

LGNO: A Local–Global Neural Operator for Hyperbolic Conservation Laws

Hao Wang^a, Chi-Wang Shu^{b,1}, Qi Tang^{a,2,*}

^a*School of Computational Science and Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA.*

^b*Division of Applied Mathematics, Brown University, Providence, Rhode Island 02912, USA.*

Abstract

Solutions of hyperbolic conservation laws exhibit both smooth structures across large scales and sharp localized features such as shocks and contact discontinuities, making them difficult to approximate accurately with existing neural operators. The Fourier Neural Operator (FNO) captures long-range interactions well but tends to smear localized structures through excessive numerical dissipation. To address this, we propose a Local–Global Neural Operator (LGNO) that learns a one-step discrete flow map by combining a global FNO branch for representing smooth dynamics at large scales with a local multiresolution branch for enhancing localized discontinuities and nonsmooth features. The model is trained with a one-step loss that combines a physical space prediction term and a spectral penalty on high frequencies to suppress spurious oscillations near steep fronts. On a large collection of benchmarks in one and two dimensions, LGNO consistently outperforms FNO baselines with matched parameter counts, reducing one-step errors by factors of 2–5 and remaining significantly more accurate over long autoregressive rollouts. Most strikingly, although it is trained only on short-time data from a high-order WENO-Z scheme, the long-time rollout of LGNO on a coarse 256^2 grid exhibits lower numerical dissipation than the same WENO-Z scheme run on a finer 512^2 grid, while being orders of magnitude cheaper to evaluate. These results suggest that, with an appropriate architecture and training objective, learned operators can effectively learn discrete flow maps. They further suggest that such learned operators have the potential to control long-time numerical dissipation better than the conventional shock-capturing schemes that generate the training data.

Keywords: Hyperbolic conservation laws, Shock capturing, Data-driven operator learning

1. Introduction

Conservation laws arise in a broad range of applications, including gas dynamics [1, 2] and plasma physics such as magnetohydrodynamics [3, 4]. Many of these applications are governed by hyperbolic systems of partial differential equations (PDEs) [5, 6]. A defining feature of such systems is that even smooth initial data can develop sharp gradients and discontinuities in finite time [7]. This makes their numerical approximation fundamentally challenging: a successful method must accurately propagate nonlinear waves, sharply resolve localized structures such as shocks and contact discontinuities, and remain robust over long-time integration.

A large body of conventional numerical methods has been developed for conservation laws, including finite volume methods [6], discontinuous Galerkin methods [8, 9], and high order nonoscillatory reconstruction techniques such as the WENO scheme [10] and its variants such as WENO-Z [11] and TENO [12]. While these methods differ in formulation, their effectiveness is closely tied to the structure of the underlying equations [13, 14]. In particular, shock-dominated dynamics is governed by strongly localized interactions, and classical high-resolution schemes are carefully designed around local reconstruction, numerical fluxes, and nonlinear stabilization mechanisms. These ingredients allow them to balance accuracy in smooth regions with robustness near discontinuities [15]. However the robustness comes at the price of numerical

*Corresponding authors

Email addresses: wanghaomathe@gmail.com (Hao Wang), chi-wang_shu@brown.edu (Chi-Wang Shu), qtang@gatech.edu (Qi Tang)

¹Research was partially supported by NSF grant DMS-2309249.

²Research was partially supported by the U.S. Department of Energy Office of Science, Early Career Research Program under Award Number DE-SC0026277.

dissipation: reducing this dissipation while preserving fine-scale structures typically requires higher-order reconstructions, less dissipative stabilization mechanisms, or finer meshes, each of which substantially increases the computational cost.

In recent years, machine learning methods for PDEs have made rapid progress, especially in neural surrogate models for solution prediction [16] and in operator learning approaches such as the Fourier Neural Operator (FNO) [17] and the Deep Operator Network (DeepONet) [18]. By learning the solution operator or discrete flow map directly, these approaches offer a promising route for accelerated simulation in repeated-query settings [19, 20]. Studies in approximation theory indicate that the complexity of operator learning depends strongly on the regularity of the target map, with smoother operators generally being easier to approximate efficiently [21]. For nonlinear conservation laws, however, shocks and other discontinuities naturally introduce nonsmooth solution structures, which makes accurate operator learning substantially more difficult [22].

While neural operators with nonlinear reconstruction, such as FNO, can in principle represent discontinuous solutions [23], representational capacity alone does not guarantee practical performance. Existing studies have shown that FNO can handle relatively simple discontinuous examples, but their accuracy deteriorates as the discontinuity structures become more complex [22]. Related neural-operator approaches for discontinuous Riemann problems in the compressible Euler equations have also shown promising results, but their accuracy near shocks and contact discontinuities remains limited [24]. This gap has motivated a growing line of structure-aware learning methods that incorporate ideas from classical shock-capturing schemes, for example by learning local numerical fluxes or neural finite-volume structures [25, 26], indicating that locality and conservation structure are useful inductive biases for learning nonsmooth dynamics.

Motivated by this perspective, this work introduces a Local-Global Neural Operator (LGNO) for hyperbolic conservation laws that learns a one-step discrete flow map by coupling two complementary components: a global branch, built on an FNO, that captures coherent large-scale behavior and long-range interactions, and a local multiresolution convolutional branch that sharpens localized nonsmooth structures. The model is trained with a one-step loss combining a physical-space prediction term and a high-frequency spectral penalty that suppresses spurious oscillations. This design aligns the architecture with the coexistence of smooth global waves and localized nonsmooth features that characterizes conservation law solutions.

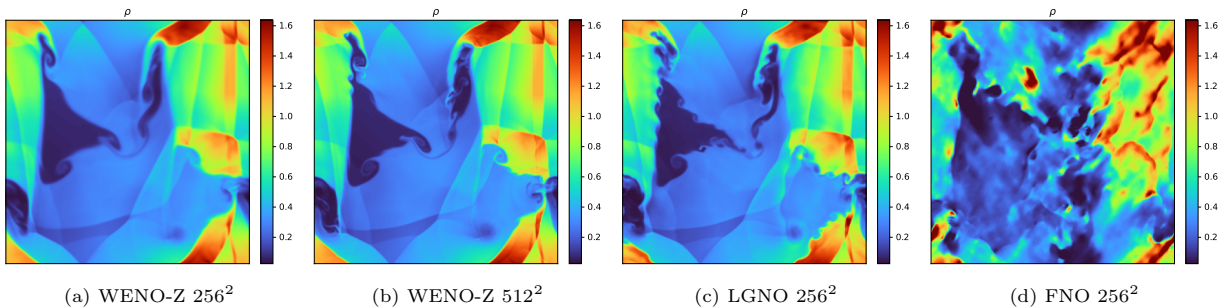


Figure 1: **LGNO is less dissipative than WENO-Z at a higher resolution.** Density field for the two-dimensional Euler equations after 50 autoregressive rollout steps with $\Delta t = 0.01$. Trained only on short-time WENO-Z data at 512^2 resolution, the learned LGNO on the coarse 256^2 grid (c) preserves more vortical structures along the contact discontinuities and shear layers than the WENO-Z reference at both 256^2 (a) and 512^2 (b), reflecting lower long-time numerical dissipation, whereas the FNO baseline at 256^2 (d) is severely degraded over the long rollout and loses the coherent structure.

The main contributions of this work are threefold. First, we develop a local multiresolution branch that complements the global FNO branch, together with a simple learned boundary padding for outflow boundary conditions. Second, through an extensive study on standard one- and two-dimensional hyperbolic benchmarks, we show that LGNO captures shock and vortical structures significantly more accurately than a parameter-matched FNO. Third, and most notably, we demonstrate that, although it is trained only on short-time data from the WENO-Z scheme, LGNO on a coarse 256^2 grid produces long-time rollouts that are less dissipative than the same scheme run on a finer 512^2 grid, while being orders of magnitude cheaper to evaluate, as illustrated in Figure 1.

This may appear counterintuitive, since the training data are produced by the shock-capturing scheme itself; the reason is that such schemes are accurate over short times but accumulate excessive numerical

dissipation when their dissipative reconstruction acts at every step over a long integration, whereas the learned one-step flow map of LGNO inherits only the lower dissipation of the short-time training data as it is composed over long rollouts. This indicates that, with an appropriate architecture and training procedure, learned operators have the potential to control long-time numerical dissipation better than the shock-capturing schemes that generate their training data.

The remainder of this paper is organized as follows. Section 2 reviews hyperbolic conservation laws and operator learning. Section 3 presents the proposed LGNO and its data-driven training strategy. Section 4 evaluates LGNO on several benchmark problems against parameter-matched FNO baseline and the WENO-Z scheme, and Section 5 concludes the paper. Implementation details, computational cost comparisons, additional numerical results, and benchmark setups are provided in the appendices.

2. Background and Related Work

2.1. Hyperbolic Conservation Laws and Conventional Numerical Schemes

We consider systems of hyperbolic conservation laws posed on a spatial domain $\Omega \subset \mathbb{R}^d$:

$$\frac{\partial \mathbf{u}}{\partial t} + \sum_{\alpha=1}^d \frac{\partial \mathbf{f}_\alpha(\mathbf{u})}{\partial x_\alpha} = 0, \quad \mathbf{x} \in \Omega, \quad t > 0. \quad (1)$$

Here $\mathbf{u} = (u_1, \dots, u_m)^\top \in \mathbb{R}^m$ denotes the vector of conserved variables, and $\mathbf{f}_\alpha : \mathbb{R}^m \rightarrow \mathbb{R}^m$, $\alpha = 1, \dots, d$, are the Cartesian components of the flux. The system is supplemented with an initial condition

$$\mathbf{u}(\mathbf{x}, 0) = \mathbf{u}_0(\mathbf{x}), \quad \mathbf{x} \in \Omega,$$

together with suitable boundary conditions on $\partial\Omega$.

A defining feature of nonlinear hyperbolic conservation laws is that even smooth initial data may develop steep gradients and discontinuities in finite time [7], so that classical solutions generally fail to exist globally and one must work with weak solutions. In integral form, Eq. (1) implies the conservation relation

$$\frac{d}{dt} \int_K \mathbf{u}(\mathbf{x}, t) \, d\mathbf{x} + \int_{\partial K} \mathbf{F}(\mathbf{u}) \cdot \mathbf{n} \, ds = 0, \quad (2)$$

for every control volume $K \subset \Omega$, where $\mathbf{F}(\mathbf{u}) = (\mathbf{f}_1(\mathbf{u}), \dots, \mathbf{f}_d(\mathbf{u}))$ and \mathbf{n} is the outward unit normal. Since weak solutions are in general nonunique, additional admissibility conditions, such as entropy inequalities, are required to select the physically relevant solution [5].

From a numerical standpoint, the solution structure of conservation laws is heterogeneous. In smooth regions, global high-order approximations are extremely effective. For periodic problems with sufficiently smooth solutions, for instance, Fourier spectral methods approximate the solution by a truncated global expansion and satisfy the standard estimate

$$\|\mathbf{u} - \mathbf{u}_N\|_{L^2(\Omega)} \lesssim N^{-s} \|\mathbf{u}\|_{H^s(\Omega)},$$

when $\mathbf{u}(\cdot, t) \in H^s(\Omega)$, with exponential convergence for analytic solutions [27]. This advantage, however, relies on regularity. Once shocks or contact discontinuities emerge, the loss of regularity is localized near them, whereas the Fourier basis remains globally supported. Spectral convergence is then lost, and truncated expansions develop Gibbs oscillations whose amplitude stays $O(1)$ near the discontinuity even as the resolution increases [28]. Global spectral methods thus capture smooth coherent structures and long-range interactions efficiently, but resolve sharply localized nonsmooth features poorly.

Classical shock-capturing methods are designed precisely for this regime. Finite volume methods build directly on Eq. (2). In one space dimension, let $I_j = [x_{j-\frac{1}{2}}, x_{j+\frac{1}{2}}]$ be a cell of width Δx , and let

$$\bar{\mathbf{u}}_j(t) = \frac{1}{\Delta x} \int_{I_j} \mathbf{u}(x, t) \, dx$$

denote the cell average. A semi-discrete conservative update takes the form

$$\frac{d}{dt}\bar{\mathbf{u}}_j(t) = -\frac{1}{\Delta x}\left(\widehat{\mathbf{f}}_{j+\frac{1}{2}} - \widehat{\mathbf{f}}_{j-\frac{1}{2}}\right), \quad (3)$$

where $\widehat{\mathbf{f}}_{j+\frac{1}{2}}$ is a numerical flux at the cell interface. Godunov-type methods, approximate Riemann solvers, and high-resolution finite-volume schemes differ mainly in how these interfacial fluxes are reconstructed, evaluated, and stabilized [6, 13].

Among high-order shock-capturing methods, ENO and WENO schemes exemplify this local, nonlinear design [15]. They build interface values from local stencils and adapt the reconstruction nonlinearly to the local smoothness, yielding high-order accuracy where the solution is smooth while biasing away from nonsmooth stencils near discontinuities to suppress spurious oscillations and robustly capture localized wave structures. Unlike global spectral approximations, their effectiveness comes from localized reconstruction and nonlinear adaptation to the loss of regularity rather than from global smoothness.

This contrast suggests that learned operators for conservation laws should not rely on global spectral information alone, but should combine global mechanisms for smooth coherent structures with local mechanisms for sharply localized nonsmooth features [13, 14, 15].

2.2. Neural Operators and Operator Learning

Neural operators approximate mappings between function spaces. Let \mathcal{X} and \mathcal{Y} be Banach spaces of functions on the spatial domain $\Omega \subset \mathbb{R}^d$; one seeks to approximate an operator

$$\mathcal{G} : \mathcal{X} \rightarrow \mathcal{Y}, \quad \mathbf{a}(\mathbf{x}) \mapsto \mathbf{u}(\mathbf{x}), \quad (4)$$

where $\mathbf{a}(\mathbf{x})$ is an input function and $\mathbf{u}(\mathbf{x})$ the corresponding output. For conservation laws, it is often natural to take function spaces adapted to weak solutions, such as subsets of $L^1(\Omega; \mathbb{R}^m)$ or $BV(\Omega; \mathbb{R}^m)$.

In practice, neural operators act not on the raw input and output spaces but through higher-dimensional latent feature spaces, giving the common architecture

$$\mathcal{G}_\theta = Q \circ \mathcal{L}_{L-1} \circ \cdots \circ \mathcal{L}_0 \circ P.$$

Here $P : \mathcal{X} \rightarrow \mathcal{Z}_0$ is a lifting map that embeds the input function into a latent feature space, $\mathcal{L}_\ell : \mathcal{Z}_\ell \rightarrow \mathcal{Z}_{\ell+1}$ are operator layers acting on latent features, and $Q : \mathcal{Z}_L \rightarrow \mathcal{Y}$ is a projection map that returns the latent representation to the output space. The kernel-inspired operator structure resides mainly in the intermediate layers \mathcal{L}_ℓ , while P and Q supply the feature lifting and projection needed for an expressive neural parameterization.

For many PDEs, the solution map can be viewed as an operator induced by an integral kernel. In the linear setting, one may write

$$\mathbf{u}(\mathbf{x}) = \int_{\Omega} K(\mathbf{x}, \mathbf{y}) \mathbf{a}(\mathbf{y}) \, d\mathbf{y},$$

where K plays the role of a Green's kernel. This motivates neural operator layers that model feature transformations through learned kernel interactions and a nonlinear activation, giving the generic form

$$(\mathcal{L}_\ell \mathbf{v})(\mathbf{x}) = \sigma_\ell(W_\ell \mathbf{v}(\mathbf{x}) + (\mathcal{K}_\ell \mathbf{v})(\mathbf{x})),$$

where W_ℓ is a pointwise linear map on channel features, σ_ℓ a nonlinear activation, and \mathcal{K}_ℓ an integral operator of the form

$$(\mathcal{K}_\ell \mathbf{v})(\mathbf{x}) = \int_{\Omega} \kappa_\ell(\mathbf{x}, \mathbf{y}; \theta) \mathbf{v}(\mathbf{y}) \, d\mathbf{y}.$$

This kernel formulation abstracts many neural operator architectures, particularly those built on global integral or spectral representations.

Among these, the FNO is one of the most influential. In translation-invariant settings it replaces the kernel integral by a spectral convolution: with \mathbf{v}_ℓ the latent state at layer ℓ , the update takes the form

$$\mathbf{v}_{\ell+1}(\mathbf{x}) = \sigma(W_\ell \mathbf{v}_\ell(\mathbf{x}) + \mathcal{F}^{-1}(R_\ell(\mathbf{k}) \widehat{\mathbf{v}}_\ell(\mathbf{k}))(\mathbf{x})),$$

where $\widehat{\mathbf{v}}_\ell = \mathcal{F}(\mathbf{v}_\ell)$ denotes the Fourier transform of \mathbf{v}_ℓ , \mathcal{F}^{-1} is the inverse transform, and $R_\ell(\mathbf{k})$ is a learnable matrix-valued multiplier in Fourier space. In practice, $R_\ell(\mathbf{k})$ is nonzero only on a prescribed band of low-frequency modes, so the FNO acts as a learned global spectral low-pass filter alongside a local pointwise linear path. This is very efficient for smooth problems and for dynamics dominated by large-scale coherent structures, since the Fourier basis captures long-range interactions naturally. Because the representation is global and truncated in frequency, however, its inductive bias is poorly aligned with the localized nonsmooth structures of shock-dominated conservation laws.

For time-dependent PDEs it helps to distinguish the solution operator from the discrete flow map. Let $\mathbf{u}(\cdot, t)$ solve the conservation law with initial data \mathbf{u}_0 ; the continuous solution operator at time t is

$$\mathcal{S}_t : \mathcal{X} \rightarrow \mathcal{X}, \quad \mathbf{u}_0 \mapsto \mathbf{u}(\cdot, t).$$

Equivalently, for a time increment Δt , the associated flow map is written formally as

$$\mathcal{G}_{\Delta t} : \mathcal{X} \rightarrow \mathcal{X}, \quad \mathbf{u}^n \mapsto \mathbf{u}^{n+1}.$$

Operator learning may target either the full solution operator \mathcal{S}_t or the one-step flow map $\mathcal{G}_{\Delta t}$.

In this work we adopt the latter viewpoint, with a fixed Δt . For a mesh with spacing h , let \mathcal{X}_h be the finite-dimensional space of grid functions; the exact discrete flow map is

$$\mathcal{G}_{\Delta t, h} : \mathcal{X}_h \rightarrow \mathcal{X}_h, \quad \mathbf{u}_h^n \mapsto \mathbf{u}_h^{n+1}.$$

This grid-function perspective matches time advancement in conventional numerical schemes and is the setting of our numerical experiments. For brevity we drop the subscript h when no ambiguity arises.

A simple residual parameterization of the learned flow map is

$$\mathcal{G}_\theta(\mathbf{u}) = \mathbf{u} + \Delta t \Phi_\theta(\mathbf{u}; \Delta t), \tag{5}$$

where Φ_θ models the increment of the discrete solution over one time step Δt .

Given one-step training data

$$\mathcal{D} = \{(\mathbf{u}_i, \mathbf{v}_i)\}_{i=1}^N, \quad \mathbf{v}_i = \mathcal{G}_{\Delta t}(\mathbf{u}_i),$$

generated by a high-order, well-resolved numerical scheme $\mathcal{G}_{\Delta t}$ (whose internal time step due to CFL is typically tens to a hundred times smaller than the sampled increment Δt), we seek a neural approximation \mathcal{G}_θ of this discrete flow map. The parameters θ are determined by minimizing

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^N d_{\mathcal{X}}(\mathcal{G}_\theta(\mathbf{u}_i), \mathbf{v}_i),$$

where $d_{\mathcal{X}}$ is a distance on the state space, for example an L^1 -type loss. When trajectory data are available, the objective may also include multi-step rollout losses, but the goal is unchanged: to learn an operator \mathcal{G}_θ that accurately approximates the discrete flow map while respecting the structure of the underlying conservation law.

3. Local-Global Neural Operator (LGNO) for Conservation Laws

3.1. Conservative One-Step Flow Map Parameterization

We adopt the residual parameterization of the discrete flow map in Eq. (5), in which Φ_θ gives the cell-centered increment over one step. The network output is thus interpreted neither as the physical flux $\mathbf{F}(\mathbf{u})$ nor as a classical finite-volume numerical flux. We represent Φ_θ with the LGNO architecture introduced below, or, for the baseline, with an FNO.

When the underlying conservation law preserves the spatial average, as under periodic boundary conditions, we enforce the same property by projecting the learned increment onto the mean-zero subspace. Writing $\langle \cdot \rangle$ for the spatial average, computed over grid cells in the discrete setting and applied component-wise for systems,

$$\langle \mathbf{v} \rangle = |\Omega|^{-1} \int_{\Omega} \mathbf{v}(\mathbf{x}) d\mathbf{x}, \quad \Pi_0 \Phi_\theta = \Phi_\theta - \langle \Phi_\theta \rangle, \quad \mathcal{G}_\theta(\mathbf{u}) = \mathbf{u} + \Delta t \Pi_0 \Phi_\theta(\mathbf{u}; \Delta t).$$

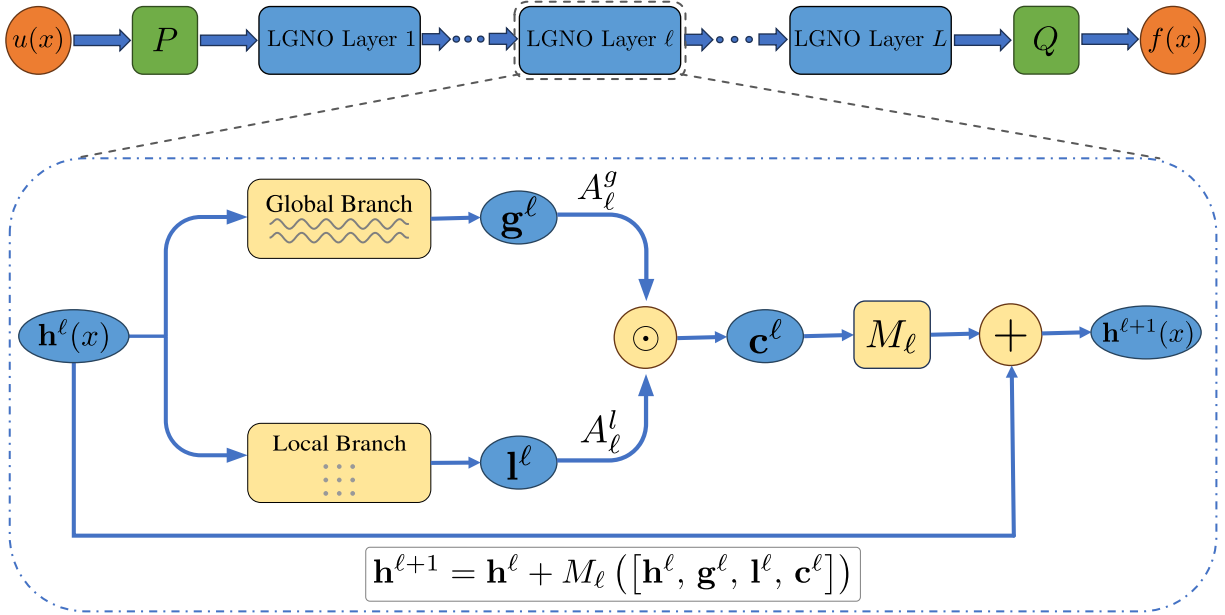


Figure 2: **Schematic of the LGNO architecture.** The input state \mathbf{u}^n is first lifted to a latent feature field \mathbf{h}^1 by a pointwise map. Each hidden layer has two branches: the global spectral branch applies learnable Fourier multipliers to the low-frequency modes, while the local branch extracts localized features by convolution. Their outputs are coupled multiplicatively and then fused with the incoming hidden state through a pointwise residual mixing map. After L such layers, a pointwise output network maps the final latent feature \mathbf{h}^{L+1} to \mathbf{f}_θ , which parameterizes the learned one-step update.

Proposition 3.1 (Exact global conservation). *With the projected update $\mathcal{G}_\theta(\mathbf{u}) = \mathbf{u} + \Delta t \Pi_0 \Phi_\theta(\mathbf{u})$ above, for any network parameters θ , time step Δt , and input state \mathbf{u} ,*

$$\langle \mathcal{G}_\theta(\mathbf{u}) \rangle = \langle \mathbf{u} \rangle,$$

so the learned flow map preserves the total of each conserved component exactly.

This holds by construction, since $\langle \Pi_0 \Phi_\theta \rangle = 0$; it is an exact algebraic property, at both the continuous and discrete levels.

Remark 3.2 (Flux-form parameterization). A natural alternative is to learn a flux-form update as in Eq. (3), but we found that this did not significantly improve accuracy. Moreover, when only solution snapshots are available, the numerical flux is not uniquely determined, since the discrete divergence has a nontrivial null space; recovering a unique flux then requires an additional constraint, which complicates the framework without a clear benefit. We therefore adopt the simpler residual increment with the global conservation projection above.

3.2. Local-Global Operator Architecture

We now specify the LGNO architecture that represents the update Φ_θ . It combines a global spectral branch, which captures smooth large-scale dynamics, with a local branch, which resolves sharp localized features; its overall structure is summarized in Figure 2.

Given a discrete state \mathbf{u}^n , a pointwise lifting map produces an initial latent feature field

$$\mathbf{h}^1 = \sigma(P_\theta(\mathbf{u}^n)),$$

where, at each grid point, P_θ applies the same affine map to the channel vector. Here “pointwise” refers to the spatial action, while the channel dimension is lifted from d to d_h ; this is equivalently a convolution with a 1×1 kernel. The nonlinear activation σ then yields the latent feature $\mathbf{h}^1 \in \mathbb{R}^{d_h \times N}$.

Each of the L hidden layers combines a global spectral branch and a local multiresolution branch, which are coupled multiplicatively and then fused with the incoming state by a residual map. Let \mathbf{h}^ℓ denote the hidden feature at layer ℓ ; for a discretization with N grid points, $\mathbf{h}^\ell \in \mathbb{R}^{d_h \times N}$, with discrete Fourier transform $\hat{\mathbf{h}}^\ell \in \mathbb{C}^{d_h \times N}$.

Global branch. The spectral branch is defined as

$$\mathbf{g}^\ell = \sigma \left(W_\ell^s \mathbf{h}^\ell + \mathcal{F}^{-1} \left(R_\ell(\mathbf{k}) \widehat{\mathbf{h}}^\ell(\mathbf{k}) \right) \right). \quad (6)$$

Here \mathcal{F} denotes the discrete Fourier transform. We write $R_\ell(\mathbf{k}) \in \mathbb{C}^{d_h \times d_h}$ for the Fourier multiplier at mode \mathbf{k} . For each frequency \mathbf{k} , $R_\ell(\mathbf{k})$ acts on the channel vector $\widehat{\mathbf{h}}(\mathbf{k}) \in \mathbb{C}^{d_h}$ as

$$(R_\ell(\mathbf{k}) \widehat{\mathbf{h}}(\mathbf{k}))_i = \sum_{j=1}^{d_h} R_{\ell,i,j}(\mathbf{k}) \widehat{h}_j(\mathbf{k}), \quad i = 1, \dots, d_h.$$

In practice, R_ℓ carries learnable entries only on a prescribed band of low-frequency modes and is zero elsewhere, so the spectral branch acts as a trainable low-pass filter. The term $W_\ell^s \in \mathbb{R}^{d_h \times d_h}$ is a pointwise linear map on the channel vector at each location. For periodic boundary conditions the spectral convolution is applied directly; for nonperiodic settings, the feature is padded before the transform as a practical boundary treatment.

Local branch. We propose to include a local branch to incorporate stencil-type locality, a key inductive bias in conventional high-order schemes for conservation laws. This locality is important for shocks, contact discontinuities, and sharp gradients, whose evolution is governed by local wave interactions and may be difficult to capture using a global spectral operator. At layer ℓ , the local feature is defined by

$$\mathbf{l}^\ell = \sigma(\mathcal{C}_\ell(\mathbf{h}^\ell)), \quad \mathbf{l}^\ell \in \mathbb{R}^{d_h \times N}, \quad (7)$$

where $\mathcal{C}_\ell : \mathbb{R}^{d_h \times N} \rightarrow \mathbb{R}^{d_h \times N}$ is a local multiresolution operator.

The multiresolution design is inspired by the coarse-to-fine feature-fusion strategy of U-Net [29]. Instead of applying local convolutions only on the original grid, we first restrict the hidden feature to a coarser grid, where a fixed-size convolution kernel corresponds to a larger effective neighborhood in the original physical mesh. This enlarges the effective receptive field while keeping the additional computational cost small. The processed coarse feature is then interpolated back to the original resolution and fused with the original feature:

$$\mathcal{C}_\ell(\mathbf{h}) = W_\ell^c [\mathbf{h}, \mathcal{I}_\ell \mathcal{K}_\ell \mathcal{P}_\ell \mathbf{h}]. \quad (8)$$

Here \mathcal{P}_ℓ denotes average pooling by a factor of two, \mathcal{I}_ℓ denotes linear interpolation back to the original resolution, and W_ℓ^c is a pointwise linear map that fuses the fine- and coarse-resolution features.

In our implementation, \mathcal{K}_ℓ is a two-layer convolutional network on the coarse grid:

$$\mathcal{K}_\ell(\mathbf{z}) = \sigma(\text{Conv}_{\ell,2}(\sigma(\text{Conv}_{\ell,1}(\mathbf{z}))))), \quad \mathbf{z} = \mathcal{P}_\ell \mathbf{h}. \quad (9)$$

Here $\text{Conv}_{\ell,1}$ and $\text{Conv}_{\ell,2}$ denote learned convolutional operations on the coarse grid, each including its convolution kernel and bias term. The convolutions use small kernels with boundary padding chosen according to the prescribed boundary condition, so that the coarse-grid resolution is preserved. Thus, the local branch injects stencil-type information while using the coarse grid to provide a larger effective receptive field at low cost.

LGNO layer. The global and local features are then coupled through an interaction operator. In the present architecture, we use multiplicative coupling and define

$$\mathbf{c}^\ell = (A_\ell^g \mathbf{g}^\ell) \odot (A_\ell^l \mathbf{l}^\ell), \quad \mathbf{c}^\ell \in \mathbb{R}^{d_h \times N}, \quad (10)$$

where $A_\ell^g, A_\ell^l \in \mathbb{R}^{d_h \times d_h}$ are pointwise linear maps on the channel vector and \odot denotes componentwise multiplication. This interaction allows the local response to be modulated by the global spectral feature, while preserving the branchwise separation of the two representations.

The hidden state is updated by

$$\mathbf{h}^{\ell+1} = \mathcal{L}_\ell(\mathbf{h}^\ell) := \mathbf{h}^\ell + M_\ell([\mathbf{h}^\ell, \mathbf{g}^\ell, \mathbf{l}^\ell, \mathbf{c}^\ell]), \quad \ell = 1, \dots, L, \quad (11)$$

where $\mathbf{h}^{\ell+1} \in \mathbb{R}^{d_h \times N}$, $[\mathbf{h}^\ell, \mathbf{g}^\ell, \mathbf{l}^\ell, \mathbf{c}^\ell] \in \mathbb{R}^{4d_h \times N}$, and $M_\ell \in \mathbb{R}^{d_h \times 4d_h}$ is a pointwise linear map on the concatenated channel vector. This residual update lets each layer refine the latent representation while retaining information from the incoming state.

Finally, after L layers, a pointwise output map Q_θ produces the field required by the update rule:

$$\mathbf{f}_\theta = Q_\theta(\mathbf{h}^{L+1}). \quad (12)$$

Here $Q_\theta : \mathbb{R}^{d_h} \rightarrow \mathbb{R}^d$ is applied at each grid point, implemented as a two-layer pointwise network with a nonlinear activation in between. The output \mathbf{f}_θ parameterizes the learned increment in the discrete flow map.

A concrete implementation of one LGNO layer \mathcal{L}_ℓ is summarized in Algorithm 3.1.

Algorithm 3.1 Implementation of one LGNO layer \mathcal{L}_ℓ

Require: Hidden feature field $\mathbf{h}^\ell \in \mathbb{R}^{d_h \times N}$; Fourier multiplier R_ℓ ; pointwise linear maps $W_\ell^s, W_\ell^c, A_\ell^g, A_\ell^l, M_\ell$; average-pooling operator \mathcal{P}_ℓ ; coarse-grid convolutional network \mathcal{K}_ℓ ; interpolation operator \mathcal{I}_ℓ

Ensure: Updated hidden feature field $\mathbf{h}^{\ell+1}$

Global spectral branch

- 1: Compute the global feature

$$\mathbf{g}^\ell \leftarrow \sigma \left(W_\ell^s \mathbf{h}^\ell + \mathcal{F}^{-1} \left(R_\ell(\mathbf{k}) \widehat{\mathbf{h}}^\ell(\mathbf{k}) \right) \right)$$

Local multiresolution branch

- 2: Compute the local feature

$$\mathbf{l}^\ell \leftarrow \sigma \left(W_\ell^c \left[\mathbf{h}^\ell, \mathcal{I}_\ell \mathcal{K}_\ell \mathcal{P}_\ell \mathbf{h}^\ell \right] \right)$$

Global–local interaction and residual update

- 3: Compute the coupling

$$\mathbf{c}^\ell \leftarrow (A_\ell^g \mathbf{g}^\ell) \odot (A_\ell^l \mathbf{l}^\ell)$$

- 4: Update the hidden state

$$\mathbf{h}^{\ell+1} \leftarrow \mathbf{h}^\ell + M_\ell \left([\mathbf{h}^\ell, \mathbf{g}^\ell, \mathbf{l}^\ell, \mathbf{c}^\ell] \right)$$

- 5: **return** $\mathbf{h}^{\ell+1}$
-

3.3. Boundary Padding for Outflow

For two-dimensional outflow problems, waves and coherent structures may reach the computational boundary during autoregressive rollout. In this regime, simple boundary copying or fixed extrapolation can provide inaccurate ghost values and introduce artifacts into local convolutional stencils. We therefore use a learned ghost-cell padding operator for the local convolutional operations under outflow boundary conditions. This padding is applied to latent feature maps in the local branch.

Let $\mathbf{x} \in \mathbb{R}^{B \times d_h \times H \times W}$ be a latent feature map before a local convolution, where B is the batch size, d_h is the number of channels, and $H \times W$ is the spatial resolution. For an odd convolution kernel of size k , we set the ghost width to

$$g_w = \frac{k-1}{2}.$$

The padding operator constructs ghost features from boundary-adjacent interior information before applying the convolution.

Given a context width c , we first predict the left and right ghost columns from boundary-adjacent interior strips:

$$\mathbf{g}_L = \mathcal{G}_L(\mathbf{x}_{:, :, :, 1:c}), \quad \mathbf{g}_R = \mathcal{G}_R(\mathbf{x}_{:, :, :, W-c+1:W}),$$

where \mathcal{G}_L and \mathcal{G}_R are shallow local convolutional networks whose outputs satisfy

$$\mathbf{g}_L, \mathbf{g}_R \in \mathbb{R}^{B \times d_h \times H \times g_w}.$$

The horizontally padded feature map is then

$$\mathbf{x}_{LR} = [\mathbf{g}_L, \mathbf{x}, \mathbf{g}_R] \in \mathbb{R}^{B \times d_h \times H \times (W+2g_w)}.$$

We next predict the top and bottom ghost rows from boundary-adjacent strips of \mathbf{x}_{LR} :

$$\mathbf{g}_T = \mathcal{G}_T((\mathbf{x}_{LR})_{:,1:c,:}), \quad \mathbf{g}_B = \mathcal{G}_B((\mathbf{x}_{LR})_{:,H-c+1:H,:}),$$

where \mathcal{G}_T and \mathcal{G}_B are defined analogously and output

$$\mathbf{g}_T, \mathbf{g}_B \in \mathbb{R}^{B \times d_h \times g_w \times (W+2g_w)}.$$

Thus the corner ghost features are generated as part of the top and bottom ghost rows. The final padded feature map is

$$\text{Pad}_{\text{out}}(\mathbf{x}) = [\mathbf{g}_T; \mathbf{x}_{LR}; \mathbf{g}_B] \in \mathbb{R}^{B \times d_h \times (H+2g_w) \times (W+2g_w)}.$$

Here $[\cdot, \cdot]$ denotes horizontal concatenation along the width dimension, and $[\cdot; \cdot]$ denotes vertical concatenation along the height dimension.

The operator Pad_{out} is applied before each local convolution in the local branch. The subsequent convolution is performed with stride one and without additional zero padding, so that the output has the same spatial resolution as the input feature map. This learned padding provides a lightweight treatment of outflow boundaries for the neural local stencil, allowing the local branch to maintain a complete receptive field near the boundary without injecting zero-padding artifacts.

3.4. Training Loss

The architecture above induces a discrete flow map \mathcal{G}_θ for the one-step update, which we train in a supervised manner on snapshot pairs generated by a high-order numerical scheme. Our training objective combines a physical-space prediction loss with a high-frequency spectral penalty that suppresses oscillatory error near discontinuities.

Let \mathbf{u}^{n+1} and $\hat{\mathbf{u}}_\theta^{n+1}$ denote the reference and predicted next states on the discrete grid Ω_h , and let d be the number of physical components. The physical-space loss is

$$\mathcal{L}_{\text{phys}} = \begin{cases} \frac{1}{|\Omega_h|} \|\hat{\mathbf{u}}_\theta^{n+1} - \mathbf{u}^{n+1}\|_{L^1}, & d = 1, \\ \frac{1}{d} \sum_{m=1}^d \frac{\|\hat{\mathbf{u}}_{\theta,m}^{n+1} - \mathbf{u}_m^{n+1}\|_{L^1}}{\|\mathbf{u}_m^{n+1}\|_{L^1} + \varepsilon}, & d > 1, \end{cases} \quad (13)$$

where $\varepsilon > 0$ is a small constant for numerical stability. Thus, scalar problems use the standard mean absolute error, while multi-component systems use a componentwise relative L^1 loss to balance variables with different physical scales.

To suppress high-frequency oscillatory error, let $\mathbf{e} = \hat{\mathbf{u}}_\theta^{n+1} - \mathbf{u}^{n+1}$ be the one-step prediction error and $\hat{\mathbf{e}}(k) = \mathcal{F}(\mathbf{e})(k)$ its discrete Fourier coefficient at mode k . With K the number of retained Fourier modes and $k_0 = \lfloor \kappa K \rfloor$ for a prescribed fraction $\kappa \in (0, 1)$, we define

$$\mathcal{L}_{\text{hf}} = \frac{1}{K - k_0} \sum_{k=k_0}^{K-1} \|\hat{\mathbf{e}}(k)\|_2^2. \quad (14)$$

Here $\|\hat{\mathbf{e}}(k)\|_2^2 = \sum_{m=1}^d |\hat{e}_m(k)|^2$ for multi-component systems, and reduces to $|\hat{e}(k)|^2$ when $d = 1$. The total training loss is

$$\mathcal{L} = \mathcal{L}_{\text{phys}} + \lambda_{\text{hf}} \mathcal{L}_{\text{hf}}, \quad (15)$$

where $\lambda_{\text{hf}} > 0$ controls the strength of the spectral penalty.

Remark 3.3 (Symmetry-based extension of the state range). Since the learned flow map is applied at inference time, exact scaling symmetries of the governing equations can be used as preprocessing and postprocessing transformations. Such symmetries are present in several homogeneous hyperbolic conservation laws, including inviscid Burgers, the shallow-water equations, and the compressible Euler equations, although the precise scaling weights are equation-dependent. When such a symmetry is available, a test state whose magnitude lies outside the training range can be mapped into a comparable range, advanced by the learned operator in the rescaled variables, and then mapped back, preserving the corresponding dynamics. We use this idea in the two-dimensional Euler shock tests in [Appendix A](#), where the detailed transformation is given.

Table 1: One-step test relative L^1 errors for the proposed LGNO model and the FNO baseline, reported as mean \pm standard deviation over the test set. The last column gives the mean-error ratio FNO/LGNO, where values larger than 1 indicate better performance of LGNO. LGNO consistently outperforms the FNO baseline across all reported benchmarks, reducing the mean error by factors between 2.18 and 5.09.

Problem	LGNO	FNO	FNO/LGNO
Linear advection	$4.57 \times 10^{-4} \pm 1.16 \times 10^{-3}$	$1.99 \times 10^{-3} \pm 2.91 \times 10^{-3}$	4.36
Burgers 1D	$2.38 \times 10^{-4} \pm 3.94 \times 10^{-4}$	$5.19 \times 10^{-4} \pm 5.96 \times 10^{-4}$	2.18
SWE 1D	$5.27 \times 10^{-4} \pm 5.41 \times 10^{-4}$	$1.16 \times 10^{-3} \pm 1.23 \times 10^{-3}$	2.20
Euler 1D	$1.23 \times 10^{-3} \pm 1.30 \times 10^{-3}$	$3.85 \times 10^{-3} \pm 3.74 \times 10^{-3}$	3.13
Burgers 2D	$3.53 \times 10^{-4} \pm 2.69 \times 10^{-4}$	$9.81 \times 10^{-4} \pm 8.63 \times 10^{-4}$	2.78
Euler 2D periodic	$3.87 \times 10^{-3} \pm 2.31 \times 10^{-3}$	$1.97 \times 10^{-2} \pm 8.86 \times 10^{-3}$	5.09
Euler 2D outflow	$1.67 \times 10^{-3} \pm 1.07 \times 10^{-3}$	$7.33 \times 10^{-3} \pm 3.08 \times 10^{-3}$	4.39

4. Numerical Results

In this section, we evaluate the proposed LGNO on several benchmark problems: linear advection, Burgers, shallow water, and compressible Euler equations. All problems use periodic boundary conditions except the final two-dimensional Euler case, which considers both periodic and outflow boundary conditions. Additional two-dimensional Euler shock tests, based on standard two-dimensional Riemann problems not included in the training data, are reported in [Appendix A](#) as a further robustness check in strongly discontinuous regimes.

We compare LGNO against a parameter-matched FNO baseline, using comparable parameter counts on every benchmark except the two-dimensional Burgers case, on which the FNO baseline is substantially larger. At the single-step level, LGNO already reduces the mean relative L^1 error by factors of 2.18 to 5.09 (Table 1). These per-step gains compound under autoregressive rollout, so that the long-time advantage of LGNO is far larger than the one-step errors alone would suggest. LGNO remains stable and accurate over long horizons, whereas the FNO baseline degrades steadily and, in some of the two-dimensional Euler cases, fails entirely by producing unphysical solutions. We examine this long-horizon behavior and the solution quality for each benchmark in turn.

A wall-clock cost comparison against WENO-Z, reported in [Appendix B.2](#), shows that the learned models substantially reduce inference cost relative to the WENO-Z scheme. Model, training, and benchmark configurations are detailed in [Appendix B.1](#) and [Appendix B.3](#).

4.1. Linear Advection Equation

We begin with the linear advection equation

$$u_t + u_x = 0, \quad x \in [0, 1]. \quad (16)$$

Although the dynamics are simple, accurate long-time prediction remains nontrivial for discontinuous data: the solution is transported without intrinsic smoothing, so any representation error in the profile propagates directly over time. For this reason it is a standard benchmark for assessing both accuracy and numerical dissipation in scheme design, and here it provides a clean test of whether LGNO improves on a purely spectral baseline when the target is poorly represented by a truncated Fourier series.

We evaluate the LGNO and FNO models under autoregressive rollout of a representative discontinuous initial condition up to $T = 3$. With the learned time step $\Delta t = 0.05$, this amounts to a long rollout of 60 steps. Since $T = 3$ corresponds to three full periods of the unit-speed equation, the reference solution coincides with the initial condition. As shown in [Figure 3](#), although both models use a conservative update, the FNO baseline accumulates noticeable amplitude drift and fails to maintain the flat states, whereas LGNO stays closely aligned with the reference. This drift is not inconsistent with mass conservation, since the conservative update preserves the total mass but not the profile shape. The FNO jump is also noticeably displaced from its exact location, whereas LGNO matches it well. The relative L^1 error curve confirms this, remaining consistently below the FNO baseline throughout the rollout.

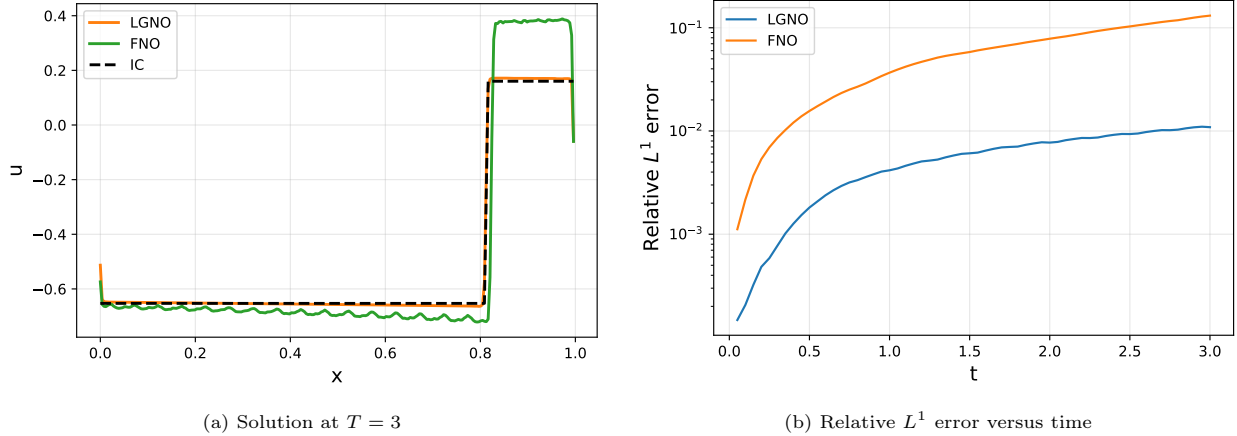


Figure 3: **Long-horizon rollout performance for linear advection.** Panel (a) compares the predictions at $T = 3$ with the reference solution, which coincides with the initial condition due to periodic unit-speed transport. Panel (b) shows the rollout relative L^1 error over time for the same representative discontinuous initial condition.

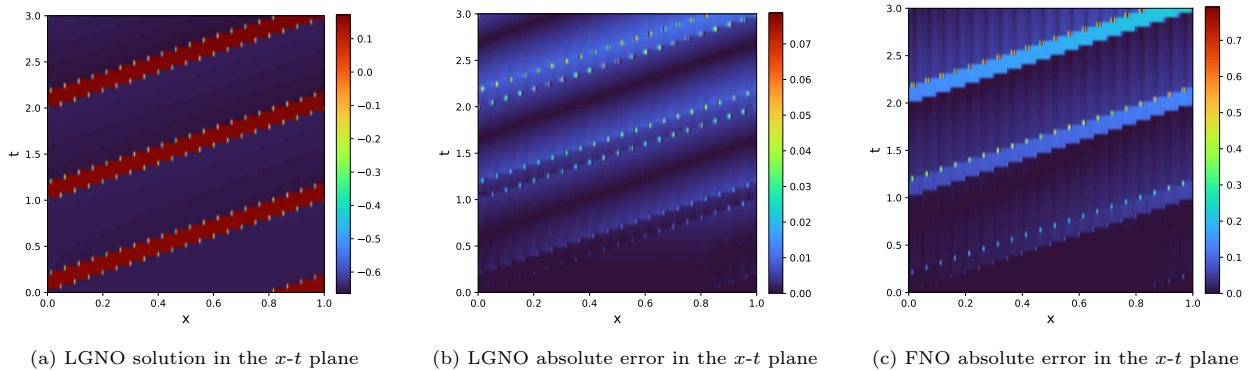


Figure 4: **Spatiotemporal solution and error for linear advection.** Panel (a) shows the LGNO solution in the x - t plane. Panels (b) and (c) compare the spatiotemporal absolute errors of LGNO and the FNO baseline. LGNO preserves the translation pattern better and exhibits significantly smaller errors over the full rollout horizon.

The spatiotemporal plots in Figure 4 reinforce this over the full rollout: the LGNO x - t solution preserves the translation pattern with little visible distortion and its error map stays uniformly small, whereas the FNO baseline shows much larger error accumulation.

Figure 5 compares LGNO and WENO-Z on the same 256-cell mesh at two rollout times, $T = 3$ and $T = 6$, to contrast their dissipation behavior. For this representative discontinuous profile, LGNO is less diffusive than the same-mesh WENO-Z solution. At $T = 6$ (a 120-step rollout), however, LGNO exhibits a small amplitude drift that could be further controlled with a conservative bound-preserving correction if needed.

4.2. One-Dimensional Burgers Equation

We next consider the one-dimensional inviscid Burgers equation

$$u_t + \left(\frac{u^2}{2} \right)_x = 0, \quad x \in [0, 1]. \quad (17)$$

Nonlinear transport and progressive steepening make this a standard test of whether the learned model maintains accuracy as the solution develops increasingly localized structures.

We evaluate both models under autoregressive rollout up to $T = 4$ for a representative, randomly selected initial condition. In the left panel of Figure 6, this initial condition is overlaid as a dashed line on a separate right axis; since its amplitude is much larger than that of the steepened solution at $T = 4$, plotting it on its

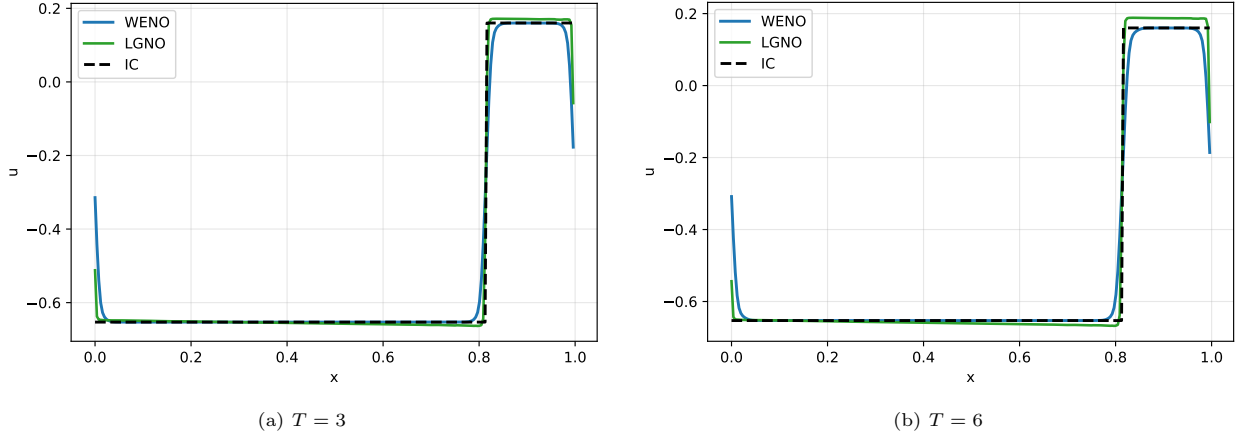


Figure 5: **Comparison of LGNO and WENO-Z in the linear advection problem.** All methods are evaluated on the same 256-cell mesh and compared at two rollout times. LGNO preserves sharper transitions than the same-mesh WENO-Z solution, suggesting less diffusive behavior in this example, though over long times it shows a larger amplitude error than WENO-Z.

own scale keeps the final-time comparison among the reference, LGNO, and FNO more apparent. LGNO stays close to the reference and preserves the main shock and ramp structures, whereas the FNO baseline develops larger local deviations and spurious oscillations near steep fronts. The rollout relative L^1 error of LGNO also remains consistently and substantially lower throughout.

Figure 7 examines the rollout in the $x-t$ plane. LGNO preserves the moving shock and ramp structures over the full interval, and although the dominant errors of both models localize near shocks, LGNO produces smaller error bands than the FNO baseline. Overall, LGNO is more effective for long-time prediction of nonlinear wave dynamics.

4.3. One-Dimensional Shallow Water Equations

The shallow water system is given by

$$\mathbf{U}_t + \mathbf{F}(\mathbf{U})_x = 0, \quad \mathbf{U} = (h, hu)^\top, \quad \mathbf{F}(\mathbf{U}) = \left(hu, hu^2 + \frac{1}{2}gh^2 \right)^\top, \quad x \in [0, 1], \quad g = 9.80665, \quad (18)$$

where h is the fluid height and u the velocity. Compared with the scalar Burgers equation, this system has multiple coupled fields and richer wave interactions, testing whether LGNO remains effective for a multi-component hyperbolic system.

Figure 8 shows predicted solutions at $T = 3$ for two representative test examples; the prediction is performed on the conserved variables, from which the physical variables are recovered at the final time. In each panel, the randomly selected initial condition is overlaid as a dashed line on a separate right axis for both the height and velocity subplots; because it spans a much larger range than the final-time solution, this separate scaling keeps the comparison among the reference, LGNO, and FNO clear.

In both examples, LGNO accurately captures the wave structures in the height and velocity fields, remaining close to the reference in smooth regions and near sharp transitions. The FNO baseline captures the coarse propagation pattern but produces unphysical oscillations in h , including in smooth regions away from any discontinuity, together with visible deviations in u . Its error therefore spreads into the smooth parts of the solution rather than staying localized at the shocks and contact discontinuities. LGNO thus yields a more stable and accurate prediction of the coupled shallow water dynamics.

4.4. One-Dimensional Euler Equations

We then consider the one-dimensional compressible Euler equations for an ideal gas,

$$\mathbf{U}_t + \mathbf{F}(\mathbf{U})_x = 0, \quad x \in [0, 1], \quad \mathbf{U} = (\rho, m, E)^\top, \quad (19)$$

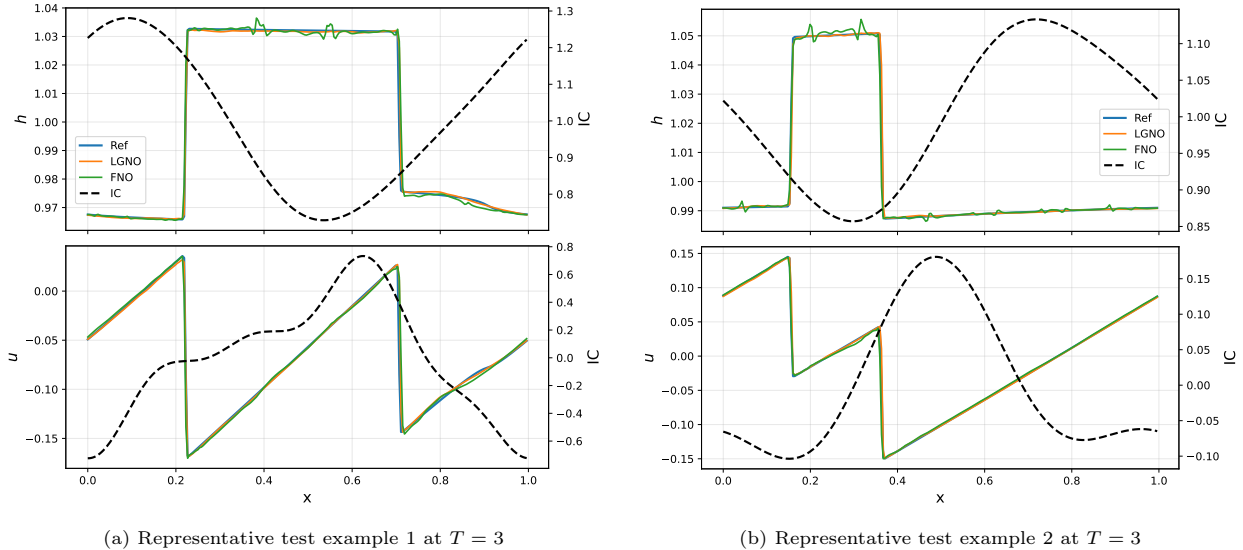


Figure 8: **Final-time solution profiles for the one-dimensional shallow water equations.** Panels (a) and (b) show two representative test examples at $T = 3$. Each panel reports the fluid height h on top and the velocity u on the bottom, with the initial condition shown as a dashed line on a separate right axis. LGNO remains close to the reference solution in both variables, while the FNO baseline shows larger deviations, including unphysical oscillations in h and visible deviations in u .

where

$$\mathbf{F}(\mathbf{U}) = \left(m, \frac{m^2}{\rho} + p, (E + p)u \right)^\top, \quad m = \rho u, \quad p = (\gamma - 1) \left(E - \frac{1}{2} \rho u^2 \right).$$

This is the most demanding one-dimensional example: its coupled conserved variables, nonlinear wave interactions, and sharp features make it a critical test of whether LGNO improves over the FNO baseline.

Figure 9 compares the conserved variables (ρ, m, E) and the recovered velocity and pressure (u, p) at the final time $T = 1$; in each subplot, the initial condition is overlaid as a dashed line on a separate right axis, plotted on its own scale. LGNO stays close to the reference across the conserved variables and yields more accurate velocity and pressure fields, capturing the shocks and contact discontinuities at their correct positions. The FNO baseline, by contrast, develops larger oscillatory errors and does not propagate the shocks at the correct speed. These oscillations are amplified in the velocity and pressure recovered from the conserved fields, while LGNO remains well behaved in both representations, giving sharper transitions with far fewer spurious oscillations.

The spatiotemporal plots in Figure 10 extend this to the full rollout up to $T = 5$, where all panels share a common contour range so that the three solutions are compared on the same scale. LGNO closely follows the reference wave structure and keeps the shocks and contacts on their correct trajectories. Under this common scale, the FNO baseline shows much larger errors than LGNO, with its discontinuity fronts advancing at incorrect speeds. The difference is more pronounced in the velocity and pressure fields, where the FNO baseline becomes overly diffusive, smearing sharp transitions in u and p into broad bands and washing out fine structures by the end of the rollout.

4.5. Two-Dimensional Burgers Equation

We then consider the two-dimensional Burgers equation

$$u_t + \left(\frac{u^2}{2} \right)_x + \left(\frac{u^2}{2} \right)_y = 0, \quad (x, y) \in [0, 1]^2. \quad (20)$$

This extends the study to a genuinely two-dimensional setting, assessing whether LGNO retains its advantage when the solution develops nontrivial two-dimensional structure over long rollouts.

Figure 11 compares a representative one-dimensional slice of the solution at $T = 3$ with the rollout relative L^1 error. LGNO remains close to the reference across the domain, including near steep transitions,

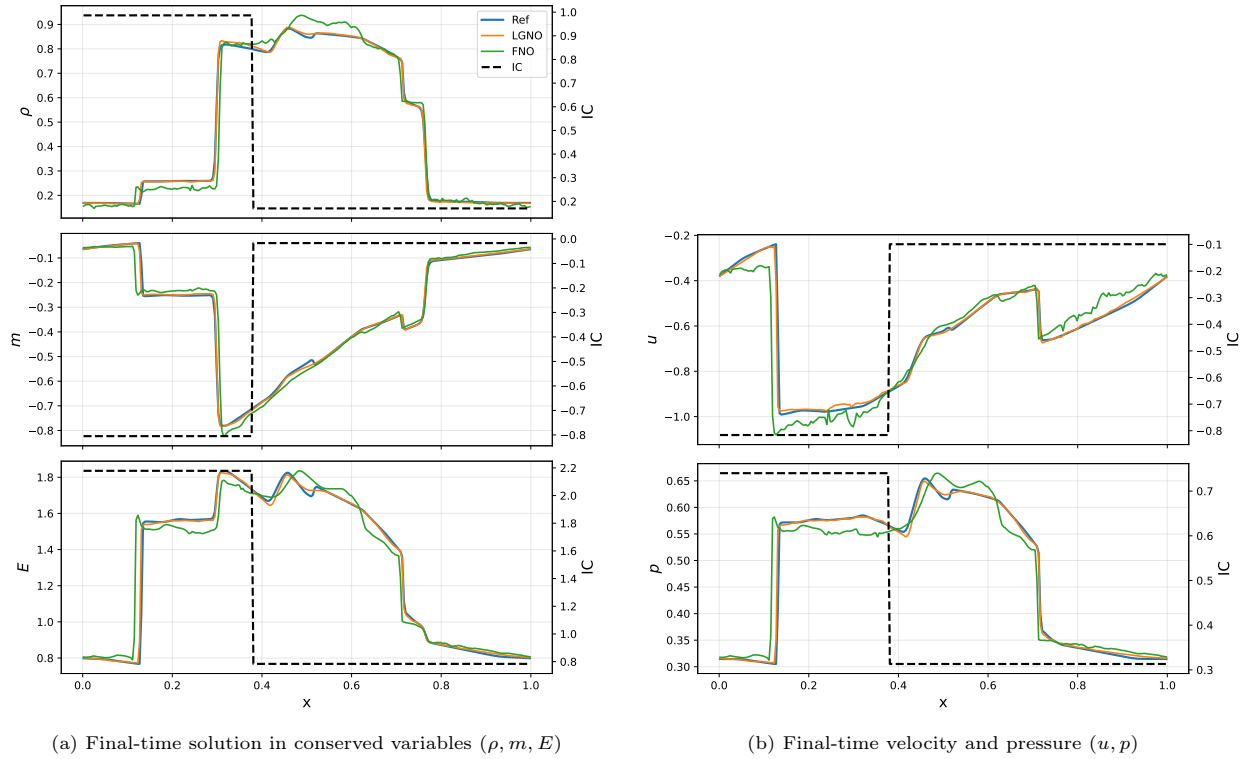


Figure 9: **Solution at $T = 1$ for the one-dimensional Euler equations.** Panel (a) compares the reference solution with the LGNO and FNO predictions in the conserved variables (ρ, m, E) , and panel (b) shows the recovered velocity and pressure (u, p) . In each subplot, the initial condition is shown as a dashed line on a separate right axis. LGNO remains consistently closer to the reference solution and places the shocks and contact discontinuities at their correct positions, whereas the FNO baseline exhibits larger oscillatory errors and fails to propagate the shocks at the correct speed.

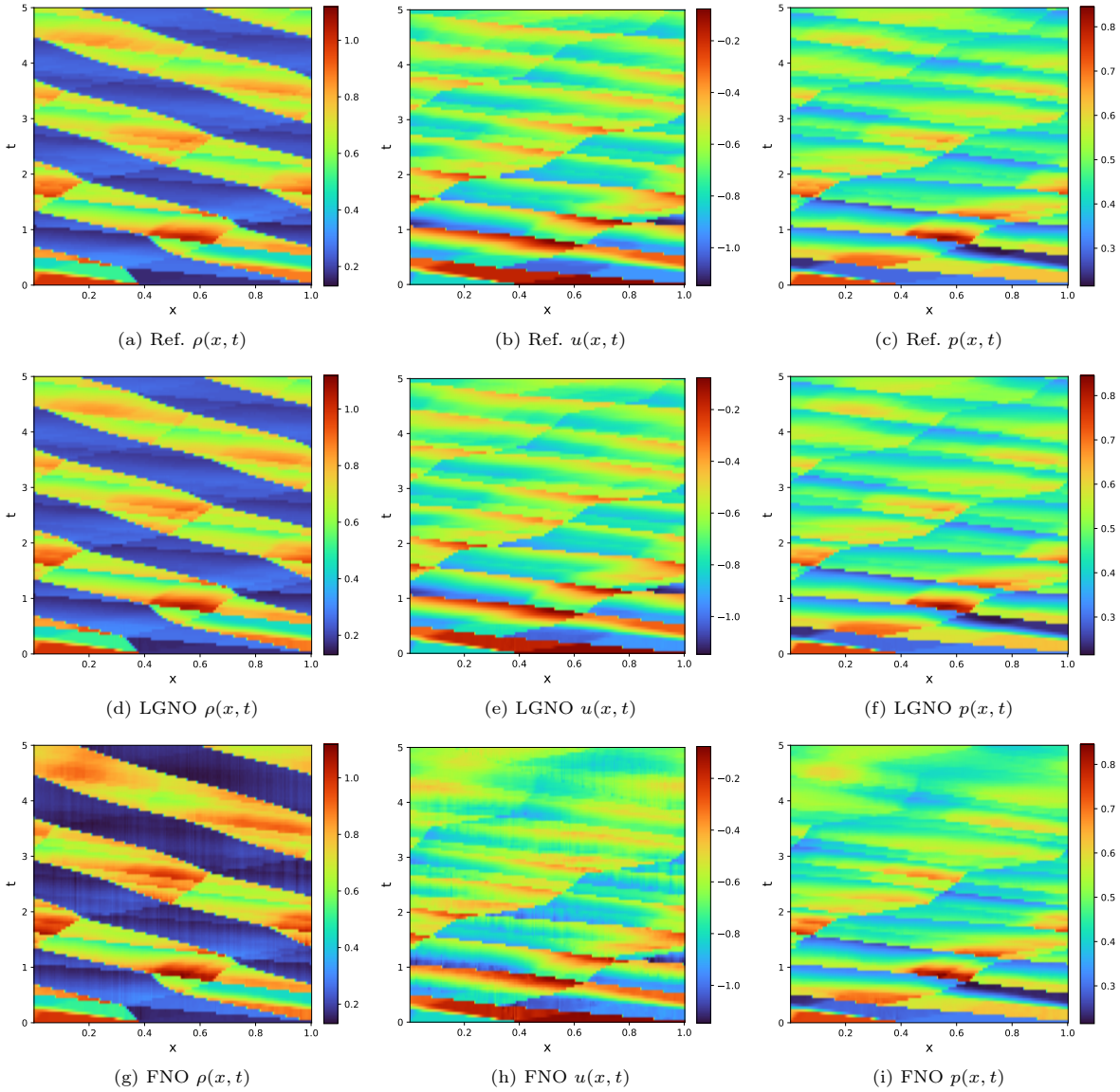


Figure 10: **Spatiotemporal evolution for the one-dimensional Euler equations.** Panels (a)–(c) show the reference solution, panels (d)–(f) show LGNO, and panels (g)–(i) show the FNO baseline. Within each row, the columns correspond to the density $\rho(x, t)$, velocity $u(x, t)$, and pressure $p(x, t)$, and all panels share a common contour range so that the three solutions are shown on the same scale. LGNO closely tracks the reference wave evolution across all three primitive variables, with the shock and contact fronts advancing at the correct speeds. The FNO baseline shows much larger errors than LGNO and becomes overly diffusive over the long rollout.

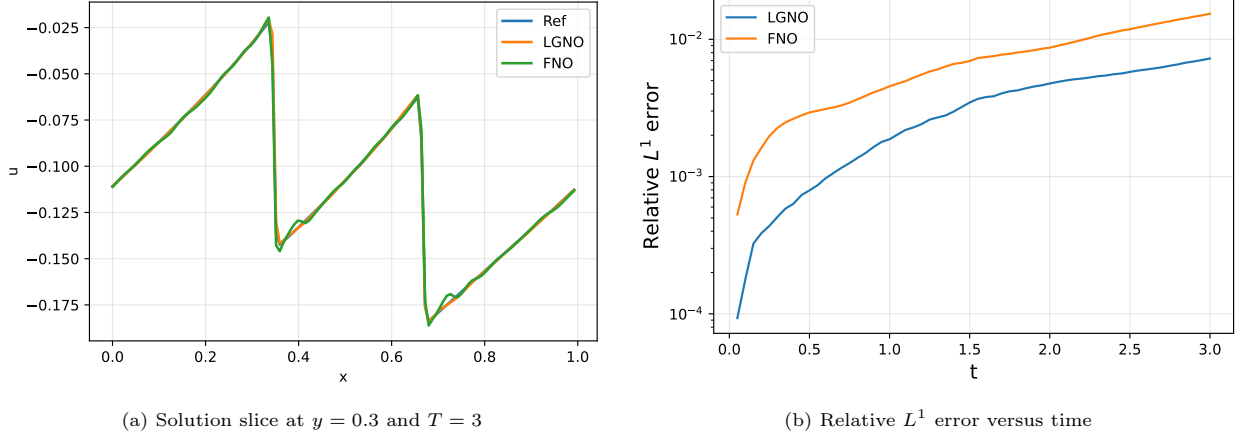


Figure 11: **Long-horizon rollout diagnostics for the two-dimensional Burgers equation.** Panel (a) compares a representative one-dimensional solution slice at $y = 0.3$ and $T = 3$, and panel (b) shows the rollout relative L^1 error over time. LGNO remains closer to the reference near steep transitions and achieves consistently smaller accumulated error than the FNO baseline.

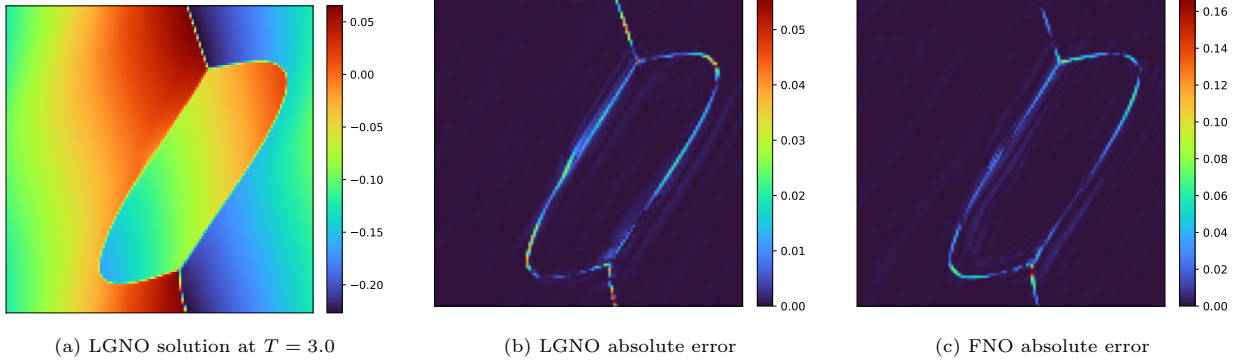


Figure 12: **Solution and error snapshots for the two-dimensional Burgers equation.** Panels (a)–(c) show the results at $T = 3.0$: panel (a) presents the LGNO solution, while panels (b) and (c) compare the absolute error fields of LGNO and the FNO baseline. The dominant errors concentrate near the evolved sharp fronts, but LGNO shows a visibly smaller error magnitude than the FNO baseline.

whereas the FNO baseline shows larger deviations both at the sharp features and in neighboring smooth regions. The error curve confirms this, with LGNO staying consistently below the FNO baseline throughout the rollout.

Figure 12 gives a two-dimensional view at the final time $T = 3$. LGNO captures the evolved structures and curved fronts well; the dominant errors of both models concentrate near these sharp interfaces, but LGNO maintains a visibly smaller error magnitude than the FNO baseline. LGNO thus retains its advantage in the genuinely two-dimensional regime.

4.6. Two-Dimensional Euler Equations with Periodic Boundary Conditions

We next consider the two-dimensional compressible Euler equations

$$\mathbf{U}_t + \mathbf{F}(\mathbf{U})_x + \mathbf{G}(\mathbf{U})_y = 0, \quad (x, y) \in [0, 1]^2, \quad (21)$$

where

$$\mathbf{U} = (\rho, \rho u, \rho v, E)^\top, \quad \mathbf{F}(\mathbf{U}) = (\rho u, \rho u^2 + p, \rho uv, (E + p)u)^\top, \quad \mathbf{G}(\mathbf{U}) = (\rho v, \rho uv, \rho v^2 + p, (E + p)v)^\top.$$

The pressure is given by the ideal-gas equation of state

$$p = (\gamma - 1) \left(E - \frac{1}{2} \rho (u^2 + v^2) \right), \quad \gamma = 1.4. \quad (22)$$

We first study this system under periodic boundary conditions. Table 1 already shows that LGNO achieves a substantially smaller one-step mean relative L^1 error than the FNO baseline. To examine the rollout behavior, we show a representative autoregressive prediction to $T = 0.5$ with $\Delta t = 0.01$, corresponding to 50 rollout steps.

Figure 13 compares the conserved variables from WENO-Z on the 256^2 mesh, the high-resolution WENO-Z reference on the 512^2 mesh, and the learned predictions on the 256^2 mesh. For each variable, all panels use the colorbar limits determined by the high-resolution reference. LGNO stays qualitatively closer to the reference wave pattern and preserves localized structures more clearly than the FNO baseline; notably, the density and momentum fields retain fine vortical and roll-up structures that are weakened in the WENO-Z solutions even on the 512^2 mesh. This indicates a relatively low-dissipation character and points to LGNO’s potential as a learned solver. It should not be read as LGNO being more accurate than the high-resolution reference, but rather as its preserving sharper localized structures under long rollout. The FNO prediction, by contrast, loses much of the coherent structure and develops stronger oscillatory artifacts.

We also verified the primitive variables (ρ, u, v, p) recovered from the conserved fields (not shown for brevity): LGNO remains stable after this conversion, whereas the FNO baseline becomes severely unstable in the velocity fields, with isolated unphysical values of u and v reaching magnitudes of order 10^3 .

4.7. Two-Dimensional Euler Equations with Outflow Boundary Conditions

We finally consider the same two-dimensional Euler system (21)–(22) under outflow boundary conditions. This setting is more challenging due to the boundary condition. In this experiment, the network is trained to predict the primitive variables (ρ, u, v, p) rather than the conserved variables $(\rho, \rho u, \rho v, E)$. Near the outflow boundaries, the local branch uses the learned ghost-padding strategy of Section 3.3 in place of periodic extension, and the positivity of ρ and p is enforced through a generalized softplus parametrization with $\beta = 100$ on the corresponding output channels. All models are trained with learning rate 5×10^{-4} , and additional implementation details are provided in Appendix B.3.6.

Figure 14 shows a representative autoregressive rollout at $T = 0.4$, corresponding to 40 rollout steps. For each primitive variable, the four method panels share the same colorbar range and are therefore directly comparable. LGNO preserves the main wave interactions and discontinuity structures more accurately than the FNO baseline: the density field retains fine roll-up patterns near the interacting wave fronts, whereas the FNO prediction loses much of the coherent structure and develops noisy artifacts. LGNO thus provides a more stable and less dissipative rollout for this challenging outflow problem.

5. Conclusion

We have proposed the Local–Global Neural Operator (LGNO), a neural operator for hyperbolic conservation laws that learns a one-step discrete flow map by coupling a global Fourier branch, which captures smooth large-scale dynamics and long-range interactions, with a local multiresolution branch, which resolves sharp localized structures such as shocks and contact discontinuities. The two branches are combined through a multiplicative interaction, and the model is trained with a one-step loss that augments a physical-space prediction term with a high-frequency spectral penalty to suppress spurious oscillations near steep fronts. For average-preserving boundary conditions, a mean-zero projection makes the learned update exactly conservative, and a simple learned padding extends the local branch to outflow boundaries.

Across several one- and two-dimensional benchmark problems, LGNO consistently outperforms parameter-matched FNO baselines, reducing one-step errors by factors of two to five and remaining markedly more accurate over long autoregressive rollouts. It advances shocks and contact discontinuities at the correct speeds and preserves fine vortical and roll-up structures that the FNO baseline smears or destabilizes. Most notably, although it is trained only on short-time data from a high-order WENO-Z scheme, the long-time rollout of LGNO on a coarse grid exhibits lower numerical dissipation than the same scheme run on a finer grid, while remaining orders of magnitude cheaper to evaluate. This behavior arises because a high-order shock-capturing scheme injects numerical dissipation at every step of a long integration, whereas the learned flow map carries only the low dissipation present in its short-time training data. These results suggest that, with an appropriate architecture and training objective, learned operators have the potential to control long-time numerical dissipation better than the conventional shock-capturing schemes that generate their training data.

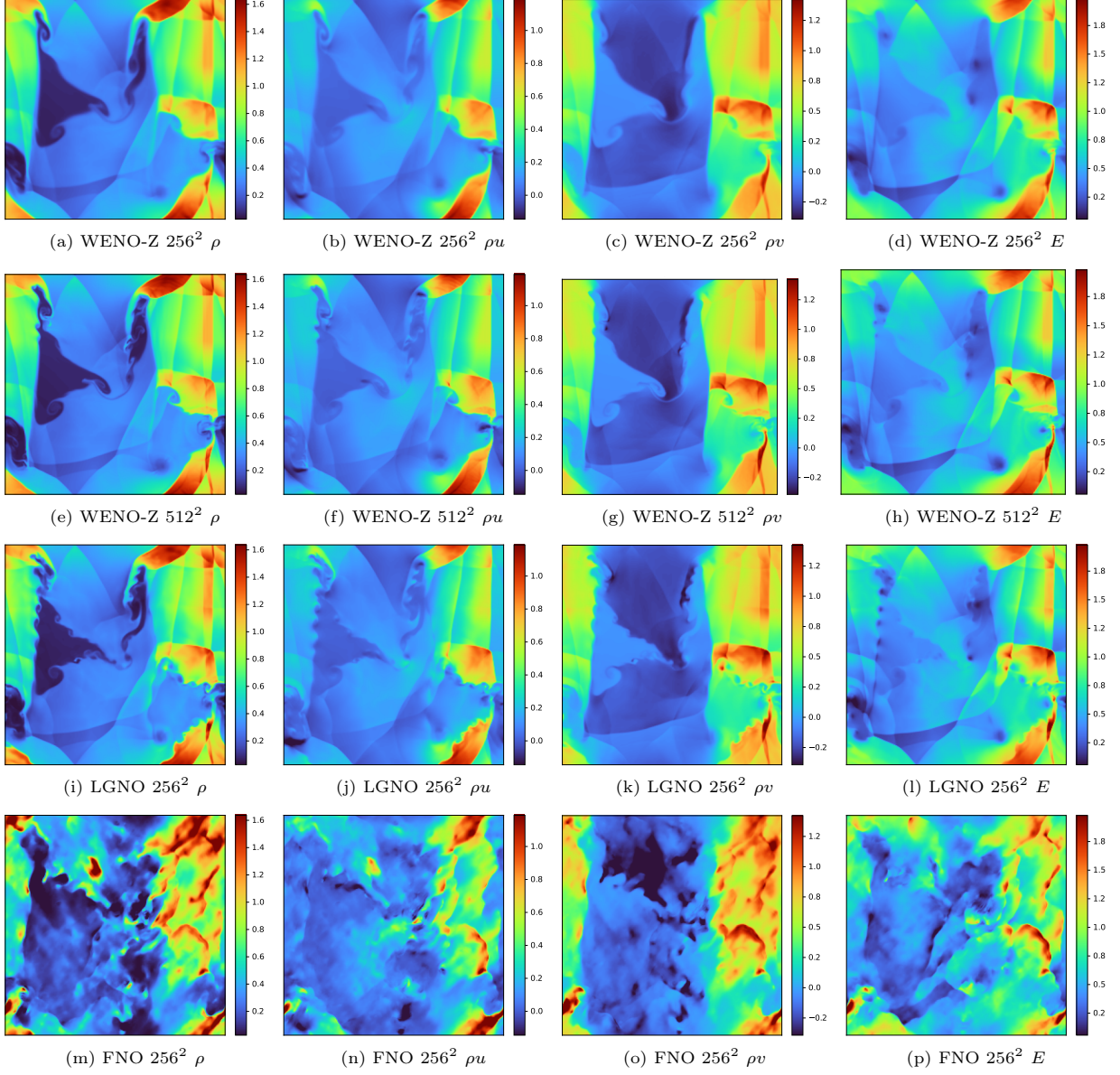


Figure 13: **Representative rollout results for the two-dimensional Euler equations with periodic boundary conditions at $T = 0.5$.** The first row shows the WENO-Z solution on the same 256^2 mesh as the learned models, and the second row shows the high-resolution WENO-Z solution on a 512^2 mesh. The third and fourth rows show the LGNO and FNO predictions on the 256^2 mesh. From left to right, the columns correspond to density, x -momentum, y -momentum, and total energy. For each variable, the four panels in the corresponding column use the same colorbar range, allowing direct comparison across methods. LGNO better preserves the localized structures than the FNO baseline, which loses coherent wave patterns and develops stronger oscillatory artifacts. Notably, LGNO on the 256^2 mesh also retains fine vortical and roll-up structures that are smeared in the WENO-Z solutions even on the finer 512^2 mesh, reflecting its lower long-time numerical dissipation.

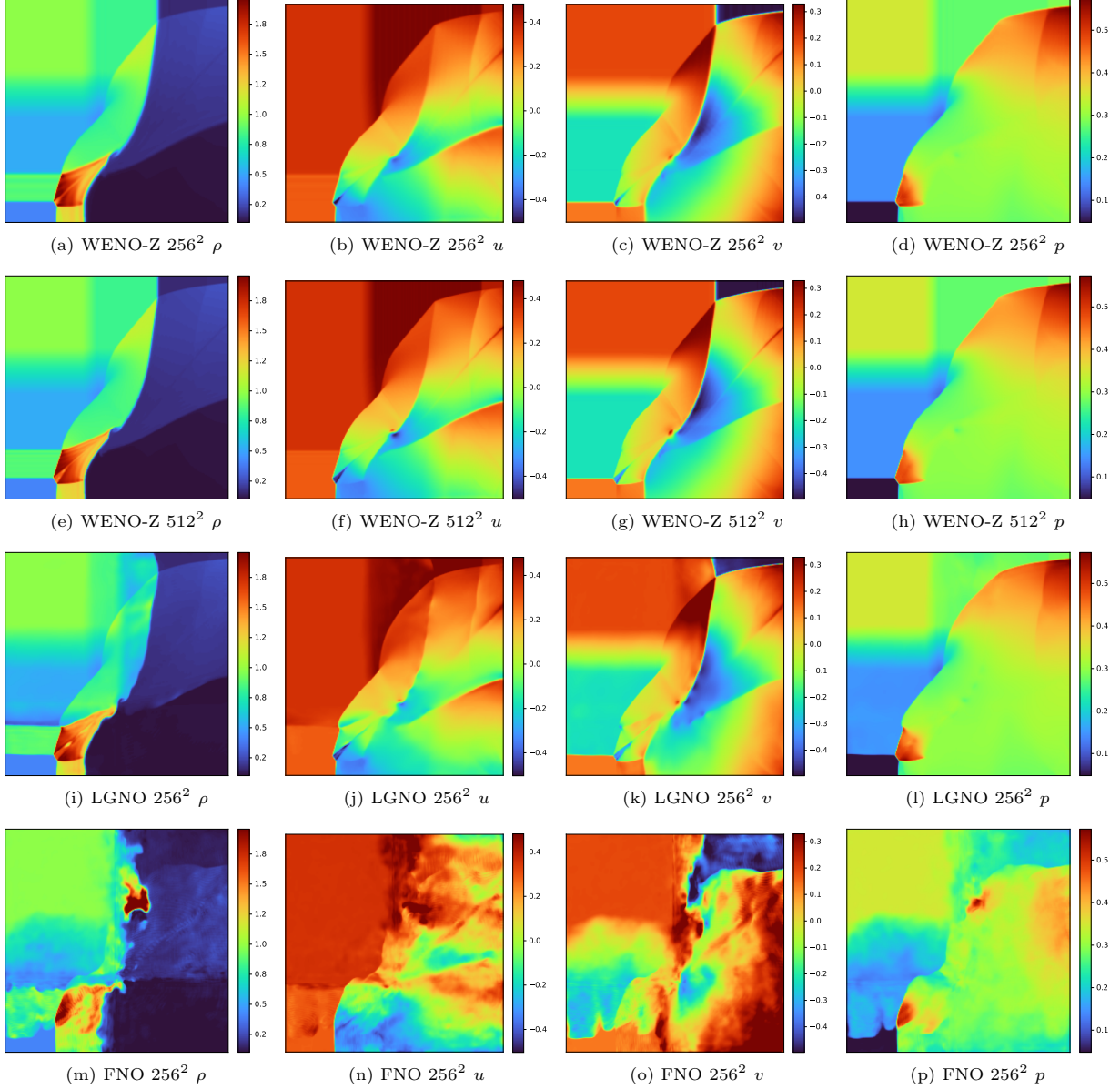


Figure 14: **Representative rollout results for the two-dimensional Euler equations with outflow boundary conditions at $T = 0.4$.** Panels (a)–(d) show the WENO-Z solution on the 256^2 mesh, panels (e)–(h) show the WENO-Z reference solution on the 512^2 mesh, panels (i)–(l) show the LGNO prediction on the 256^2 mesh, and panels (m)–(p) show the FNO prediction on the 256^2 mesh, all after 40 autoregressive rollout steps. The columns correspond to the primitive variables ρ , u , v , and p . For each primitive variable, the four panels in the corresponding column use the same colorbar range, allowing direct comparison across methods. LGNO better preserves the main wave structures and fine patterns, while the FNO baseline develops stronger distortions and oscillatory artifacts during rollout.

Several directions remain open. On the theoretical side, it would be valuable to analyze the approximation and stability properties of the local–global architecture and to characterize when the learned flow map stays low-dissipative over long horizons. On the practical side, we plan to extend LGNO to more complex systems and higher dimensions, to enforce local conservation and bound-preserving structure, to study robustness under extrapolation beyond the training distribution, and to develop efficient training strategies for large-scale problems.

6. Reproducibility

To support reproducibility, we release our implementation of LGNO, the training and evaluation scripts, the WENO-Z data-generation code, and the trained model weights for all benchmark experiments reported in this paper. The code and pretrained checkpoints are available at <https://github.com/shanxue-w/ConservationLaws>. The repository includes the configuration files needed to reproduce the reported autoregressive rollouts and figures.

Acknowledgment

This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231 using NERSC award ASCR-ERCAP0037224.

Appendix A. Generalization Test: Two-Dimensional Euler Shock Tests

To further study the robustness of the learned flow map for the two-dimensional Euler equations, we include four shock tests based on standard quadrant Riemann benchmarks [30]. *These tests do not appear in the training set and thus probe generalization to unseen initial data; since the states remain within the training distribution, this is an interpolation rather than an extrapolation test.* They assess whether the learned rollouts reproduce the main shock and contact structures, especially their locations, under strong wave interactions.

Test setup. The initial data are prescribed by piecewise constant quadrant states in primitive variables. We divide the domain into four regions,

$$\begin{aligned}\mathcal{R}_1 &= \{x \geq 1/2, y \geq 1/2\}, & \mathcal{R}_2 &= \{x < 1/2, y \geq 1/2\}, \\ \mathcal{R}_3 &= \{x < 1/2, y < 1/2\}, & \mathcal{R}_4 &= \{x \geq 1/2, y < 1/2\}.\end{aligned}$$

On each region \mathcal{R}_i , the primitive variables are set to a constant state,

$$(p, \rho, u, v)(x, y) = (p_i, \rho_i, u_i, v_i), \quad (x, y) \in \mathcal{R}_i, \quad i = 1, 2, 3, 4.$$

We label the four cases S1–S4 and run each under both outflow and periodic boundary conditions. With outflow conditions, the tests are the usual quadrant shock-interaction setup, evolved to $T = 0.25$ so the main waves stay away from the boundary. With periodic conditions, the same quadrant data are periodized; we therefore use a longer rollout to $T = 0.5$ to further stress the learned periodic flow map under strong discontinuities and interacting waves. The periodic case is a stress test of the learned flow map, not a canonical self-similar Riemann problem.

Shock-state definitions. The four shock tests are specified below. In each display, the small 2×2 block indicates the spatial arrangement of the four regions: the upper row corresponds to $y \geq 1/2$, and the lower row corresponds to $y < 1/2$. The primitive states are listed in the order (p, ρ, u, v) .

SHOCK TEST S1. $\begin{array}{c|c} \mathcal{R}_2 & \mathcal{R}_1 \\ \hline \mathcal{R}_3 & \mathcal{R}_4 \end{array}$: the initial data are

$$\begin{array}{cc|cc} p_2 = 0.4 & \rho_2 = 0.5197 & p_1 = 1 & \rho_1 = 1 \\ u_2 = -0.7259 & v_2 = 0 & u_1 = 0 & v_1 = 0 \\ \hline p_3 = 1 & \rho_3 = 1 & p_4 = 0.4 & \rho_4 = 0.5197 \\ u_3 = -0.7259 & v_3 = -0.7259 & u_4 = 0 & v_4 = -0.7259 \end{array}$$

SHOCK TEST S2. $\frac{\mathcal{R}_2}{\mathcal{R}_3} \mid \frac{\mathcal{R}_1}{\mathcal{R}_4}$: the initial data are

$$\begin{array}{llll} p_2 = 0.4 & \rho_2 = 0.5197 & p_1 = 1 & \rho_1 = 1 \\ u_2 = -0.6259 & v_2 = 0.1 & u_1 = 0.1 & v_1 = 0.1 \\ p_3 = 0.4 & \rho_3 = 0.8 & p_4 = 0.4 & \rho_4 = 0.5197 \\ u_3 = 0.1 & v_3 = 0.1 & u_4 = 0.1 & v_4 = -0.6259 \end{array}$$

SHOCK TEST S3. $\frac{\mathcal{R}_2}{\mathcal{R}_3} \mid \frac{\mathcal{R}_1}{\mathcal{R}_4}$: the initial data are

$$\begin{array}{llll} p_2 = 1 & \rho_2 = 1 & p_1 = 0.4 & \rho_1 = 0.5197 \\ u_2 = -0.6259 & v_2 = 0.1 & u_1 = 0.1 & v_1 = 0.1 \\ p_3 = 1 & \rho_3 = 0.8 & p_4 = 1 & \rho_4 = 1 \\ u_3 = 0.1 & v_3 = 0.1 & u_4 = 0.1 & v_4 = -0.6259 \end{array}$$

SHOCK TEST S4. $\frac{\mathcal{R}_2}{\mathcal{R}_3} \mid \frac{\mathcal{R}_1}{\mathcal{R}_4}$: the initial data are

$$\begin{array}{llll} p_2 = 1 & \rho_2 = 1 & p_1 = 0.4 & \rho_1 = 0.5313 \\ u_2 = 0.7276 & v_2 = 0 & u_1 = 0 & v_1 = 0 \\ p_3 = 1 & \rho_3 = 0.8 & p_4 = 1 & \rho_4 = 1 \\ u_3 = 0 & v_3 = 0 & u_4 = 0 & v_4 = 0.7276 \end{array}$$

Outflow boundary condition. We first evaluate the four tests with outflow boundary conditions, rolling out the learned models autoregressively to $T = 0.25$ and comparing against WENO-Z reference solutions. We report density contours, which emphasize shock and contact locations, together with density fields, which give a complementary view of the global distribution.

Figure A.15 compares the density contours, a direct check of shock and contact locations, since a learned flow map that does not explicitly enforce local conservation may in principle produce incorrect shock speeds. Across S1–S4, LGNO has a comparable contour plot as the WENO-Z reference, including the dominant shock and contact locations, with discrepancies confined to strong wave-interaction regions where minor contour distortion and mild smoothing appear. The FNO baseline, in contrast, develops substantial spurious oscillations and distorted wave patterns, especially near the central interaction zones, so LGNO avoids the shock-location errors seen in FNO.

Figure A.16 shows the corresponding density fields. LGNO stays close to the reference in the large-scale distribution and retains the main high- and low-density regions across all four tests, whereas the FNO baseline shows clear unphysical artifacts: localized overshoots, oscillatory structures, and large-scale distortion. LGNO is thus markedly more robust than FNO for shock-dominated outflow rollouts of the two-dimensional Euler equations.

Symmetry-consistent scaling. The compressible Euler equations admit a two-parameter scaling invariance. We use this invariance to align the prescribed shock states with the magnitudes seen during training, while preserving the fact that the rescaled variables still solve the same Euler system. Specifically, for any constants $s > 0$ and $k > 0$, suppose that $(\rho, u, v, p)(\mathbf{x}, t)$ solves the system (21)–(22). Define the rescaled variables

$$\tilde{\rho}(\mathbf{x}, t) = s\rho(\mathbf{x}, t/k), \quad \tilde{u}(\mathbf{x}, t) = \frac{1}{k}u(\mathbf{x}, t/k), \quad \tilde{v}(\mathbf{x}, t) = \frac{1}{k}v(\mathbf{x}, t/k), \quad \tilde{p}(\mathbf{x}, t) = \frac{s}{k^2}p(\mathbf{x}, t/k).$$

Then $(\tilde{\rho}, \tilde{u}, \tilde{v}, \tilde{p})$ also satisfies the same two-dimensional compressible Euler equations on the same spatial domain. Indeed, with $\tau = t/k$, the continuity equation is multiplied by the common factor s/k . The two momentum equations are multiplied by the common factor s/k^2 , since

$$\tilde{\rho}\tilde{u} = \frac{s}{k}\rho u, \quad \tilde{\rho}\tilde{v} = \frac{s}{k}\rho v,$$

and

$$\tilde{\rho}\tilde{u}^2 + \tilde{p} = \frac{s}{k^2}(\rho u^2 + p), \quad \tilde{\rho}\tilde{u}\tilde{v} = \frac{s}{k^2}\rho uv, \quad \tilde{\rho}\tilde{v}^2 + \tilde{p} = \frac{s}{k^2}(\rho v^2 + p).$$

Similarly, the total energy rescales as

$$\tilde{E} = \frac{\tilde{p}}{\gamma - 1} + \frac{1}{2}\tilde{\rho}(\tilde{u}^2 + \tilde{v}^2) = \frac{s}{k^2}E,$$

so the energy equation is multiplied by the common factor s/k^3 . Therefore all rescaled conservation laws reduce to the original Euler equations evaluated at (\mathbf{x}, τ) , and hence the rescaled variables remain an exact solution of the Euler system.

Under this scaling, the factor s rescales density and pressure together, while the factor k divides the velocities and the sound speed $c = \sqrt{\gamma p/\rho}$ by k . Consequently, the Mach number $\sqrt{u^2 + v^2}/c$ and the wave structure are preserved, while the flow evolves k times more slowly in time. This allows the prescribed states to be rescaled to the training range without changing the underlying Euler dynamics.

We use this invariance in the outflow tests at magnitudes compatible with the learned primitive-variable models. Density and pressure are first multiplied by a common positive factor $s_{\rho p}$,

$$(\rho, u, v, p) \mapsto (s_{\rho p}\rho, u, v, s_{\rho p}p),$$

and, for the learned models, we additionally introduce a positive velocity scale k and evaluate on

$$(s_{\rho p}\rho, u, v, s_{\rho p}p) \mapsto \left(s_{\rho p}\rho, \frac{u}{k}, \frac{v}{k}, \frac{s_{\rho p}p}{k^2} \right).$$

At the continuous level, this scaling is valid for any $k > 0$: the rescaled flow evolves more slowly by the time factor k . In the discrete rollout experiments, we choose k to be an integer so that the corresponding time-rescaled trajectory can be evaluated by taking k times as many internal time steps with the same step size. The final prediction is then mapped back to the original primitive variables before visualization. This brings the prescribed states into the range seen during data generation while, by the invariance above, leaving the underlying Euler dynamics unchanged, so it does not impact the qualitative behavior of the prediction.

No such scaling is used in the periodic tests, where the models are evaluated directly on the original variables; the periodic results therefore give a cleaner assessment of generalization to these unseen configurations.

Periodic boundary condition. We next repeat the four cases with periodic boundary conditions. Periodic extension of the quadrant states introduces additional discontinuities across the domain boundaries, making these rollouts especially challenging; with no boundary to avoid, we evolve them to the longer final time $T = 0.5$ to make it even more challenging.

Figure A.17 compares the density contours. The longer horizon produces more complex wave interactions than the outflow tests, yet LGNO still captures the dominant contour geometry of the reference, including the principal shock and contact structures; local distortion and fine-scale artifacts appear near strong interaction regions, but the main patterns remain recognizable. The FNO baseline produces much denser spurious contours and loses much of the reference shock geometry.

Figure A.18 shows the corresponding density fields. LGNO retains the large-scale density organization across S1–S4, with only localized deviations after the longer rollout, while the FNO baseline shows stronger oscillations, distorted density regions, and a more severe loss of coherent structure. Consistent with the low-dissipation behavior reported in Section 4.6, along several contact discontinuities and shear layers LGNO develops finer roll-up and vortical structures than the WENO-Z reference on the same mesh; as there, this should be read not as LGNO being more accurate than the reference but as its lower long-time numerical dissipation preserving sharper localized features. These tests reinforce that LGNO is more stable than FNO and, even on these unseen shock-interaction configurations, less dissipative than the WENO-Z scheme that generated its training data.

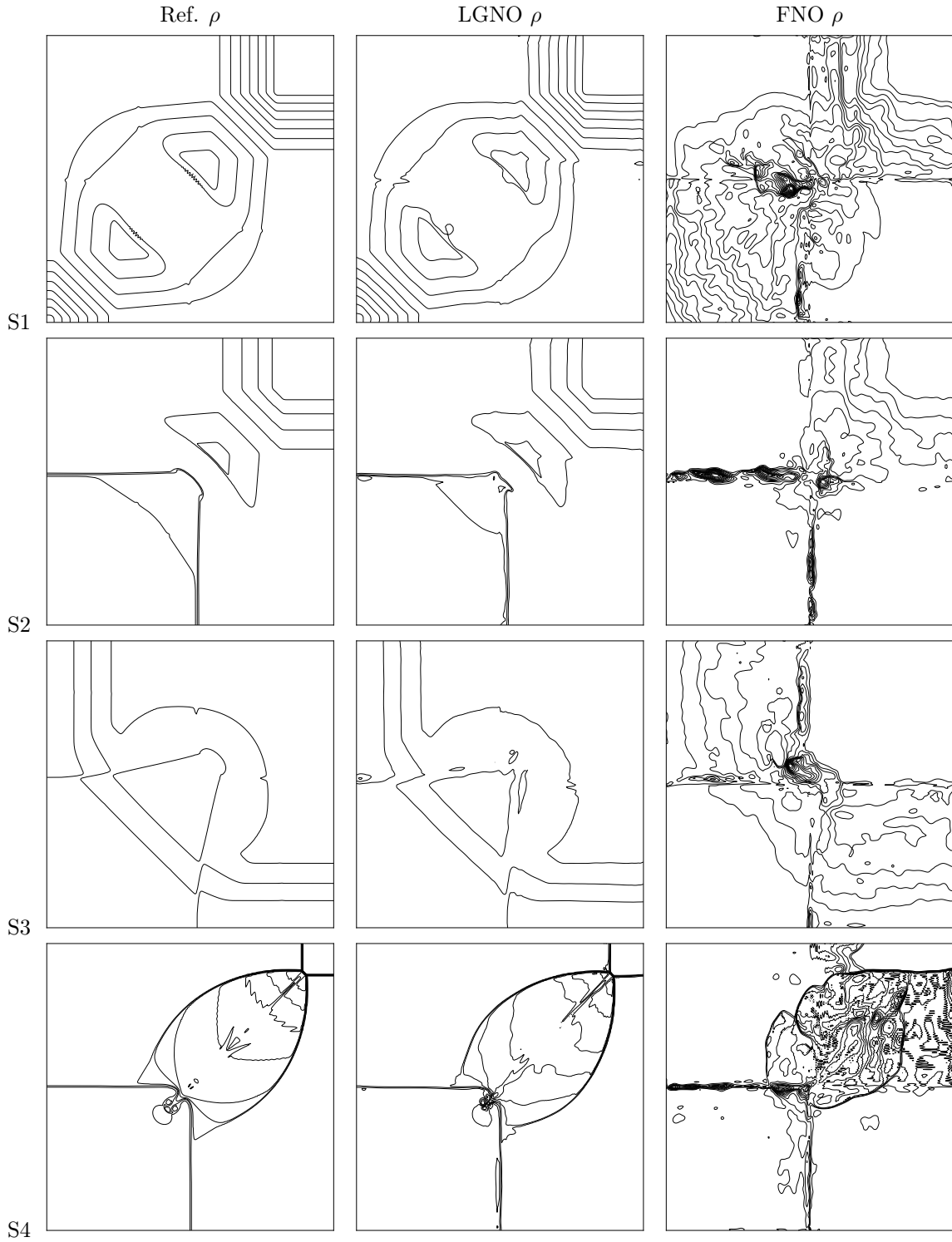


Figure A.15: **Density contours at $T = 0.25$ for the four outflow shock tests.** Rows S1–S4 are the four standard quadrant shock-interaction configurations, and columns show the WENO-Z reference, LGNO, and the FNO baseline. The contours give a direct check of shock and contact locations, which is relevant for assessing possible shock-speed errors in learned rollouts: LGNO closely matches the reference shock locations, whereas the FNO baseline develops spurious contour oscillations and distorted patterns.

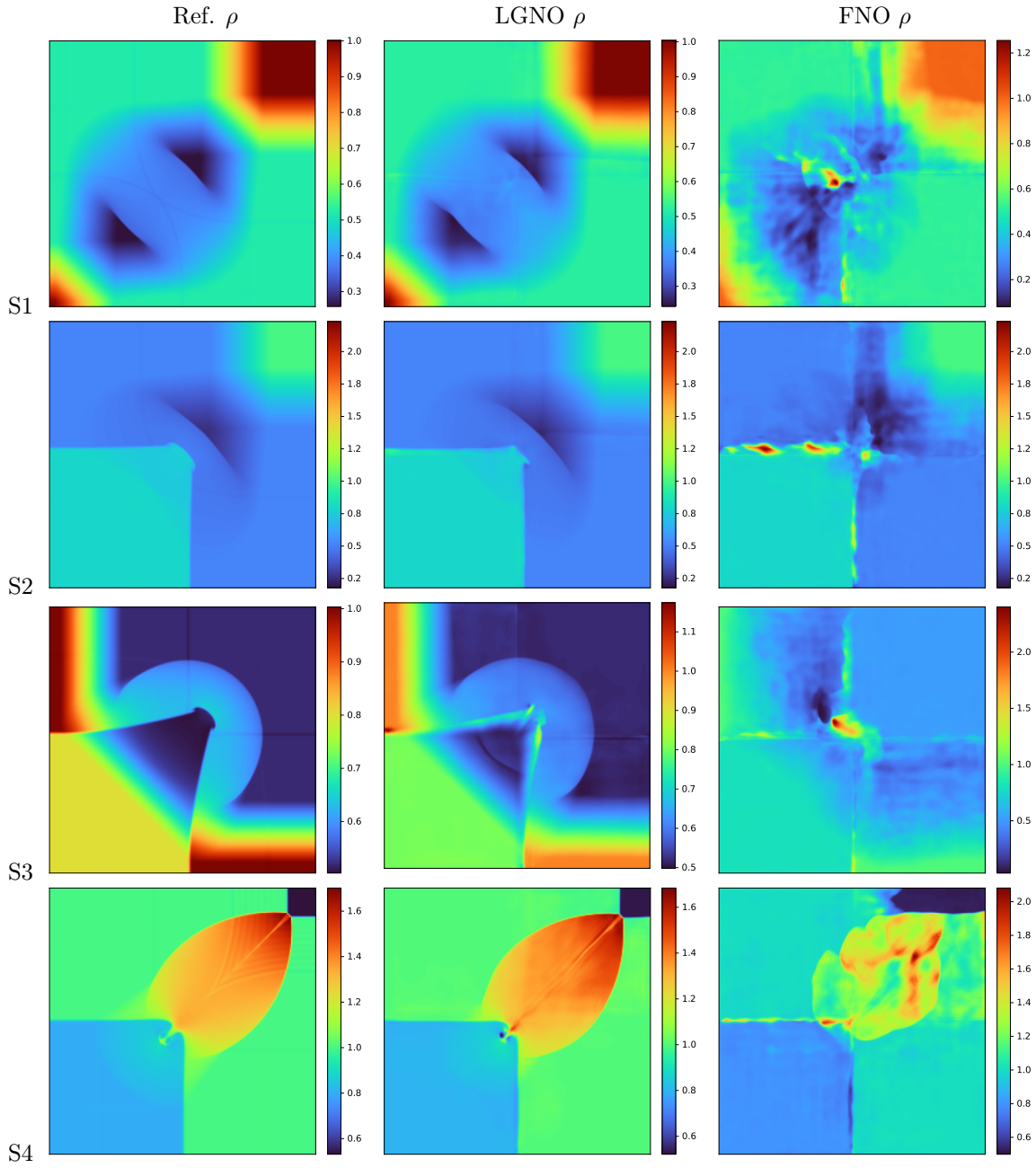


Figure A.16: **Density fields at $T = 0.25$ for the four outflow shock tests.** Rows are S1–S4 and columns show the WENO-Z reference, LGNO, and the FNO baseline. LGNO remains close to the reference, while the FNO baseline exhibits visible unphysical artifacts, including localized overshoots, oscillatory structures, and large-scale distortions of the density field.

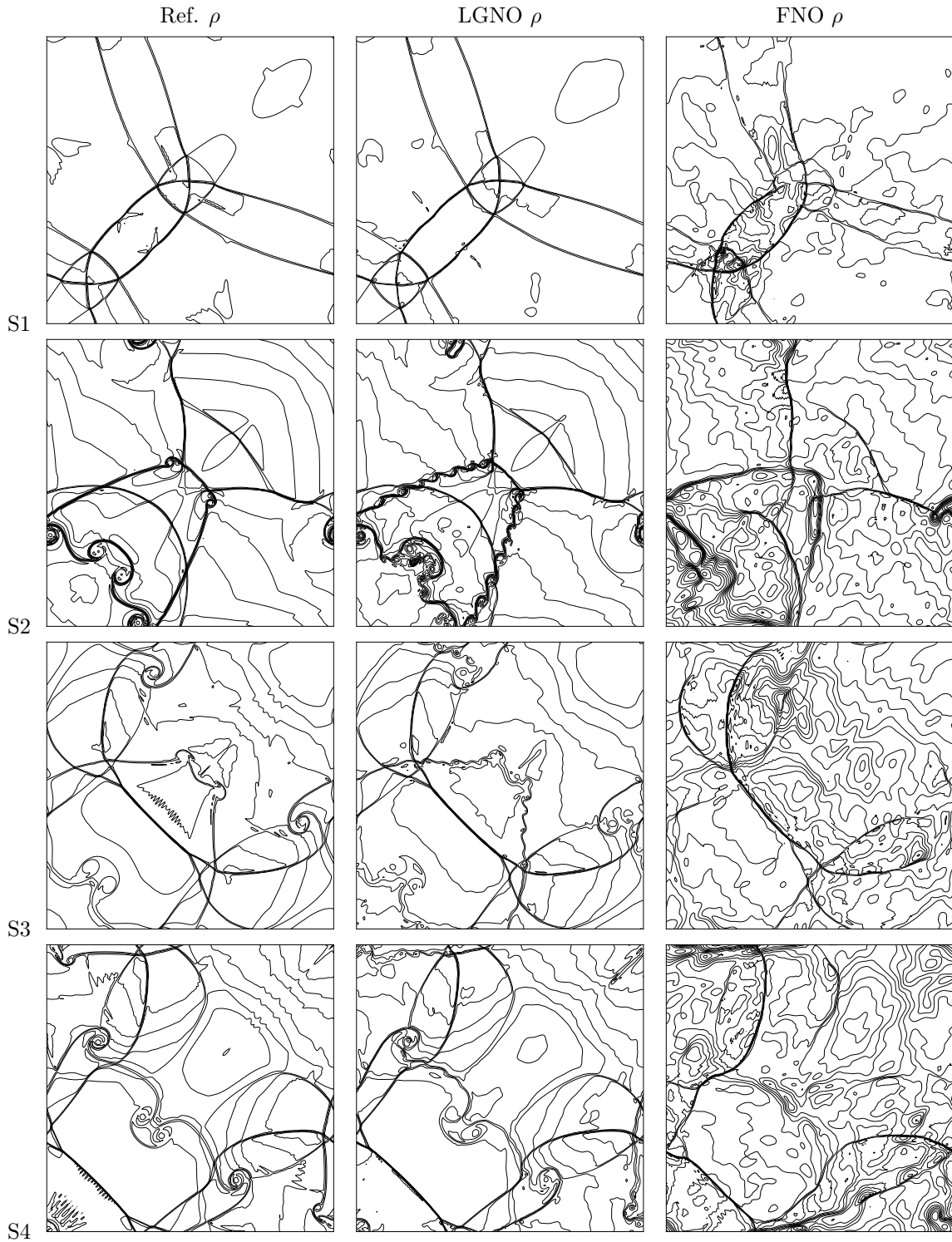


Figure A.17: **Density contours at $T = 0.5$ for the four periodic shock tests.** Rows S1–S4 are the four periodized quadrant configurations, and columns show the WENO-Z reference, LGNO, and the FNO baseline. After the longer rollout, LGNO retains the dominant shock and contact geometry, whereas the FNO baseline produces much denser spurious contours. Notably, along contacts and shear layers LGNO is visibly less dissipative than the WENO-Z reference, resolving finer roll-up and vortical structures.

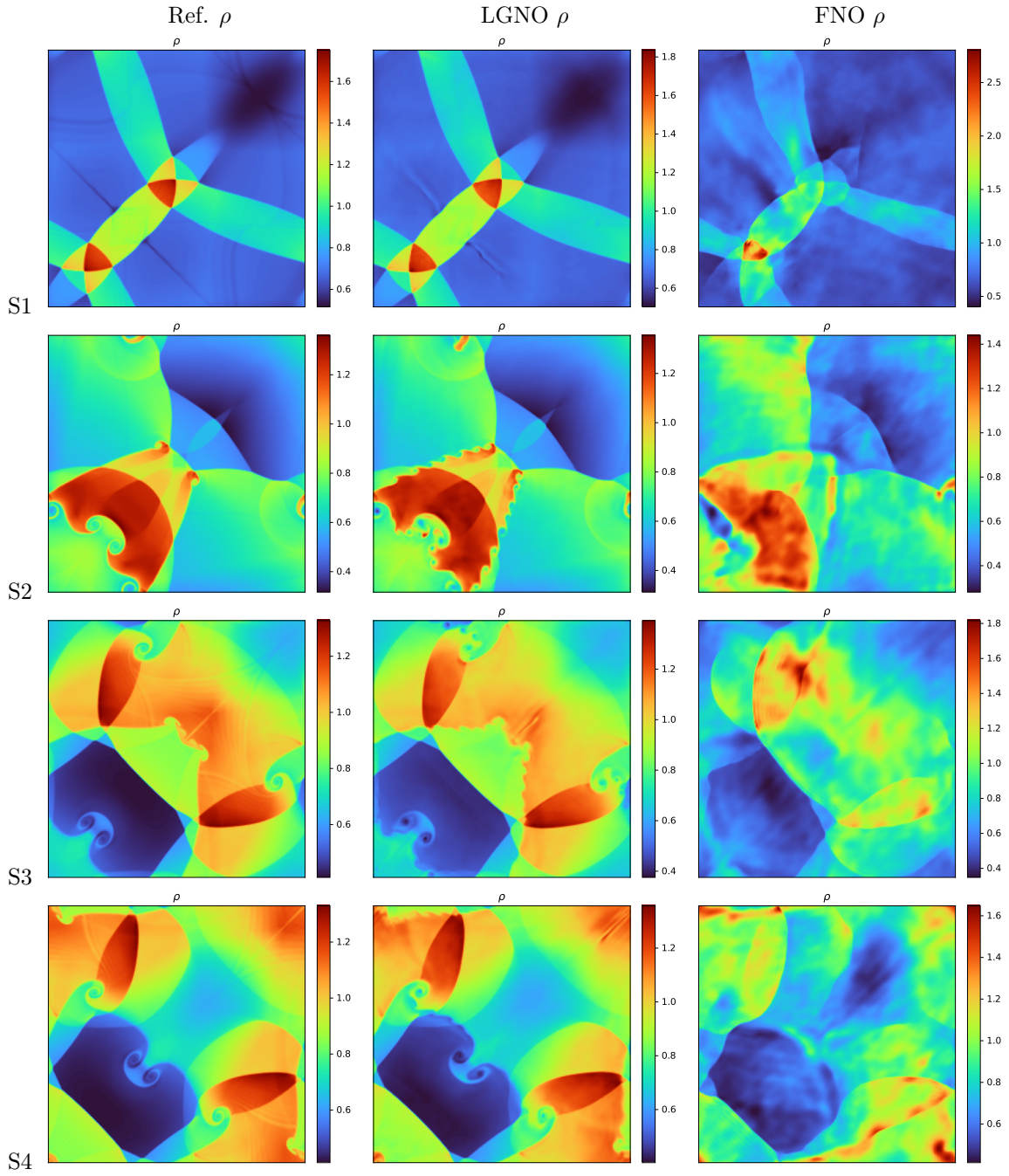


Figure A.18: **Density fields at $T = 0.5$ for the four periodic shock tests.** Rows are S1–S4 and columns show the WENO-Z reference, LGNO, and the FNO baseline. The panels use individual color scales, so the comparison focuses on the geometry of the density field and the presence of unphysical oscillations rather than on direct amplitude matching. Along several contacts and shear layers, LGNO resolves finer roll-up and vortical structures than the WENO-Z reference, reflecting its lower long-time numerical dissipation.

Appendix B. Implementation Details and Computational Cost

Appendix B.1. Model and Training Configuration

The neural operators are parameterized in a residual form,

$$u^{n+1} = u^n + \Delta t \Phi_\theta(u^n),$$

so that the network represents the discrete increment over one time step. In the periodic case, the predicted increment is projected onto the zero-mean subspace before the update for global conservations.

Unless otherwise specified, all models use width 64 and four operator blocks. Fixed coordinate channels are appended to the physical input fields. For periodic problems, the coordinates are represented by periodic features, e.g., $\sin(2\pi x)$ and $\cos(2\pi x)$ in one space dimension, with the analogous features used for each coordinate direction in two space dimensions. For non-periodic boundary conditions, the raw spatial coordinates are used directly. The retained Fourier modes and the corresponding numbers of trainable parameters are summarized in Table B.2. The benchmark-specific numerical setups, including the reference solvers, initial data, and snapshot spacing, are given in Appendix B.3.

Table B.2: Model sizes, retained Fourier modes, and local convolution kernel sizes used in the numerical comparisons.

Problem	LGNO modes	FNO modes	LGNO kernel	LGNO params	FNO params
Linear advection	16	24	5	841,729	811,970
Burgers	16	24	5	841,729	811,777
SWE	16	24	5	841,858	811,970
Euler	16	24	5	841,987	812,163
Burgers2D	16×16	24×24	5	17,750,017	37,774,209
Euler2D periodic	24×24	24×24	5	39,508,356	37,774,788
Euler2D outflow (primitive)	24×24	24×24	7	41,744,772	37,774,660

Unless otherwise noted, the LGNO and FNO models are trained with the AdamW optimizer [31] using an initial learning rate of 10^{-3} , weight decay 10^{-4} , and a cosine annealing learning-rate schedule [32]; benchmark-specific deviations are noted in Appendix B.3. All experiments are run on NVIDIA A100 GPUs on the NERSC Perlmutter system.

Appendix B.2. Computational Cost Comparison

We compare the wall-clock cost of WENO-Z, LGNO, and the FNO baseline on the same evaluation meshes, to give a rough sense of inference cost rather than a careful benchmark. The neural models use the same configurations as in the main text and Table B.2. Timings are measured on a single core of an AMD EPYC 7713 CPU for WENO-Z and on an NVIDIA A100 GPU for the neural models. The WENO-Z solver is a vectorized NumPy implementation; it is not performance-optimized (no multithreading or compiled kernels) and uses a component-wise reconstruction without characteristic decomposition. A characteristic-wise WENO-Z, as commonly used for nonlinear systems, would be a few times more expensive, whereas a compiled or parallel implementation would be substantially faster; the reported WENO-Z times therefore do not represent an optimized solver.

For each problem, we advance one representative test sample to the same final time: WENO-Z uses CFL-limited explicit time stepping, while the learned models are applied autoregressively with the fixed learned time step. The evaluation settings are summarized in Table B.3. WENO-Z is timed on the same spatial mesh as the learned models.

Table B.4 reports the corresponding wall-clock times and speedups. Both learned models advance the solution faster than this WENO-Z implementation on the same mesh, with the gap growing from modest factors for the inexpensive one-dimensional scalar problems to one to three orders of magnitude for the two-dimensional Euler tests. Several factors contribute to these numbers and pull in different directions: the learned models run on a GPU while WENO-Z runs on a single CPU core; for the two Euler2D cases the neural models are evaluated in single precision with TF32 tensor-core acceleration, which raises A100 throughput by roughly an order of magnitude relative to double precision, whereas WENO-Z remains in double precision; and the WENO-Z baseline is unoptimized and component-wise, as noted above. The

reported speedups should therefore be read as practical, order-of-magnitude guidance on deployment cost, not as a controlled comparison between two optimized codes, since neither code was tuned for performance. A careful, hardware- and precision-normalized comparison is left to future work.

The FNO baseline is generally faster than LGNO in raw inference time, since LGNO adds local operations to the global spectral branch. As shown in the preceding experiments, this additional cost buys improved stability and accuracy in shock-dominated rollouts. Both learned models remain considerably cheaper to evaluate than the present WENO-Z implementation.

Table B.3: Evaluation settings used in the computational cost comparison. The learned models are evaluated autoregressively with the fixed time step Δt shown below. WENO-Z is advanced to the same final time using CFL-limited explicit time stepping with the listed CFL number.

Problem	Mesh	Boundary	Learned Δt	Learned steps	Final time	WENO CFL
Linear advection	256	periodic	0.05	60	3.0	0.4
Burgers	256	periodic	0.05	80	4.0	0.4
Shallow water	256	periodic	0.05	60	3.0	0.4
Euler	256	periodic	0.05	20	1.0	0.4
Burgers2D	128 ²	periodic	0.05	60	3.0	0.4
Euler2D	256 ²	outflow	0.01	40	0.4	0.45
Euler2D	256 ²	periodic	0.01	50	0.5	0.45

Table B.4: Wall-clock times for advancing one test sample to the specified final time using WENO-Z, LGNO, and the FNO baseline on the same evaluation meshes. WENO-Z runs as a single-core, unoptimized, component-wise NumPy implementation (no characteristic decomposition) in double precision, while the neural models run on an A100 GPU in double precision, except for the two Euler2D tests, which use single precision with TF32. Speedups are relative to this WENO-Z implementation on the same mesh and time interval, computed from the unrounded timings. Because neither code is performance-optimized and the settings differ in hardware and precision, these numbers should be read as general guidance on deployment cost rather than a careful benchmark.

Problem	WENO-Z (s)	LGNO (s)	FNO (s)	LGNO speedup	FNO speedup
Linear advection	0.927	0.317	0.179	2.9×	5.2×
Burgers	0.661	0.471	0.267	1.4×	2.5×
Shallow water	9.545	0.325	0.180	29.4×	53.0×
Euler	3.862	0.115	0.066	33.7×	58.8×
Burgers2D	4.021	0.471	0.278	8.5×	14.4×
Euler2D outflow	511.263	0.698	0.244	732.8×	2098.0×
Euler2D periodic	809.503	0.484	0.310	1672.5×	2611.3×

Appendix B.3. Numerical Setups

The training datasets are generated from reference trajectories computed on fine grids (e.g., 512²) and then transferred to the learning grids (e.g., 256²) by conservative averaging. All benchmarks use periodic boundary conditions except the two-dimensional Euler outflow case.

Appendix B.3.1. Linear Advection Equation

For the linear advection equation (16), the training data consist of 10,000 input-target pairs on a 256-cell grid ($\Delta t = 0.05$), generated from 1,000 trajectories using 10 consecutive transitions each; batch size 64.

The initial condition is sampled from a mixture of discontinuous and smooth periodic profiles, with 60% discontinuous and 40% smooth. The discontinuous component is piecewise constant,

$$u_0(x) = \sum_{j=1}^m c_j \mathbf{1}_{I_j}(x),$$

where $m \in \{2, 3\}$. The periodic partition $\{I_j\}_{j=1}^m$ is determined by one or two random breakpoints, with a minimum cyclic separation of 0.15, and the segment values c_j are sampled independently from the uniform

distribution on $[-0.8, 0.8]$. The smooth component is a Fourier series,

$$\tilde{u}_0(x) = \sum_{k=1}^K (a_k \cos(2\pi kx) + b_k \sin(2\pi kx)), \quad K \in \{2, 3\},$$

with raw coefficients decaying as k^{-2} ; each profile is then normalized by its maximum absolute value and multiplied by an amplitude sampled uniformly from $[0.5, 1.0]$.

Appendix B.3.2. One-Dimensional Burgers Equation

For the Burgers equation (17), the training data consist of 10,000 input-target pairs on a 256-cell grid ($\Delta t = 0.05$), generated from 500 trajectories evolved to $T = 1$ using 20 consecutive transitions each; batch size 64.

The initial condition is generated from a Fourier series,

$$\tilde{u}_0(x) = \sum_{k=1}^5 (a_k \cos(2\pi kx) + b_k \sin(2\pi kx)),$$

where a_k and b_k are independent Gaussian random variables with mean zero and variances proportional to k^{-4} . The profile is then normalized by its maximum absolute value and multiplied by an amplitude A sampled uniformly from $[0.5, 1.0]$. A constant mean shift B , sampled uniformly from $[-\frac{1}{2}A, \frac{1}{2}A]$, is then added. This yields smooth initial data that develop shocks during the evolution.

Appendix B.3.3. One-Dimensional Shallow Water Equations

For the one-dimensional shallow water equations (18), the training data consist of 10,000 input-target pairs on a 256-cell grid ($\Delta t = 0.05$), generated from 500 trajectories evolved to $T = 1$ using 20 consecutive transitions each; batch size 64.

The initial condition is sampled from a mixture of three components. For clarity, we describe the sampling procedure in primitive variables (h, u) . The first component (30%) is a smooth bidirectional wave, obtained by perturbing the Riemann invariants of the shallow water system,

$$w_{\pm}(x) = u(x) \pm 2\sqrt{gh(x)}.$$

Let $(h_0, u_0) = (1, 0)$, $c_0 = \sqrt{gh_0}$, and base invariants $w_{\pm}^{(0)} = u_0 \pm 2c_0$. We then sample two independent smooth Fourier profiles $s_+(x)$ and $s_-(x)$, normalized to have unit maximum absolute value, and define

$$w_{\pm}(x) = w_{\pm}^{(0)} + A s_{\pm}(x), \quad A/c_0 \in [0.08, 0.22].$$

The number of retained Fourier modes is sampled from $\{2, 3, 4\}$, with higher modes damped in the random series. The primitive variables are recovered from the perturbed invariants by

$$h(x) = \frac{1}{g} \left(\frac{w_+(x) - w_-(x)}{4} \right)^2, \quad u(x) = \frac{w_+(x) + w_-(x)}{2}.$$

A mild positivity safeguard is applied to avoid near-dry states.

The second component (40%) is a coupled smooth state of the form

$$h(x) = h_0 + \varepsilon_h s_h(x), \quad u(x) = u_0 + A_u s_u(x),$$

where ε_h is sampled uniformly from $[0.10, 0.22]$ and A_u/c_0 from $[0.15, 0.45]$. The functions s_h and s_u are Fourier series with decay slopes sampled from $[1, 3]$, and the largest retained Fourier mode is sampled between 2 and 4.

The third component (30%) is a periodic smoothed Riemann state,

$$h(x) = h_{\text{out}} + (h_{\text{in}} - h_{\text{out}}) \chi(x), \quad u(x) = u_{\text{out}} + (u_{\text{in}} - u_{\text{out}}) \chi(x),$$

where

$$\chi(x) = \frac{1}{2} \left(1 + \tanh \left(\frac{w - |d(x, x_c)|}{\delta} \right) \right).$$

Here $d(x, x_c)$ is the periodic distance from x to the center x_c . The parameters are sampled uniformly: $h_{\text{in}} \in [1.2, 2.0]$, $h_{\text{out}} \in [0.2, 0.9]$, $u_{\text{in}}, u_{\text{out}} \in [-0.8, 0.8]$, $w \in [0.08, 0.22]$, and $\delta \in [0.01, 0.04]$.

Appendix B.3.4. One-Dimensional Euler Equations

For the one-dimensional Euler equations (19), the training data consist of 10,000 input-target pairs on a 256-cell grid ($\Delta t = 0.05$), generated from 500 trajectories evolved to $T = 1$ using 20 consecutive transitions each; batch size 64.

The initial condition is sampled from a mixture of random two-state Riemann data and smooth perturbations, with 80% Riemann and 20% smooth. The Riemann component has constant left and right states meeting at a discontinuity x_0 sampled from the middle 60% of the domain, with $\rho, p \in [0.05, 1.20]$ and $u \in [-1, 1]$ on each side; the two states must differ by at least 30% in density and pressure and by at least 0.3 in velocity. The smooth component has

$$\rho(x) = \rho_0(1 + \alpha_\rho S_\rho(x)), \quad p(x) = p_0(1 + \alpha_p S_p(x)), \quad u(x) = u_0 + \alpha_u S_u(x),$$

where S_ρ, S_p, S_u are Fourier series with $\|S_\bullet\|_{L^\infty} \leq 1$ and $K \in \{1, 2, 3\}$ modes, base state $\rho_0, p_0 \in [0.30, 1.10]$, $u_0 \in [-1, 1]$, and amplitudes $\alpha_\rho, \alpha_p \in [0.12, 0.65]$, $\alpha_u \in [0.10, 0.70]$.

Appendix B.3.5. Two-Dimensional Burgers Equation

For the two-dimensional Burgers equation (20), the training data consist of 10,000 input-target pairs on a 128×128 grid ($\Delta t = 0.05$), generated from 500 trajectories evolved to $T = 1$ using 20 consecutive transitions each; batch size 16.

The initial condition is generated from a Fourier series,

$$\tilde{u}_0(x, y) = \sum_{k_x=0}^{K_x} \sum_{k_y=0}^{K_y} \hat{u}_{k_x, k_y} \cos(2\pi(k_x x + k_y y) + \phi_{k_x, k_y}),$$

where K_x and K_y are sampled independently from $\{0, 1, 2, 3, 4\}$. For each nonzero mode (k_x, k_y) , the coefficient magnitude is proportional to $(k_x^2 + k_y^2)^{-0.75}$ and the phase ϕ_{k_x, k_y} is sampled uniformly from $[0, 2\pi]$. The resulting field is normalized by its maximum absolute value and multiplied by an amplitude A sampled uniformly from $[0.4, 1.0]$. It is then shifted by a constant B , sampled uniformly from $[-\frac{1}{2}A, \frac{1}{2}A]$.

Appendix B.3.6. Two-Dimensional Euler Equations with Outflow Boundary Conditions

For the two-dimensional Euler equations (21)–(22) with outflow boundary conditions, the training data consist of 10,000 input-target pairs on a 256×256 grid ($\Delta t = 0.01$), generated from 250 trajectories evolved to $T = 0.4$ using 40 consecutive transitions each; the reference solutions are computed on a 512×512 grid and transferred to the 256×256 grid. The models use batch size 16, initial learning rate 5×10^{-4} , and cosine decay over 500 epochs.

The initial condition is a piecewise constant quadrant state,

$$U_0(x, y) = \begin{cases} U_1, & x \geq x_0, y \geq y_0, \\ U_2, & x < x_0, y \geq y_0, \\ U_3, & x \geq x_0, y < y_0, \\ U_4, & x < x_0, y < y_0, \end{cases}$$

where the split point (x_0, y_0) is sampled in the interior of the domain. Each quadrant state is sampled independently in primitive variables (ρ, u, v, p) . The density ρ_j is sampled from the log-uniform distribution on $[0.08, 1.0]$, and the pressure p_j is sampled from the log-uniform distribution on $[0.04, 0.5]$. The velocity components u_j, v_j are sampled uniformly from $[-0.5, 0.5]$, and the state is required to satisfy the Mach bound $\sqrt{u_j^2 + v_j^2} / \sqrt{\gamma p_j / \rho_j} < 3.5$; if violated, the velocity vector is rescaled until the bound holds.

For the outflow experiment, the models are trained in primitive variables (ρ, u, v, p) , in which case we do not enforce global conservation. Instead, we add a softplus activation to guarantee positivity. Given $\mathbf{u}^n = (\rho^n, u^n, v^n, p^n)$, the model first produces an unconstrained prediction

$$\tilde{\mathbf{u}}^{n+1} = \mathbf{u}^n + \Delta t \Phi_\theta(\mathbf{u}^n; \Delta t).$$

A shifted softplus map $\text{softplus}_\beta(z) := \frac{1}{\beta} \log(1 + e^{\beta z})$ is then applied to density and pressure for positivity preservation,

$$\rho^{n+1} = \rho_{\min} + \text{softplus}_\beta(\tilde{\rho}^{n+1} - \rho_{\min}), \quad p^{n+1} = p_{\min} + \text{softplus}_\beta(\tilde{p}^{n+1} - p_{\min}),$$

while the velocities pass through unchanged, $u^{n+1} = \tilde{u}^{n+1}$ and $v^{n+1} = \tilde{v}^{n+1}$. Here $\rho_{\min} = 10^{-6}$, $p_{\min} = 10^{-8}$, and $\beta = 100$. LGNO uses an outflow boundary operator with local kernel size 7.

Appendix B.3.7. Two-Dimensional Euler Equations with Periodic Boundary Conditions

For the periodic case, the training data consist of 10,000 input-target pairs on a 256×256 grid ($\Delta t = 0.01$), generated from 200 trajectories evolved to $T = 0.5$ using 50 consecutive transitions each; reference solutions are computed on a 512×512 grid and transferred to the 256×256 grid. Batch size 16.

The initial condition is sampled from a mixture of discontinuous quadrant states and smooth periodic states, with 80% discontinuous and 20% smooth. The discontinuous component uses the same piecewise constant quadrant form as in the outflow case, now with ρ_j, p_j log-uniform on $[0.08, 1.0]$, u_j, v_j uniform on $[-1, 1]$, and the same Mach bound $\sqrt{u_j^2 + v_j^2} / \sqrt{\gamma p_j / \rho_j} < 3.5$ (rescaling the velocity if violated).

The smooth component has

$$\log \rho = \log \bar{\rho} + S_\rho, \quad \log p = \log \bar{p} + S_p, \quad u = \bar{u} + S_u, \quad v = \bar{v} + S_v,$$

where each $S_\bullet(x, y)$ is a low-frequency Fourier sum with 2, 3, or 4 distinct modes selected from $\{0, 1, 2, 3\}^2 \setminus \{(0, 0)\}$. The amplitudes of S_ρ and S_p are sampled uniformly from $[0.08, 0.22]$, while the amplitudes of S_u and S_v are sampled uniformly from $[0.05, 0.35]$. If the resulting state exceeds Mach number 3.5, the velocity field is uniformly rescaled to satisfy the same admissibility threshold.

References

- [1] G.-Q. Chen, D. Wang, The Cauchy problem for the Euler equations for compressible fluids, in: Handbook of Mathematical Fluid Dynamics, Vol. 1, Elsevier, 2002, pp. 421–543.
- [2] G. K. Batchelor, An introduction to fluid dynamics, Cambridge University Press, 2000.
- [3] A. Dedner, F. Kemm, D. Kröner, C.-D. Munz, T. Schnitzer, M. Wesenberg, Hyperbolic divergence cleaning for the MHD equations, Journal of Computational Physics 175 (2) (2002) 645–673.
- [4] A. J. Christlieb, J. A. Rossmann, Q. Tang, Finite difference weighted essentially non-oscillatory schemes with constrained transport for ideal magnetohydrodynamics, Journal of Computational Physics 268 (2014) 302–325.
- [5] R. J. LeVeque, Numerical Methods for Conservation Laws, Birkhäuser, 1992.
- [6] R. J. LeVeque, Finite Volume Methods for Hyperbolic Problems, Cambridge University Press, 2002.
- [7] C. M. Dafermos, Hyperbolic conservation laws in continuum physics, 3rd Edition, Vol. 325 of Grundlehren der mathematischen Wissenschaften, Springer, 2005.
- [8] J. S. Hesthaven, T. Warburton, Nodal discontinuous Galerkin methods: algorithms, analysis, and applications, Vol. 54 of Texts in Applied Mathematics, Springer, 2008.
- [9] M. Peng, Z. Sun, K. Wu, OEDG: Oscillation-eliminating discontinuous galerkin method for hyperbolic conservation laws, Mathematics of Computation 94 (353) (2025) 1147–1198.
- [10] G.-S. Jiang, C.-W. Shu, Efficient implementation of weighted ENO schemes, Journal of Computational Physics 126 (1) (1996) 202–228.
- [11] R. Borges, M. Carmona, B. Costa, W. S. Don, An improved weighted essentially non-oscillatory scheme for hyperbolic conservation laws, Journal of Computational Physics 227 (6) (2008) 3191–3211.

- [12] L. Fu, X. Y. Hu, N. A. Adams, A family of high-order targeted ENO schemes for compressible-fluid simulations, *Journal of Computational Physics* 305 (2016) 333–359.
- [13] K. W. Morton, T. Sonar, Finite volume methods for hyperbolic conservation laws, *Acta Numerica* 16 (2007) 155–238.
- [14] C.-W. Shu, High order WENO and DG methods for time-dependent convection-dominated PDEs: A brief survey of several recent developments, *Journal of Computational Physics* 316 (2016) 598–613.
- [15] C.-W. Shu, Essentially non-oscillatory and weighted essentially non-oscillatory schemes, *Acta Numerica* 29 (2020) 701–762.
- [16] A. Zhou, A. Barati Farimani, Predicting change, not states: An alternate framework for neural PDE surrogates, *Computer Methods in Applied Mechanics and Engineering* 441 (2025) 117990.
- [17] Z. Li, N. B. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, A. Anandkumar, Fourier neural operator for parametric partial differential equations, in: *International Conference on Learning Representations*, 2021.
- [18] L. Lu, P. Jin, G. Pang, Z. Zhang, G. E. Karniadakis, Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators, *Nature Machine Intelligence* 3 (3) (2021) 218–229.
- [19] Q. Li, T. Lin, Z. Shen, Deep learning via dynamical systems: An approximation perspective, *Journal of the European Mathematical Society* 25 (5) (2022) 1671–1709.
- [20] N. Kovachki, Z. Li, B. Liu, K. Azizzadenesheli, K. Bhattacharya, A. Stuart, A. Anandkumar, Neural operator: Learning maps between function spaces with applications to PDEs, *Journal of Machine Learning Research* 24 (89) (2023) 1–97.
- [21] S. Lanthaler, A. M. Stuart, The parametric complexity of operator learning, *IMA Journal of Numerical Analysis* 46 (2) (2026) 647–712.
- [22] P. Chauhan, S. E. Choutri, M. Ghattassi, N. Masmoudi, S. E. Jabari, Neural operators struggle to learn complex PDEs in pedestrian mobility: Hughes model case study, *Artificial Intelligence for Transportation* 1 (2025) 100005.
- [23] S. Lanthaler, R. Molinaro, P. Hadorn, S. Mishra, Nonlinear reconstruction for operator learning of PDEs with discontinuities, in: *The Eleventh International Conference on Learning Representations*, 2023.
- [24] A. Peyvan, V. Oommen, A. D. Jagtap, G. E. Karniadakis, Riemannonets: Interpretable neural operators for riemann problems, *Computer Methods in Applied Mechanics and Engineering* 426 (2024) 116996.
- [25] T. Kim, M. Kang, Approximating numerical fluxes using Fourier Neural Operators for hyperbolic conservation laws, *Communications in Computational Physics* 37 (2) (2025) 420–456.
- [26] N. Lichtlé, A. Canesse, Z. Fu, H. N. Z. Matin, M. L. D. Monache, A. M. Bayen, (u)NFV: (un)supervised neural finite volume methods for solving hyperbolic PDEs, in: *The Fourteenth International Conference on Learning Representations*, 2026.
- [27] J. Shen, T. Tang, L.-L. Wang, *Spectral methods: algorithms, analysis and applications*, Vol. 41 of *Springer Series in Computational Mathematics*, Springer, 2011.
- [28] D. Gottlieb, C.-W. Shu, On the Gibbs phenomenon and its resolution, *SIAM Review* 39 (4) (1997) 644–668.
- [29] O. Ronneberger, P. Fischer, T. Brox, U-Net: Convolutional networks for biomedical image segmentation, in: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, Springer, 2015, pp. 234–241.
- [30] P. D. Lax, X.-D. Liu, Solution of two-dimensional riemann problems of gas dynamics by positive schemes, *SIAM Journal on Scientific Computing* 19 (2) (1998) 319–340.

- [31] I. Loshchilov, F. Hutter, Decoupled weight decay regularization, in: International Conference on Learning Representations, 2019.
- [32] I. Loshchilov, F. Hutter, SGDR: Stochastic gradient descent with warm restarts, in: International Conference on Learning Representations, 2017.