005	0.4	80	1000 / 100 / 100	100^*	0
2D Euler	128^2	0.005	0.4	80	20 / 10 / 10	100	0

Shock-dominated initial conditions. All shock-dominated OOD sets replace the training initial-condition family by periodic random step functions: the number of steps is drawn uniformly from $\{1, \ldots, 5\}$ in one dimension, the step locations are drawn uniformly on the domain, and each plateau value is drawn uniformly from an interval stated per family below. In two dimensions the domain is partitioned into the Voronoi cells of seed points drawn uniformly on the periodic domain, and each cell is assigned a constant value for the field variables. Number of cells used per family is stated below.

B.2 1D CUBIC CONSERVATION LAW

Equation and parameters. $u_t + f(u)_x = 0$ with $f(u) = c_1 u + c_2 u^2 + c_3 u^3$ and $(c_1, c_2, c_3) \sim \mathcal{U}[-1, 1]^3$, following the family introduced by Yang & Osher (2024).

Initial conditions. Mean-zero periodic Gaussian random fields with covariance $k(x, x') = \exp\big(-(1 - \cos 2\pi(x - x'))\big)$.

Reference solver. PyClaw with a scalar Riemann solver written for the cubic flux, the MC total-variation-diminishing limiter, one wave family, target CFL number 0.5, and maximum CFL number 0.9.

OOD sets. (i) Shock-dominated initial conditions with plateau values in $[-1, 1]$ and the training coefficient distribution. (ii) The sine flux $f(u) = a\sin(bu)$, $(a, b) \sim \mathcal{U}[-1, 1]^2$, with the training GRF initial conditions. (iii) The sine flux with shock-dominated initial conditions. Models trained on the cubic family are evaluated on all three without retraining.

B.3 1D SHALLOW-WATER SYSTEM

Equation and parameters. With h the water height, $m = hu$ the momentum, and $\boldsymbol{q} = (h, m)^\top$,

$$\boldsymbol{q}_t + \boldsymbol{F}(\boldsymbol{q})_x = \mathbf{0}, \qquad \boldsymbol{F}(\boldsymbol{q}) = \begin{pmatrix} \alpha\, m \\ \gamma\, m^2/h + \frac{1}{2}\beta h^2 \end{pmatrix}, \qquad \alpha, \gamma \sim \mathcal{U}[0.5, 1.5],\ \beta \sim \mathcal{U}[8, 12]. \quad (10)$$

The standard shallow-water equations are recovered at $(\alpha, \gamma, \beta) = (1, 1, g)$.

Initial conditions. $m(0)$ is a Gaussian random field with Gaussian covariance $k(x, x') = \sigma^2 \exp\big(-s^2|x - x'|^2/\ell^2\big)$, $\sigma^2 = 0.5$, $\ell = 0.3$, $s = \sqrt{\pi}/2$; $h(0)$ is a lognormal random field with the same covariance, which keeps the height positive. Both are generated on the periodic lattice with GSTools. A height floor $h_{\min} = 10^{-8}$ is used in divisions and square roots.

Reference solver. PyClaw with a Roe-type approximate Riemann solver, the MC limiter, two wave families, target CFL number 0.5, and maximum CFL number 0.9.

OOD set. Shock-dominated $h(0)$ with plateau values in $[0.5, 4.5]$, which respects positivity; $m(0)$ keeps its training distribution, since step functions in both fields produce excessively irregular solutions.

B.4 1D VISCOUS BURGERS EQUATION

Equation and parameters. $u_t + (au^2)_x = \nu u_{xx}$ with $a \sim \mathcal{U}[0.5, 1.5]$ and $\nu \sim \mathcal{U}[0.005, 0.015]$. This is the one family with an explicit diffusive term; it is included as a controlled test outside the strictly hyperbolic class.

Initial conditions. As for the 1D cubic family.

Reference solver. An explicit finite-volume scheme in which the advective term uses the Rusanov flux and the diffusive term a second-order centered difference. The internal step is chosen adaptively from the advective and diffusive stability limits with CFL number 0.4, and the last step before each saved time is shortened so that snapshots fall exactly on the output grid.

OOD set. Shock-dominated initial conditions with plateau values in $[-1, 1]$.

B.5 2D CUBIC CONSERVATION LAW

Equation and parameters. $u_t + f(u)_x + f(u)_y = 0$ with the cubic f of Section B.2 applied in both directions and $(c_1, c_2, c_3) \sim \mathcal{U}[-1, 1]^3$.

Initial conditions. Spectral Gaussian random fields: white noise ξ on the grid is filtered in Fourier space by $\hat{G}(k_x, k_y) = \exp\big(-\frac{\ell^2}{2}(k_x^2 + k_y^2)\big)$ with correlation length $\ell = 0.3$, i.e. $u_0 = \mathcal{F}^{-1}[\hat{G}\,\mathcal{F}[\xi]]$, followed by mean subtraction and normalization to unit variance.

Reference solver. A first-order finite-volume scheme with the Rusanov flux $\hat{f}(u_L, u_R) = \frac{1}{2}\big(f(u_L) + f(u_R)\big) - \frac{1}{2}\alpha(u_R - u_L)$, $\alpha = \max(|f'(u_L)|, |f'(u_R)|)$, applied in both directions in the update Eq. (2), with 32 substeps between consecutive saved times. We note that the solver is first order, so the reference trajectories can be more dissipative than those of the 1D families.

OOD set. Two-dimensional shock-dominated initial conditions (periodic Voronoi partitions with 2-5 cells, values in $[-1, 1]$) with the training coefficient distribution, solver, and grid unchanged.

B.6 2D EULER EQUATIONS

Equation and parameters. The compressible Euler equations for an ideal gas (LeVeque, 2002; Toro, 2009), in conserved variables $\boldsymbol{q} = (\rho, \rho u, \rho v, E)^\top$,

$$\boldsymbol{q}_t + \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(E+p) \end{pmatrix}_x + \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(E+p) \end{pmatrix}_y = \mathbf{0}, \qquad E = \tfrac{1}{2}\rho(u^2+v^2) + \frac{p}{\gamma - 1}, \tag{11}$$

where the ratio of specific heats γ is the only parameter. Since $\gamma = 1 + 2/f$ for a gas with $f \geq 3$ degrees of freedom, physically admissible values lie in $(1, 5/3]$; we sample $\gamma \sim \mathcal{U}[1.2, 1.5]$.

Initial conditions. Primitive variables are sampled and then converted to conserved variables. Each of $\tilde{\rho}, \tilde{u}, \tilde{v}, \tilde{p}$ is an independent mean-zero Gaussian random field with the Gaussian covariance of Section B.3, $\sigma = 1$, $\ell = 0.2$, $s = \sqrt{\pi}/2$, generated with GSTools, and the fields are shifted and scaled as $(\rho, u, v, p) = (1, 0, 0, 1) + (0.2, 0.6, 0.6, 0.2) \odot (\tilde{\rho}, \tilde{u}, \tilde{v}, \tilde{p})$. This keeps density and pressure positive in practice, with no trajectory removed (Table 3), and places the local Mach number $M = |\boldsymbol{v}|\sqrt{\rho/\gamma p}$ in the transonic regime $M \sim 1$.

Reference solver. PyClaw with the built-in Roe solver with sonic entropy fix (`euler_4wave_2D`) (LeVeque, 2002), with the minmod limiter, two transverse waves, and target and maximum CFL numbers of 0.5.

OOD set. Shock-dominated initial conditions in the primitive variables. One periodic Voronoi partition with 2-8 cells is drawn per initial condition, and each of ρ, u, v, and p is assigned a constant value per cell, drawn uniformly from $[0.6, 1.4]$ for ρ and p and $[-0.8, 0.8]$ for u and v. This keeps both density and pressure positive. The fields are then converted to conserved variables as for the training set, with the coefficient distribution, solver, and grid unchanged.

C MODEL AND TRAINING CONFIGURATIONS

Matching parameter counts. Baseline configurations start from the original code of each method and are adjusted to a parameter count comparable to HFluxNO; hyperparameters shared across models, such as the patch size and the embedding dimension, are set to the same value. Tables 4 and 5 list the settings per family and Table 7 the resulting parameter counts.

Table 4: HFluxNO configuration per benchmark family. Encoder notation as in Section A.1, target notation as in Section A.3; the hypernetwork trunk width is $d_{\text{hyp}} = 128$ throughout. The last two rows are the per-family switches of Section 4.

		1D families	2D cubic	2D Euler
Encoder	Patch size $(\kappa_1, \ldots, \kappa_N)$	4	5×5	8×8
	Embedding dim e	128	128	128
	Layers L	2	2	2
	Recurrent width w	128	128	128
	Attention heads	8	8	8
	Feed-forward width	64	64	64
	Hypernetwork blocks B	8	4	4
Target	Stencil (r, s)	(10, 10)	(8, 8)	(8, 8)
	Transverse half-width $s_\perp$	—	1	6
	Flux conv width d_{conv}	128	128	128
	MLP layers D / width d_{mlp}	4 / 128	5 / 128	4 / 128
Switches	Coordinate channels	on	on	off
	Scale normalization	off	off	on

Table 5: Baseline configurations. Values are given as 1D / 2D where they differ.

DPOT (Hao et al., 2024)		
	Patch size	4 / 8×8
	Embedding dim	128
	Layers	10
	Heads	8
	MLP hidden dim	512
	Frequency modes	12 / 10×10
DISCO (Morel et al., 2025)		
	Patch size	4 / 8×8
	Embedding dim	128
Encoder	Time–space blocks	3
	Heads	8
	MLP hidden dim	32
	U-Net channels	4 / 8
Target	GroupNorm groups	2 / 4
	ODE tolerances (rel. / abs.)	10^{-4} / 10^{-6}
	Max. solver steps	16
ICON (Yang & Osher, 2024) (1D only, Section D)		
	Condition / QoI length	100 / 100
	Number of demos	20
	Hidden dim	256
	Layers	6
	Heads	8

Configurations of the design-choice variants. The variants of Table 1 are all trained on the 1D cubic family under the training protocol below, and each is configured so that the component being varied has a parameter count close to that of the HFluxNO component it replaces; Table 6 lists the settings and Table 7 the totals. The two non-adaptive references (Q1) are a Flux NO with the target architecture of Table 4 but widened to a parameter count comparable to the in-context models, trained on all coefficient draws with a single parameter vector, and an FNO that takes the $k = 20$ stacked states of the window as input channels and outputs $\boldsymbol{u}^{n+1}$. The component substitutions (Q2) keep every other component at its HFluxNO setting. The ViViT encoder uses space–time patches of size 4×4 (time $\times$ space) and three transformer layers, chosen so that its parameter count matches the two-layer recurrent encoder. The concatenation variant removes the hypernetwork and instead appends the context vector $\boldsymbol{c}$ to the per-face feature vector produced by the flux convolution, so that the MLP of the target network receives $d_{\mathrm{conv}}+e$ inputs; its parameters are then trained directly rather than generated, which is why its total is lower. The FNO and U-Net targets replace the flux network by a network that maps $\boldsymbol{u}^n$ to $\boldsymbol{u}^{n+1}$ directly, with parameters generated by the same hypernetwork, and are sized to the same order as the Flux NO target.

Training. All models, including the variants of Table 1, were trained identically: on the mean squared error of the one-step prediction with context length $k = 20$, for 50,000 steps of AdamW (Loshchilov & Hutter, 2019) with the default weight decay of 10^{-4}. The learning rate rose linearly from zero to 5×10^{-5} over the first 5,000 steps and decayed to zero by cosine over the remaining 45,000. Gradient clipping was not used. The batch size was 256 for the 1D families and 64 for the 2D families.

Each configuration was trained with three random seeds, which set model initialization and the dataset shuffling. For evaluation, the checkpoint with the lowest validation loss was chosen. All reported numbers are the mean and standard deviation over the three runs. Training ran on a shared cluster of NVIDIA A40, A100, and H100 GPUs, so wall-clock times depend on the assigned device and the cluster load; as a guide, one HFluxNO or DPOT run took roughly 8 hours on a 1D family and 12 hours on a 2D family, and a DISCO or ICON run took two to three times longer, the former because of its adaptive ODE solver.

Table 6: Configurations of the variants in Table 1, all on the 1D cubic family. Settings not listed are those of the 1D column of Table 4. Parameter counts are for the component named in the first column; totals are in Table 7.

Variant	Setting	Value
Fixed Flux NO (Q1)	Stencil (r, s)	10
	Flux conv width / MLP width	512
	MLP layers D	6
FNO on window (Q1)	Input channels	$k \cdot d = 20$
	Lift dim / layers	128 / 10
	Frequency modes	12
ViViT encoder (Q2)	Patch size (time, space)	4×4
	Embedding dim e	128
	Layers / heads	3 / 8
	Feed-forward width	64
Concatenation (Q2)	Target MLP input	$d_{\text{conv}} + e = 256$
	Other target settings	as HFluxNO
FNO target (Q2)	Lift dim / MLP hidden dim	48 / 48
	Frequency modes	8
	Layers	4
U-Net target (Q2)	Channels	8
	GroupNorm groups	4
Flux NO target (reference)	as Table 4	—
TRecViT encoder (reference)	as Table 4	—

Table 7: Trainable parameter counts. Left: models compared in Section 6, per family. Right: the variants of Table 1, 1D cubic family.

Model	1D families	2D cubic	2D Euler	Variant	Total
DPOT	2.29M	3.76M	6.29M	Fixed Flux NO (Q1)	1.60M
DISCO	2.15M	6.59M	6.62M	FNO on window (Q1)	2.43M
HFluxNO	2.68M	6.00M	6.34M	ViViT encoder (Q2)	2.67M
				Concatenation (Q2)	2.12M
				FNO target (Q2)	3.22M
				U-Net target (Q2)	4.01M

D SNAPSHOT BASELINE: ICON

ICON (Yang et al., 2023; Yang & Osher, 2024) is a decoder-only transformer that reads a set of example input–output pairs from one system and predicts the output for a new input in a single forward pass. It adapts from context, as HFluxNO does, but it does not advance a state in time: the target is a snapshot at a fixed lead time τ, and there is no notion of a trajectory being rolled out. As such, we separate it from the autoregressive baselines of Section 5 and report its results here instead. ICON also tokenizes every spatial point individually, which limits it to one-dimensional data at the resolutions used in this work; extending it to two dimensions requires patch-based vision-transformer-style tokenization (Cao et al., 2026). We therefore evaluate ICON on the three one-dimensional families only.

Protocol. We use the authors' code and experimental setup with the configuration of Table 5, and train two variants on each family. ICON ($\tau = 0.005$) predicts one Δt ahead, the same lead time as the autoregressive models. To obtain a rollout we hold the sampled example pairs fixed and feed each prediction back as the next query, reaching $t = 0.1$ in 20 inferences, the horizon of the rollout metric in Section 5. ICON ($\tau = 0.1$) predicts 0.1 time units ahead in one shot, the setting of Yang & Osher (2024), so one inference reaches the same horizon and two reach $t = 0.2$. Errors are the relative ℓ^2 and ℓ^∞ norms of Section 5, with mean $\pm$ std over three training runs.

Results. Table 8 compares the two variants with HFluxNO at equal times. Feeding ICON's own output back as a query drives the $\tau = 0.005$ variant to order-one error by $t = 0.1$, so it cannot substitute for an autoregressive solver at the step size of Table 2. The $\tau = 0.1$ variant reaches $t = 0.1$ in a single inference with a mean ℓ^2 error comparable to that of HFluxNO's 20-step rollout, though on a few test trajectories the inference diverges, accounting for the spread in the table. A second inference on its own output, to $t = 0.2$, increases the ℓ^2 error by more than an order of magnitude. ICON ($\tau = 0.1$) is thus best read as a snapshot predictor at a fixed lead time rather than as a solver that can be stepped, which is why its results are kept apart from Section 6. The remaining evaluations use it at one inference, $t = 0.1$.

On the out-of-distribution sets of the cubic family (Table 9) and on the shallow-water and viscous Burgers families (Table 10), its errors follow the same pattern: the ℓ^∞ error is close to one wherever the solution is shock-dominated, indicating that front positions are not resolved. In the viscous Burgers experiments, ICON occasionally produced non-finite values in a single inference; these cases are excluded from Table 10, whereas divergent but finite predictions on the cubic family are included in Table 8. Both indicate an inference-time instability.

Table 8: ICON on the in-distribution 1D cubic test set, aligned by the time reached. Each variant predicts τ ahead per inference, so reaching t takes $n = t/\tau$ inferences: the $\tau = 0.005$ variant is rolled out by feeding its prediction back as the next query, the $\tau = 0.1$ variant reaches $t = 0.2$ in two. HFluxNO is shown for reference (its 20-step rollout is the $t = 0.1$ column). Relative ℓ^2 and ℓ^∞, mean $\pm$ std over three training runs; for ICON ($\tau = 0.1$) at $t = 0.1$ the spread is dominated by a few test trajectories on which the inference diverges. "–" marks a horizon not evaluated for that model

	$t = 0.005$ $(\times 10^{-2})$			$t = 0.1$ $(\times 10^{-1})$			$t = 0.2$ $(\times 10^{-1})$		
Model	n	ℓ^2	ℓ^∞	n	ℓ^2	ℓ^∞	n	ℓ^2	ℓ^∞
ICON ($\tau = 0.005$)	1	1.38 ± 0.25	37.3 ± 5.3	20	10.6 ± 1.0	13.3 ± 0.72	–	–	–
ICON ($\tau = 0.1$)	–	–	–	1	0.105 ± 1.9	3.54 ± 0.58	2	2.43 ± 0.94	0.806 ± 0.58
HFluxNO	1	0.410 ± 0.011	1.55 ± 0.062	20	0.521 ± 0.022	3.68 ± 0.16	40	–	–

E VERIFICATION OF EXACT CONSERVATION

As discussed in the conservation paragraph of Section 4.3, the HFluxNO update conserves the total $\sum_i \boldsymbol{u}_i^n$ exactly for every parameter vector the hypernetwork produces, as a consequence of the flux-difference form and periodic padding. Since the property is structural, we verify it on a single

Table 9: ICON ($\tau = 0.1$) on the out-of-distribution sets of the 1D cubic family.

Test set	Rel. ℓ^2	Rel. ℓ^∞
Shock IC	6.47e-2 $\pm$ 3.5e-2	6.33e-1 $\pm$ 5.1e-1
Sine flux, GRF IC	6.29e-3 $\pm$ 1.8e-3	9.35e-2 $\pm$ 5.4e-2
Sine flux, shock IC	4.37e-2 $\pm$ 3.6e-2	6.43e-1 $\pm$ 1.7e-1

Table 10: ICON ($\tau = 0.1$) on the shallow-water and viscous Burgers families, in distribution and under shock-dominated initial conditions.

	In distribution		Shock IC	
Family	Rel. ℓ^2	Rel. ℓ^∞	Rel. ℓ^2	Rel. ℓ^∞
Shallow water	3.15e-1 $\pm$ 4.7e-3	9.16e-1 $\pm$ 3.2e-3	2.18e-1 $\pm$ 3.2e-3	7.78e-1 $\pm$ 5.0e-3
Viscous Burgers	5.55e-2 $\pm$ 7.5e-2	2.95e-1 $\pm$ 2.9e-1	1.03e-2 $\pm$ 1.1e-3	1.33e-1 $\pm$ 7.3e-3

family, the two-dimensional cubic conservation law. The same measurement quantifies the drift of the baselines, which must learn to conserve.

For a trajectory on the $N_1 \times N_2$ grid, let $M^n = \Delta x_1 \Delta x_2 \sum_{\boldsymbol{i}} \boldsymbol{u}_{\boldsymbol{i}}^n$ be the discrete mass at step n, and define the normalized drift

$$D^n = \frac{|M^n - M^0|}{|M^0|}, \tag{12}$$

which compares the change in mass to the magnitude of the initial state so that trajectories with different initial conditions can be pooled. For any scheme of the form Eq. (2) with periodic boundaries, including the reference solver and HFluxNO, $D^n = 0$ in exact arithmetic; what remains is floating-point round-off in single precision.

Figure 4 shows D^n over an autoregressive rollout to $t = 0.3$ on the validation set, and Table 11 summarizes it by its mean over all steps and trajectories, its final value, and its maximum. The reference solver and HFluxNO both sit at round-off level, 10^{-8} and 10^{-9}. The reference is slightly higher, which we attribute to its 32 substeps per saved step (Section B): it accumulates more round-off than HFluxNO, which takes one step.

The baselines are five to seven orders of magnitude above this level, and their drift is not a constant offset but grows with the rollout: DPOT's final-step drift is above its mean, and DISCO's final-step drift is more than three times its mean. Since these models are trained only on the one-step error, nothing constrains the mass they lose or gain from step to step, and the accumulated drift is one component of their rollout error in Fig. 2. HFluxNO has no such component, whatever the context and whatever the generated parameters.

Table 11: Normalized mass drift on the 2D cubic validation set: mean over all rollout steps and trajectories, value at the final step, and maximum over the rollout. Both conservative schemes are at round-off level.

Model	Mean	Final	Worst
Reference solver	$\mathbf{1.76 \times 10^{-8}}$	$\mathbf{2.54 \times 10^{-8}}$	$\mathbf{3.70 \times 10^{-8}}$
DPOT	1.85×10^{-3}	2.28×10^{-3}	4.46×10^{-3}
DISCO	1.33×10^{-2}	4.49×10^{-2}	4.83×10^{-2}
HFluxNO	$\mathbf{3.62 \times 10^{-9}}$	$\mathbf{4.55 \times 10^{-9}}$	$\mathbf{8.39 \times 10^{-9}}$

F FULL RESULTS

Tables 12 to 14 tabulate every number behind Fig. 2, together with the single-step errors that the figure omits, under the protocol and metrics of Section 5. The table layouts match the figure: each family pairs its in-distribution test set with its shock-dominated out-of-distribution set, and the unseen sine flux is listed separately. The single-step errors rank the models as the rollout errors do,

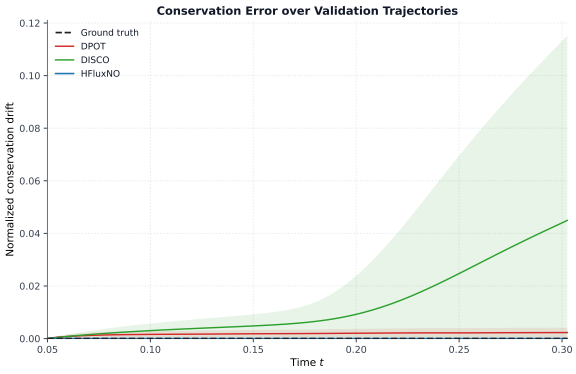


Figure 4: Normalized mass drift D^n, Eq. (12), over an autoregressive rollout on the 2D cubic validation set (mean over trajectories). The reference solver and HFluxNO stay at floating-point round-off; DPOT and DISCO accumulate drift because nothing in their architecture ties the predicted state to a flux balance.

with similar margins. We choose to report the rollout in the main text as this is the quantity a solver is used for. HFluxNO's lead has two exceptions. On shallow water under shock-dominated initial conditions, DISCO and DPOT are more accurate at a single step in both norms. DISCO is also more accurate after 20 steps in ℓ^2. On viscous Burgers under shock-dominated initial conditions, DPOT's single-step ℓ^∞ error is marginally lower.

Table 12: Single-step prediction error on every family, in distribution and under shock-dominated initial conditions (the two conditions of Fig. 2). Mean $\pm$ std over three training runs; best per column in bold.

		In distribution ($\times 10^{-2}$)		Shock IC ($\times 10^{-2}$)	
Family	Model	Rel. ℓ^2	Rel. ℓ^∞	Rel. ℓ^2	Rel. ℓ^∞
1D cubic	DPOT	0.740 ± 0.0091	2.42 ± 0.038	0.724 ± 0.020	2.45 ± 0.017
	DISCO	1.06 ± 0.20	3.90 ± 0.97	0.970 ± 0.11	3.73 ± 0.42
	HFluxNO	$\mathbf{0.410 \pm 0.012}$	$\mathbf{1.55 \pm 0.062}$	$\mathbf{0.562 \pm 0.027}$	$\mathbf{2.27 \pm 0.14}$
Shallow water	DPOT	0.849 ± 0.040	2.88 ± 0.16	7.45 ± 0.54	34.6 ± 2.8
	DISCO	1.42 ± 0.063	5.07 ± 0.19	$\mathbf{5.85 \pm 0.12}$	$\mathbf{24.6 \pm 0.43}$
	HFluxNO	$\mathbf{0.627 \pm 0.0027}$	$\mathbf{2.29 \pm 0.038}$	12.5 ± 0.28	49.6 ± 0.62
Viscous Burgers	DPOT	0.263 ± 0.00029	0.717 ± 0.0043	0.189 ± 0.0021	$\mathbf{0.508 \pm 0.017}$
	DISCO	0.343 ± 0.021	0.965 ± 0.060	0.273 ± 0.0097	0.758 ± 0.042
	HFluxNO	$\mathbf{0.132 \pm 0.0029}$	$\mathbf{0.344 \pm 0.0073}$	$\mathbf{0.164 \pm 0.0024}$	0.524 ± 0.0072
2D cubic	DPOT	0.791 ± 0.014	3.74 ± 0.096	16.1 ± 4.1	23.4 ± 3.1
	DISCO	0.318 ± 0.026	1.38 ± 0.085	1.29 ± 0.062	5.63 ± 0.43
	HFluxNO	$\mathbf{0.0683 \pm 0.00094}$	$\mathbf{0.370 \pm 0.0047}$	$\mathbf{0.757 \pm 0.11}$	$\mathbf{3.29 \pm 0.47}$
2D Euler	DPOT	1.56 ± 0.017	10.5 ± 0.15	1.68 ± 0.011	8.69 ± 0.063
	DISCO	0.992 ± 0.015	5.18 ± 0.13	0.908 ± 0.023	4.85 ± 0.12
	HFluxNO	$\mathbf{0.286 \pm 0.0046}$	$\mathbf{1.85 \pm 0.031}$	$\mathbf{0.275 \pm 0.0038}$	$\mathbf{1.75 \pm 0.028}$

G QUALITATIVE ROLLOUTS

The figures in this appendix show individual trajectories behind the aggregate numbers of Fig. 2 and Section F. They are chosen to display the two behaviors that Section 6.2 attributes to the models: HFluxNO's error stays localized at moving fronts, while the baselines' error spreads away from the fronts as the rollout proceeds.

One-dimensional cubic family. Figure 5 shows a rollout on an in-distribution test trajectory and on the hardest out-of-distribution set, the unseen sine flux with shock-dominated initial conditions,

Table 13: 20-step autoregressive rollout error, same layout as Table 12; these are the values plotted in Fig. 2.

		In distribution ($\times 10^{-1}$)		Shock IC ($\times 10^{-1}$)	
Family	Model	Rel. ℓ^2	Rel. ℓ^∞	Rel. ℓ^2	Rel. ℓ^∞
1D cubic	DPOT	1.36 ± 0.031	6.78 ± 0.097	1.58 ± 0.040	8.23 ± 0.38
	DISCO	0.822 ± 0.056	4.55 ± 0.31	0.858 ± 0.057	4.73 ± 0.26
	HFluxNO	$\mathbf{0.521 \pm 0.021}$	$\mathbf{3.68 \pm 0.16}$	$\mathbf{0.668 \pm 0.017}$	$\mathbf{4.35 \pm 0.090}$
Shallow water	DPOT	1.05 ± 0.037	4.63 ± 0.10	4.59 ± 0.061	9.77 ± 0.055
	DISCO	1.11 ± 0.058	4.45 ± 0.20	$\mathbf{3.35 \pm 0.10}$	9.40 ± 0.68
	HFluxNO	$\mathbf{0.755 \pm 0.013}$	$\mathbf{3.98 \pm 0.042}$	4.28 ± 0.066	$\mathbf{8.47 \pm 0.13}$
Viscous Burgers	DPOT	0.466 ± 0.011	1.56 ± 0.044	0.450 ± 0.032	1.78 ± 0.12
	DISCO	0.656 ± 0.34	3.43 ± 2.3	0.453 ± 0.14	2.41 ± 1.2
	HFluxNO	$\mathbf{0.172 \pm 0.0023}$	$\mathbf{0.646 \pm 0.0065}$	$\mathbf{0.221 \pm 0.0050}$	$\mathbf{0.999 \pm 0.0095}$
2D cubic	DPOT	2.19 ± 0.15	7.84 ± 0.43	8.12 ± 0.26	16.9 ± 1.2
	DISCO	0.368 ± 0.043	1.40 ± 0.11	2.04 ± 0.11	6.79 ± 0.35
	HFluxNO	$\mathbf{0.131 \pm 0.0065}$	$\mathbf{0.638 \pm 0.034}$	$\mathbf{1.76 \pm 0.24}$	$\mathbf{5.65 \pm 0.75}$
2D Euler	DPOT	2.23 ± 0.080	7.13 ± 0.21	1.77 ± 0.070	7.81 ± 0.44
	DISCO	1.27 ± 0.11	5.45 ± 0.38	1.05 ± 0.13	4.97 ± 0.61
	HFluxNO	$\mathbf{0.394 \pm 0.0084}$	$\mathbf{3.27 \pm 0.063}$	$\mathbf{0.268 \pm 0.0037}$	$\mathbf{2.37 \pm 0.035}$

Table 14: Cubic-trained models on the unseen sine flux (the right-hand group of Fig. 2), with smooth (GRF) and shock-dominated initial conditions. Mean $\pm$ std over three training runs; best per column in bold.

		Single step ($\times 10^{-2}$)		20-step rollout ($\times 10^{-1}$)	
Test set	Model	Rel. ℓ^2	Rel. ℓ^∞	Rel. ℓ^2	Rel. ℓ^∞
Sine flux, GRF IC	DPOT	0.598 ± 0.0096	1.16 ± 0.027	1.64 ± 0.074	4.92 ± 0.13
	DISCO	0.590 ± 0.10	1.25 ± 0.21	0.667 ± 0.070	2.33 ± 0.25
	HFluxNO	$\mathbf{0.266 \pm 0.0032}$	$\mathbf{0.644 \pm 0.0037}$	$\mathbf{0.414 \pm 0.012}$	$\mathbf{1.78 \pm 0.042}$
Sine flux, shock IC	DPOT	0.783 ± 0.020	2.74 ± 0.092	1.76 ± 0.047	9.29 ± 0.44
	DISCO	0.951 ± 0.20	3.88 ± 0.78	0.901 ± 0.081	5.39 ± 0.22
	HFluxNO	$\mathbf{0.485 \pm 0.029}$	$\mathbf{1.98 \pm 0.15}$	$\mathbf{0.606 \pm 0.020}$	$\mathbf{3.83 \pm 0.076}$

up to $t = 0.315$, 63 steps or about three context lengths. In both cases the HFluxNO error is confined to the immediate neighborhood of the fronts and does not grow in extent over the rollout. On the other hand, the baseline errors are of comparable magnitude at early times but develop oscillations away from the fronts as the horizon increases, in the manner described in Section 6.2. The out-of-distribution row shows that the sharper fronts and the unfamiliar flux change the level of the HFluxNO error but not its structure.

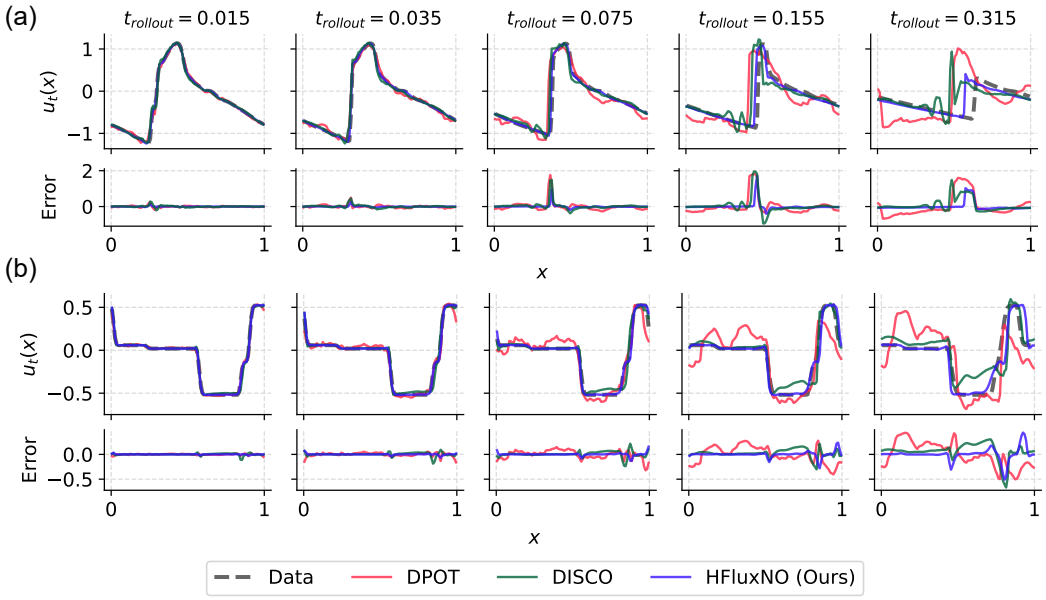


Figure 5: One-dimensional rollouts on the cubic family at $t = 0.015, 0.035, 0.075, 0.155, 0.315$. (a) In-distribution test trajectory. (b) Out-of-distribution trajectory with the unseen sine flux and shock-dominated initial conditions. Each row shows the solution (top) and the pointwise error of each model (bottom); one context length is $t = 0.1$.

Two-dimensional cubic family. Figures 6 and 7 show rollouts in distribution and under shock-dominated initial conditions. In each figure the first three columns show the reference, the HFluxNO prediction, and its pointwise absolute error, and the rightmost column compares cross-sections of all models along $y = 0.5$ with their errors. In distribution (Fig. 6) HFluxNO reproduces the large-scale profile and the positions of the steep fronts, and its error field is a thin band along those fronts; the baselines' cross-sections deviate near the sharp transitions and increasingly so at later times. Under the shift to piecewise-constant initial conditions (Fig. 7), evaluated without fine-tuning, HFluxNO keeps the piecewise structure and the plateau values, and its error again grows around the shock locations rather than across the field, whereas the baselines develop amplitude and phase errors over the rollout. Note that the reference for this family is first order (Section B), so the fronts in the reference are themselves smeared over a few cells.

Two-dimensional Euler system. Figures 8 and 9 show a rollout to $t = 0.305$, three context lengths, on an in-distribution Euler test trajectory, and Figs. 10 and 11 the same on a trajectory from the shock-dominated out-of-distribution set. Figs. 12 and 13 show the internal energy density and the speed for both. All fields are derived from the four predicted conserved variables. Two things are visible at once. The baselines lose the solution within the first 16 steps, less than one context length, and in ways characteristic of their architectures. DPOT's prediction breaks into blocks aligned with its 8×8 patch grid, the imprint of the patch embedding that aggregates its context. DISCO's is covered by oscillations trailing every front, the ringing of a non-dissipative update at a discontinuity. By $t = 0.155$ neither retains the large-scale structure.

HFluxNO, by contrast, follows the reference through $t = 0.23$, more than two context lengths, on both trajectories, apart from the first grid-scale texture that appears at about two context lengths (Fig. 3(b)). It reproduces the curved shock and the compression region of the in-distribution case,

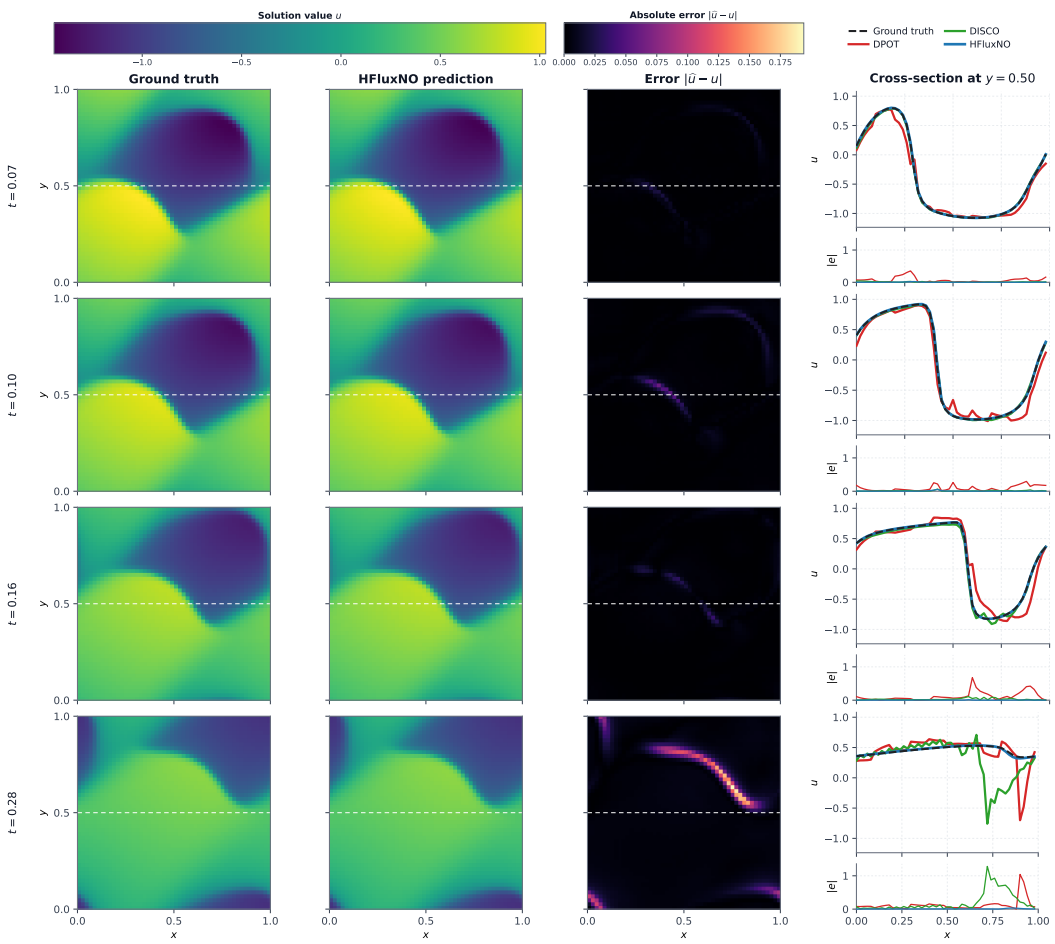


Figure 6: Two-dimensional cubic rollout, in distribution, at $t = 0.07, 0.10, 0.16, 0.28$ (rows). Columns: reference, HFluxNO prediction, pointwise absolute error $|\hat{u} - u|$, and cross-sections at $y = 0.5$ (white dashed line) for the reference and all models, with the pointwise absolute error of each in the panel below.

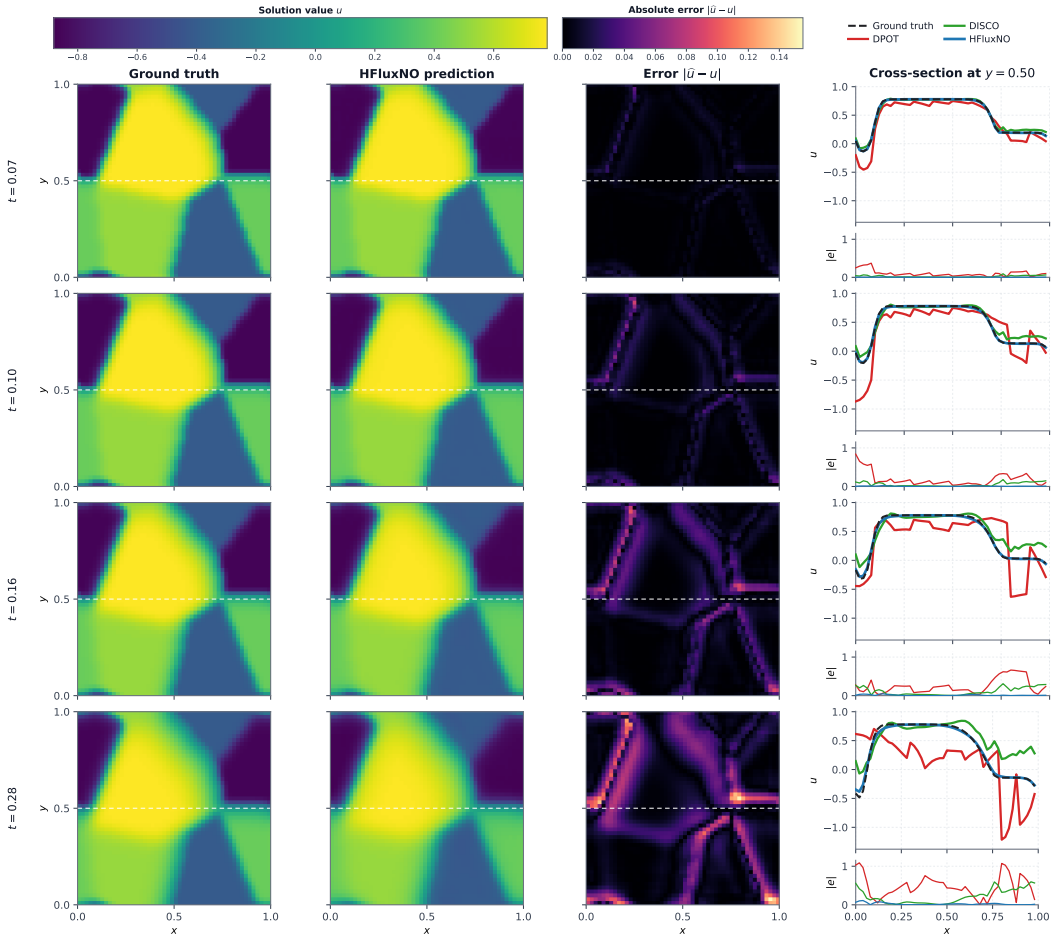


Figure 7: As Fig. 6, for shock-dominated initial conditions with the cubic flux and coefficient distribution unchanged; all models evaluated without fine-tuning.

and the straight, closely spaced fronts and plateau values of the shock-dominated case. Its density error stays confined to thin lines along the fronts.

The last frame shows where HFluxNO's own stability ends. On the shock-dominated trajectory a grid-scale checkerboard, alternating in both grid directions, appears behind the strongest front by $t = 0.23$. By $t = 0.305$ it fills the wedge between the two main fronts with an amplitude larger than the color range, while the fronts and the surrounding smooth flow remain in place. On the in-distribution trajectory the oscillation appears later and stays confined to one region behind the strongest compression, which wraps across the periodic boundary and therefore also shows at the right-hand corners of Figs. 8 and 9. Its pattern is different: stripes rather than a checkerboard, i.e. the solution alternates from cell to cell along one grid axis and is smooth along the other. Its amplitude is as large as on the shock-dominated trajectory, saturating the color scale at both ends, but the rest of the field is still correct.

Both patterns belong to a family of failures that is well documented for classical fluxes. Low-dissipation Riemann solvers develop grid-scale oscillations at strong shocks, known as odd-even decoupling and the carbuncle (Quirk, 1994; Dumbser et al., 2004). These are linear instabilities of the discrete shock structure. They are triggered by round-off, grow exponentially, and are cured only by adding dissipation to the wave fields that the flux leaves undamped. They do not depend on the boundary conditions: the instability is local to the shock, and Dumbser et al. (2004) find the same mode with periodic boundaries along the shock.

How closely our oscillations match the classical mode differs between the two trajectories. The classical mode is a sawtooth along a grid-aligned front. The stripes on the in-distribution trajectory have this one-directional form. The checkerboard on the shock-dominated trajectory does not: it alternates in both directions and fills the region behind oblique fronts. We therefore make the connection at the level of the class, a flux with too little dissipation at states it was not fitted on, and not at the level of mechanism, which we have not analyzed for a learned flux.

Neither is a failure of conservation. Mass is conserved exactly through the oscillation, which illustrates that the flux-difference form guarantees conservation and nothing about stability. Both mark the boundary of the present method on this family, and we return to them in Section 7. The classical cure, dissipation on the undamped wave fields, translates into a constraint on the learned flux that we leave to future work. The internal energy density and speed (Figs. 12 and 13) show the same behavior in the remaining variables, with each model's artifact at the same locations. This is expected, since the errors originate in the update rather than in any one channel.

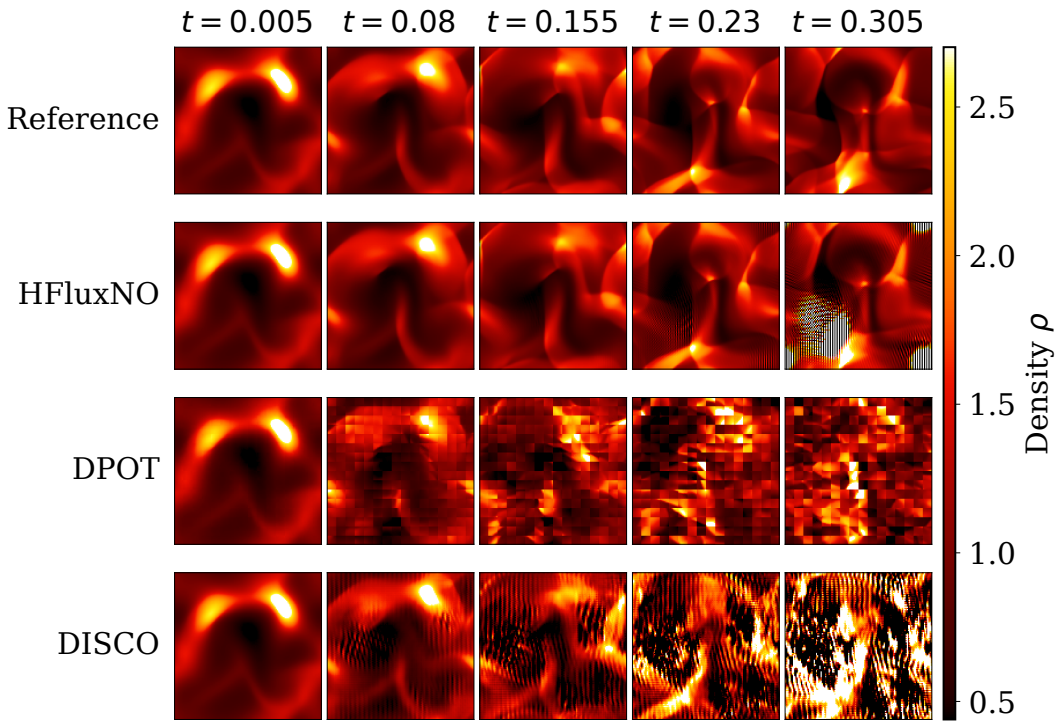


Figure 8: Two-dimensional Euler, in distribution: density ρ over a rollout to $t = 0.305$, three context lengths, for the reference and the three models. One color scale, taken from the reference over the shown times.

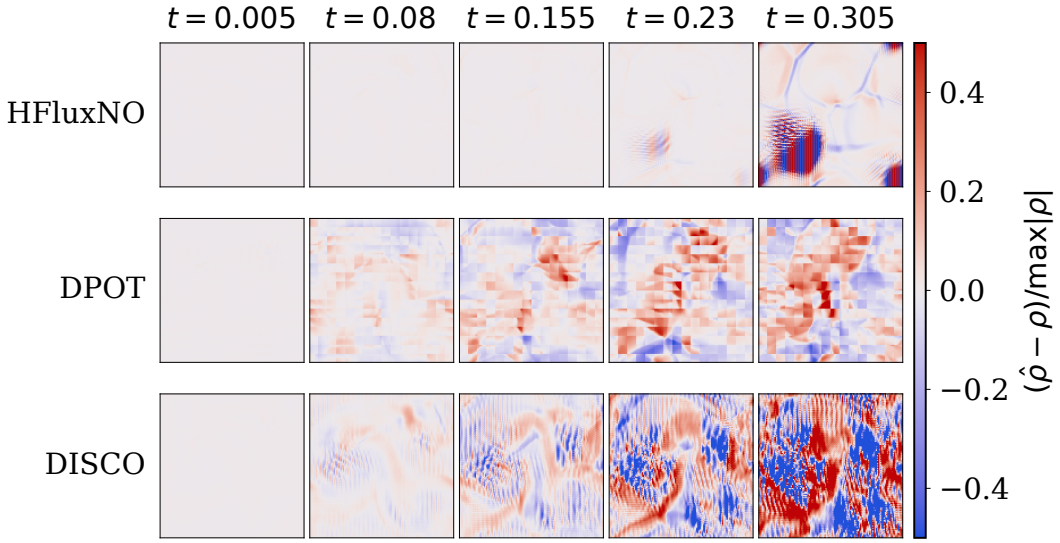


Figure 9: Signed density error $(\hat{\rho} - \rho)/\max|\rho|$ for the rollout of Fig. 8. The color scale is fixed at ± 0.5 for every panel of this and Fig. 11; values beyond it saturate. Through $t = 0.23$ HFluxNO's error is a thin doublet along the front, DPOT's is aligned with its patch grid, and DISCO's is an oscillatory train trailing each front; the saturated region in HFluxNO's last frame is the oscillation described in the text.

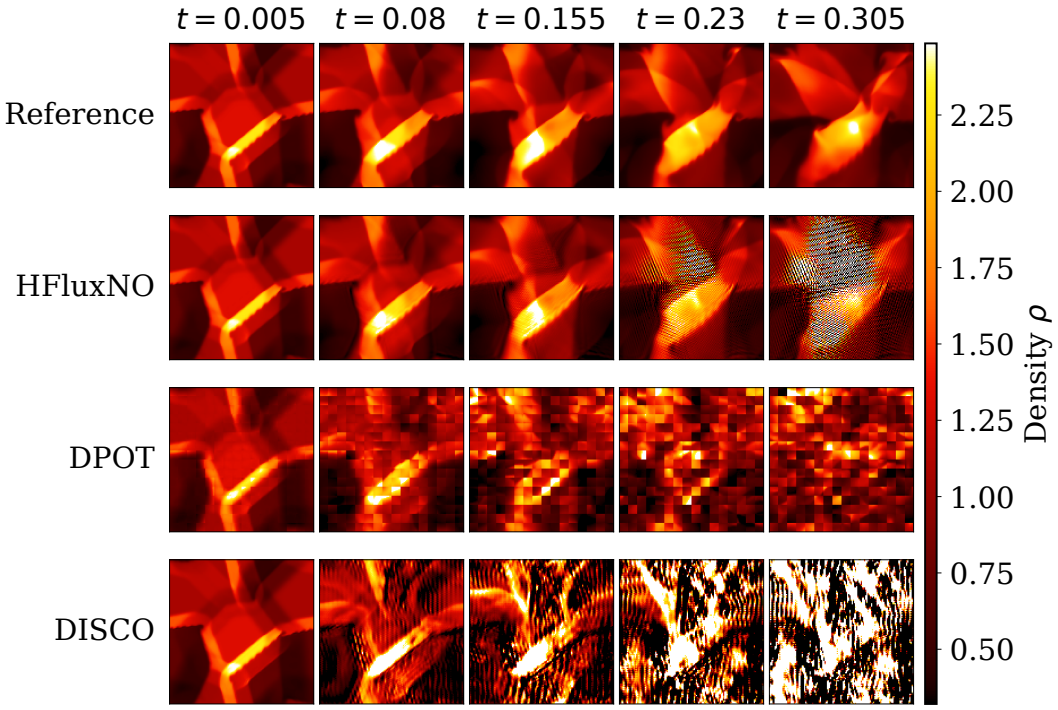


Figure 10: Two-dimensional Euler under shock-dominated initial conditions (out of distribution, no fine-tuning): density over a rollout to $t = 0.305$, layout as in Fig. 8.

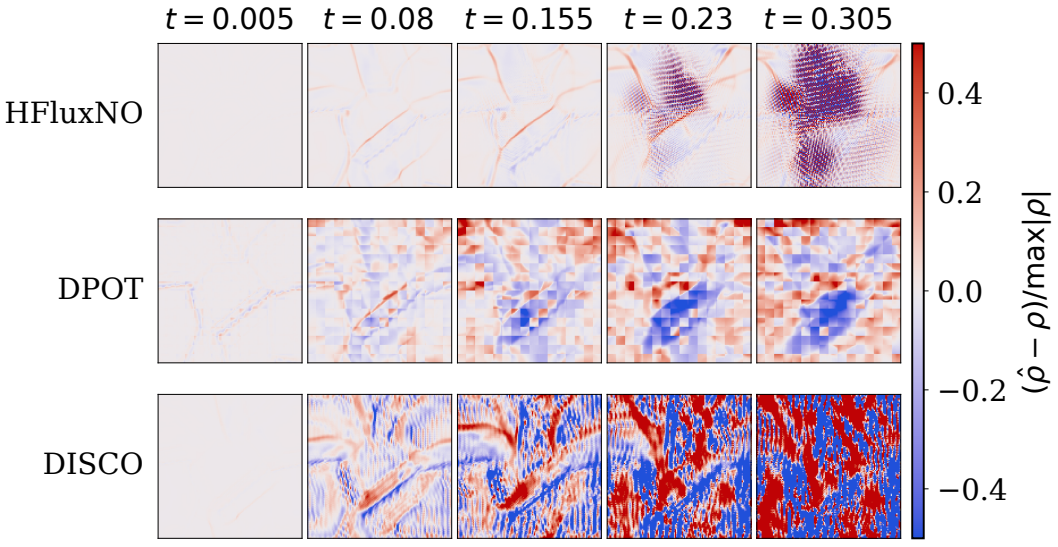


Figure 11: Signed density error for the rollout of Fig. 10, same fixed ± 0.5 scale as Fig. 9.

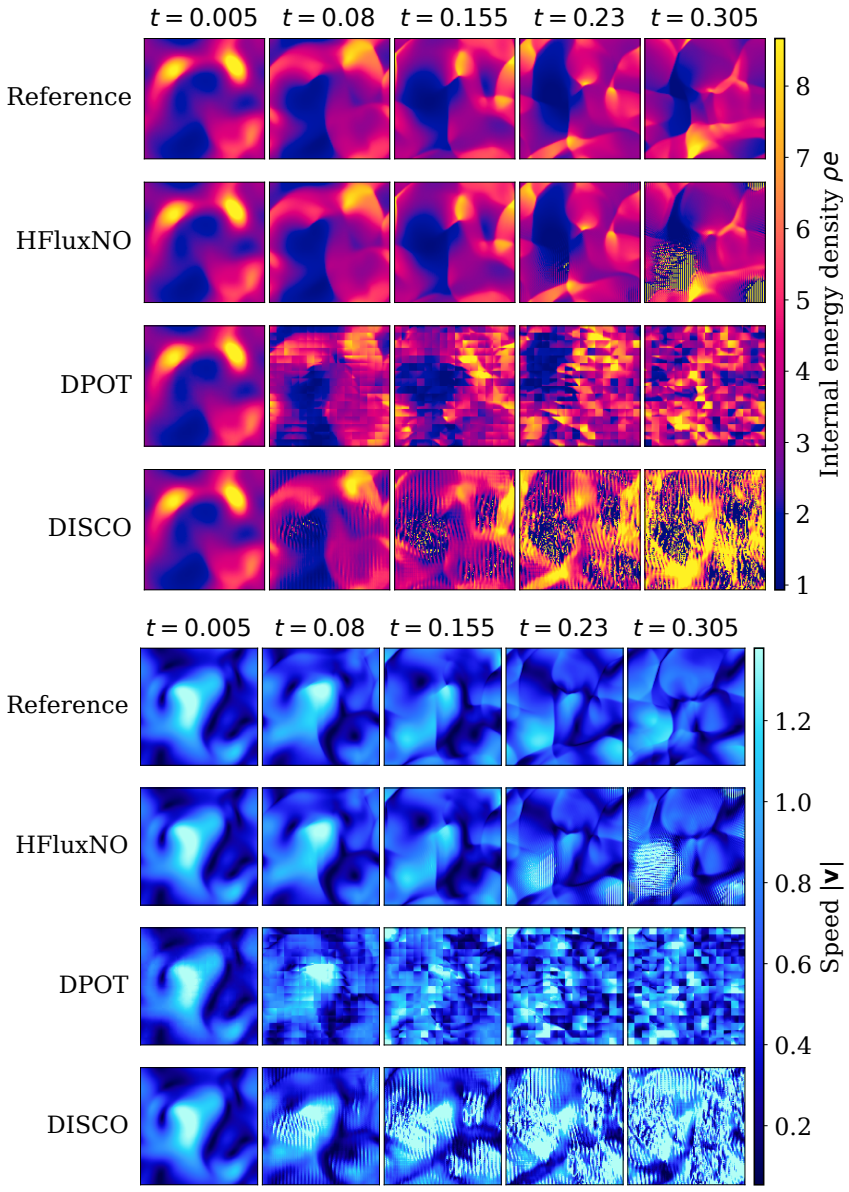


Figure 12: Internal energy density $\rho e = E - \frac{1}{2}\rho|\mathbf{v}|^2$, which equals $p/(\gamma - 1)$ (top), and speed $|\mathbf{v}|$ (bottom) for the in-distribution rollout of Fig. 8, derived from the predicted conserved variables; layout and times as there.

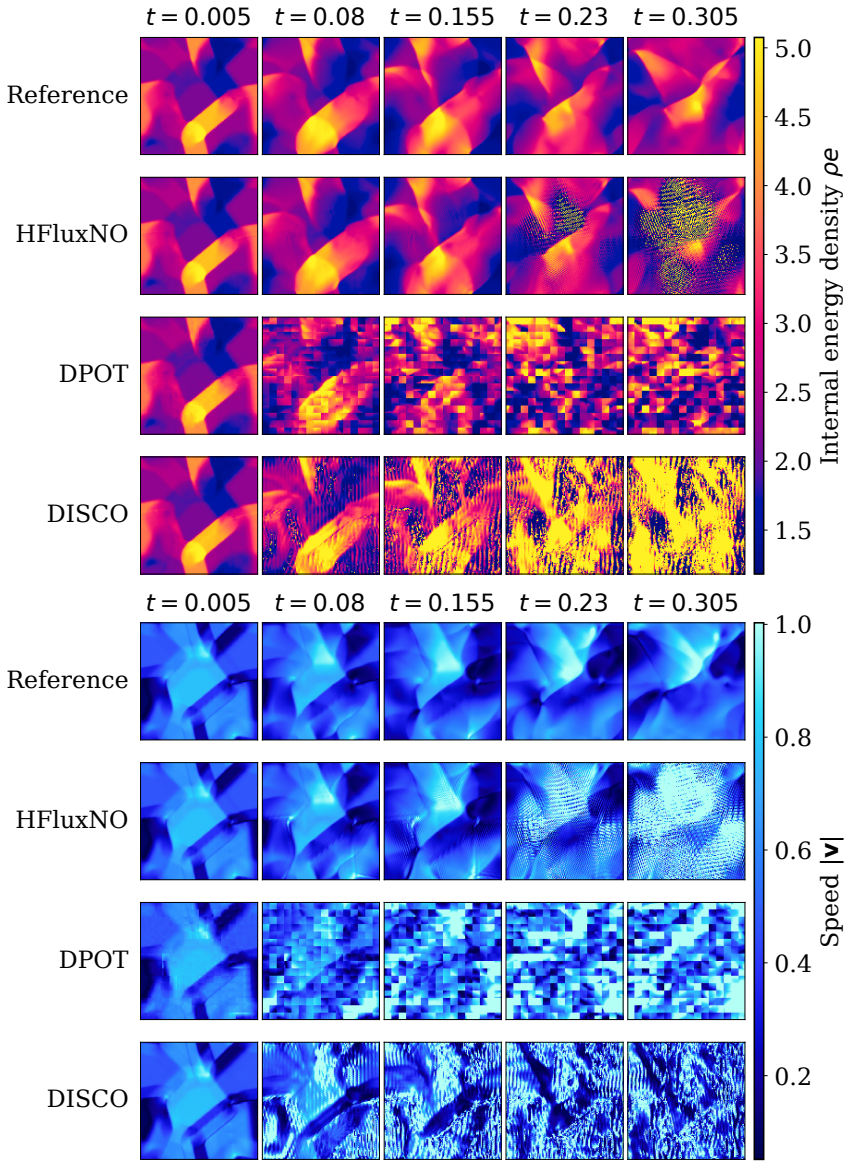


Figure 13: As Fig. 12, for the shock-dominated rollout of Fig. 10.