TRAJECTORYOP: EXPLICIT-INTERFACE OPERATOR LEARNING FOR MOVING SHOCKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural operators provide rapid predictions for families of PDE solutions, but moving shocks remain difficult to represent and learn from limited supervised numerical data. We introduce TRAJECTORYOP, an explicit-interface operator for conservation laws with a single tracked shock. TrajectoryOp consists of an FNO encoder, two state decoders, and a trajectory network. The FNO encoder extracts features from the discrete initial field, the two state decoders represent the primitive states on the two sides of the shock, and the trajectory network predicts the shock trajectory from the initial shock location. The solution is represented by the two regional states separated by the predicted shock interface. Regional PDE residuals constrain the states inside the two predicted regions, and the Rankine–Hugoniot condition couples these states to the shock motion along the predicted interface over the full time interval, including where numerical state labels are unavailable, thereby reducing the required amount of supervised numerical data. Compared with fully-supervised operator learning methods, proposed TRAJECTORYOP achieves almost performance with only about 3% supervised information, demonstrating significant advantages under such sparse supervised case.

1 INTRODUCTION

Neural operators have recently emerged as a promising paradigm for fast solution of parametric partial differential equations (PDEs) (Kovachki et al., 2023). A neural operator learns a mapping from input functions to solution functions, so that a single offline training stage suffices to predict unseen instances rapidly, without a new numerical solve for each initial condition or system parameter. Many-query tasks benefit from this substitution, including parameter studies (Benner et al., 2015), uncertainty quantification (Zhu & Zabaras, 2018), inverse problems (Arridge et al., 2019), and real-time prediction (Kurth et al., 2023), since repeated numerical PDE solves can become computationally demanding (Peherstorfer et al., 2018). Most of these successes concern elliptic and parabolic problems, whose solutions remain smooth, whereas hyperbolic conservation laws pose a different challenge (Lanthaler et al., 2023). They model the transport of conserved quantities in compressible gas dynamics, shallow water, and many other systems (LeVeque, 2002), and a nonlinear conservation law can develop shocks and contact discontinuities even from smooth initial data (Lax, 1973). The central difficulty is that a single global output representation must simultaneously describe the states on both sides of the shock and its input-dependent, time-varying location. A small error in the predicted shock location misaligns the discontinuity across the spatial domain, causing a large solution error even when the two-sided states are accurately approximated.

Operator-learning studies for hyperbolic conservation laws with discontinuous solutions broadly follow two directions. The first changes how the solution is parameterized, so that a jump can be represented at all, through nonlinear reconstruction (Lanthaler et al., 2023), a learned basis (Peyvan et al., 2024), input-dependent coordinate transformations (Zhu et al., 2024), or a lifting of the solution to a piecewise-smooth space (Dang et al., 2026). The second learns the evolution step of a numerical scheme as a network, inheriting the local structure of the scheme (Wang et al., 2026; Zdrale et al., 2026). The shock location remains a free quantity that no relation constrains, so these methods can only learn it from data and therefore depend on extensive evolution-state supervision labels (Figure 1, top).

To reduce the reliance on evolution-state supervision labels, we introduce TRAJECTORYOP, which represents the shock location explicitly and constrains its motion with the jump condition of the conservation law. The solution is represented by two regional state mappings and a trajectory net-

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

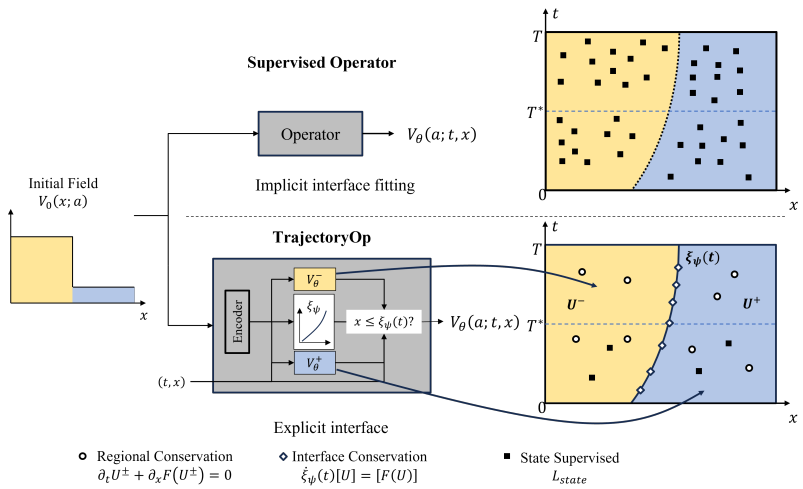


Figure 1: Overview of TRAJECTORYOP. Top: a supervised operator that fits the interface implicitly, with state supervision labels spread over the whole space–time domain. Bottom: TRAJECTORYOP, where the interface is an explicit network output, the two regional branches V_θ^- and V_θ^+ and the trajectory $\xi_\psi(t)$ are combined by the selection $x \le \xi_\psi(t)$ into the piecewise solution $V_\theta(a;t,x)$ on the right, and regional (circles) and interface (diamonds) conservation residuals replace most of the state supervision labels (squares).

work initialized at the shock location estimated from the initial field (Figure 1, bottom). The regional mappings make the strong-form PDE residual applicable inside each smooth region, while the Rankine–Hugoniot relation couples the one-sided states to the motion of the predicted interface. Because these constraints are evaluated throughout the prescribed space–time domain, they continue to constrain the state evolution and interface motion where numerical state labels are unavailable. TRAJECTORYOP therefore requires evolution-state labels only for a subset of the training cases, for a subset of the spatial points in each labeled case, and for a subset of the stored time levels.

Our main contributions are threefold.

- We formulate an explicit-interface solution operator, in which two regional state branches and a learnable shock trajectory are combined by interface selection, so that the input-dependent shock location becomes one of the operator outputs.
- We derive the physical constraints that this representation admits. The strong-form residual is imposed inside each predicted region, and the Rankine–Hugoniot relation couples the two branches to the trajectory over the complete time interval, so that the shock location and motion are determined by the conservation law.
- We evaluate TRAJECTORYOP on Burgers, shallow-water, and one- and two-dimensional Euler benchmarks. Using only 3.125% of the evolution-state supervision-label budget used by the dense baselines, sparse-supervision TRAJECTORYOP outperforms all baselines trained under the same sparse budget on each benchmark.

2 RELATED WORK

2.1 NEURAL OPERATOR LEARNING

Neural operators learn maps between function spaces. Two standard frameworks are DeepONet, which encodes sampled input data with a branch network and query coordinates with a trunk network and combines the two outputs through a finite-dimensional reconstruction, and FNO, which lifts a discretized input field to a latent field and applies learned Fourier multipliers to its spatial modes (Lu et al., 2021; Li et al., 2021a). Both are trained with paired input–solution data, so their accuracy depends on the cases, the spatial points, and the times covered by the evolution-state supervision labels (Kovachki et al., 2023).

To constrain the operator where no evolution-state supervision label is available, physics-informed DeepONet and the physics-informed neural operator (PINO) add equation residuals to data mismatch terms, and PINO further evaluates the residual at a resolution that can differ from that of the available solution data (Wang et al., 2021; Li et al., 2021b). Such residual terms let the governing equation constrain the operator where no evolution-state supervision label exists, but the operator

still relies on a single global representation, so the residual loses its pointwise meaning across a shock.

2.2 Neural operators for hyperbolic conservation laws

Operator-learning methods for discontinuous solutions follow two main routes. The first modifies the solution parameterization so that the jump can be represented explicitly. Nonlinear reconstruction (Lanthaler et al., 2023) replaces the network output with an intermediate representation from which the solution is reconstructed. This reduces the need to approximate a discontinuity with a globally smooth output basis. RiemannONets (Peyvan et al., 2024) learn basis functions adapted to the wave features of parameterized Riemann problems. R-Adaptive DeepONet (Zhu et al., 2024) learns an input-dependent coordinate transformation that concentrates the computational grid near the discontinuity. Cut-DeepONet (Dang et al., 2026) lifts the solution into a piecewise-smooth space so that the discontinuity becomes a boundary between different regions.

The second route learns the evolution step of a numerical scheme as a neural network. LGNO (Wang et al., 2026) combines a global FNO branch with a local multiresolution branch to model smooth motion together with localized discontinuities. It further uses a high-frequency spectral penalty to suppress oscillations near steep fronts. HypNO (Zdrale et al., 2026) performs physics-informed message passing on a space–time graph of finite-volume cells. Its message functions encode upwinding and entropy admissibility.

These methods improve the representation or evolution of discontinuous solutions, but the shock location is not explicitly treated as an output constrained by a conservation relation. The model must therefore learn the shock location primarily from numerical evolution-state labels. The required supervision grows with the number of labeled cases, the number of supervised spatial points, and the number of labeled time levels. This motivates a complementary approach that represents the interface explicitly and constrains its motion through the jump conditions of the governing conservation law.

2.3 Interface modeling

Front tracking (Glimm et al., 2001) represents the interface as a low-dimensional object: a point in one dimension or a surface in higher dimensions. It advances the interface according to local jump conditions by solving local Riemann problems, while computing the smooth solutions on either side separately. This preserves sharp discontinuities without introducing a numerical transition layer. Level-set methods, developed for discontinuity tracking by Aslam (2003), also treat the interface as an independent unknown. They couple the two sides through jump conditions while representing the interface as the zero level set of an evolving function. Optimization-based variants, such as the formulation of Zahr & Persson (2018), treat the interface location as an optimization variable and solve it together with the conservation-law solution.

Learned constructions also represent the interface as an object of the network. RH-piecewise PINNs (Han et al., 2025), train a shock-detection subnetwork that partitions the solution space into smooth subspaces. A shock loss constructed from jump conditions then constrains the shock location. NDNN (Lorin & Novruzi, 2024) represents the discontinuity as a curve satisfying the Rankine–Hugoniot relation and computes a local solution on each side.

These learned constructions are generally formulated for one initial-value problem at a time. A new initial condition therefore requires a new optimization or numerical solve, so the resulting interface representation is specific to that case. TrajectoryOp combines this explicit-interface treatment with operator learning. It jointly predicts a shock trajectory, represented as an explicit network output, together with two regional state branches. The regional PDE residual constrains the state branches inside their respective smooth regions, while the Rankine–Hugoniot relation couples the one-sided states to the predicted interface motion over the complete time interval. This allows TrajectoryOp to use the conservation-law structure beyond the available evolution-state supervision labels.

3 METHOD

3.1 PROBLEM FORMULATION AND SUPERVISION

We consider the initial-value problem for a hyperbolic conservation law

$$\partial_t U + \nabla_{\boldsymbol{r}} \cdot \mathcal{F}(U) = 0, \qquad (t, \boldsymbol{r}) \in (0, T) \times \Omega, \tag{1}$$

$$U(0, \boldsymbol{r}) = U_0(\boldsymbol{r}), \qquad \boldsymbol{r} \in \Omega, \tag{2}$$

where T is the final time, $\Omega \subset \mathbb{R}^d$ is the spatial domain, $d \in \{1, 2\}$, and $\mathcal{Q} = (0, T) \times \Omega$ is the space–time domain. In one dimension, we use $\boldsymbol{r} = x$; in two dimensions, we use $\boldsymbol{r} = (x, y)$. U is the conservative state, and $\mathcal{F}$ is its physical flux, with components $\mathcal{F}(U) = (F_x(U), F_y(U))$ in two dimensions and $\mathcal{F}(U) = F_x(U)$ in one dimension. We write the state in primitive variables V and use the mapping $U = \mathcal{C}(V)$ to obtain its conservative form; in particular, $U_0 = \mathcal{C}(V_0)$.

Our goal is to learn an operator that maps an initial state field to the solution at an arbitrary space–time query point. Let a denote a problem instance with initial field $V_0(\cdot; a)$. The learned operator is $\widehat{V}(a; t, \boldsymbol{r}) = \mathcal{G}_\Theta(a; t, \boldsymbol{r})$, where $\widehat{V}$ is the predicted primitive state.

Existing operator-learning methods for hyperbolic equations train this map using initial fields and evolution-state supervision labels generated by a numerical solver. Every training case provides a discrete initial field $\boldsymbol{V}_{0,h}(a)$ on the sensor grid $\mathcal{X}_h$. Evolution-state supervision labels are available only for a subset $\mathcal{D}_{\text{label}}$ of the cases. For each labeled case $a \in \mathcal{D}_{\text{label}}$, let $\mathcal{I}_T(a)$ denote the set of nonzero time indices carrying evolution-state supervision labels. For each $n \in \mathcal{I}_T(a)$, let $\mathcal{X}_{a,n}^{\text{sup}}$ denote the spatial points carrying labels at time t_n. In the fixed-budget settings used in this work, define

$$N_{\text{case}} = |\mathcal{D}_{\text{label}}|, \qquad N_{\text{time}} = |\mathcal{I}_T(a)|, \qquad N_{\text{space}} = |\mathcal{X}_{a,n}^{\text{sup}}|.$$

The evolution-state supervision tuples are

$$\left(a, t_n, \boldsymbol{r}_i, V_h(a; t_n, \boldsymbol{r}_i)\right), \quad a \in \mathcal{D}_{\text{label}},\ n \in \mathcal{I}_T(a),\ \boldsymbol{r}_i \in \mathcal{X}_{a,n}^{\text{sup}},$$

so that the total number of supervision labels is $N_{\text{case}} \times N_{\text{time}} N_{\text{space}}$.

In conventional operator-learning methods, the solution is represented by a single global state field without a separate interface variable. The shock location must therefore be inferred indirectly from labeled state values: spatial labels provide information about the jump location, while labels at different times and for different cases reveal its motion and dependence on the initial condition. At unlabeled space–time points, the interface is weakly constrained because there is neither direct interface supervision nor a valid pointwise strong-form residual across the shock. Consequently, expanding the coverage of cases, time levels, or spatial points requires more evolution-state labels to identify the input-dependent shock trajectory, thereby increasing this supervision-label budget.

3.2 WEAK-SOLUTION STRUCTURE OF A SINGLE SHOCK

To reduce this reliance on evolution-state supervision labels, we use the weak-solution structure of a single tracked shock to represent its interface explicitly and constrain it through conservation across the interface. Let x denote the coordinate in the shock-propagation direction, and write the spatial coordinate as $\boldsymbol{r} = (x, \boldsymbol{r}_\perp)$. Here, $\boldsymbol{r}_\perp$ denotes the transverse coordinate: it is absent in one dimension and equals y in two dimensions. We use $\xi(\boldsymbol{r}_\perp, t)$ to denote the x-position of the shock at time t and transverse coordinate $\boldsymbol{r}_\perp$. The shock interface in the space–time domain $\mathcal{Q}$ is therefore

$$\Gamma_\xi = \{(t, \boldsymbol{r}) \in \mathcal{Q} : x = \xi(\boldsymbol{r}_\perp, t)\}.$$

In one dimension, the interface is $x = \xi(t)$; in two dimensions, it is $x = \xi(y, t)$. It separates $\mathcal{Q}$ into the two regions

$$\mathcal{Q}_\xi^- = \{(t, \boldsymbol{r}) \in \mathcal{Q} : x < \xi(\boldsymbol{r}_\perp, t)\}, \qquad \mathcal{Q}_\xi^+ = \{(t, \boldsymbol{r}) \in \mathcal{Q} : x > \xi(\boldsymbol{r}_\perp, t)\}.$$

We denote the primitive states on the two sides by V^- and V^+, with conservative states $U^\pm = \mathcal{C}(V^\pm)$.

Within either smooth region, the solution satisfies the strong-form conservation law $\partial_t U^\pm + \nabla_{\boldsymbol{r}} \cdot \mathcal{F}(U^\pm) = 0, (t, \boldsymbol{r}) \in \mathcal{Q}_\xi^\pm$. Across the moving shock, the states and fluxes on the two sides must satisfy the Rankine–Hugoniot condition, which is $-\xi_t[U] + [F_x] - \xi_y[F_y] = 0$ in two dimensions and reduces to $-\xi_t[U] + [F_x] = 0$ in one dimension. For any quantity ϕ, let $[\phi] = \phi^+ - \phi^-$ denote its jump across Γ_ξ.

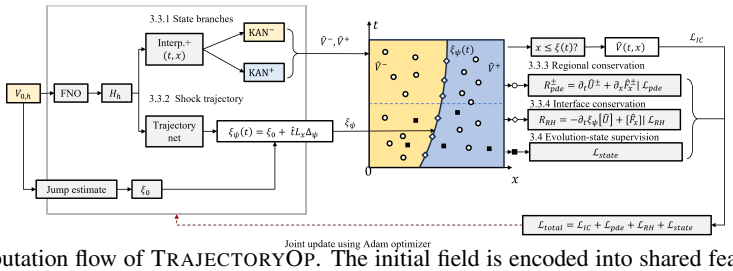


Figure 2: Computation flow of TRAJECTORYOP. The initial field is encoded into shared features for the two regional state branches and the shock trajectory network. The predicted trajectory defines the interface used for branch selection, regional conservation residuals, and the Rankine–Hugoniot interface residual. These terms are jointly optimized with the initial-condition and evolution-state supervision losses.

Thus, the weak solution is characterized by the conservation law within each region and a jump relation coupling the two states to the shock motion. We build these requirements into the operator representation and its training constraints.

3.3 OPERATOR LEARNING WITH AN EXPLICITLY SHOCK TRAJECTORY

We first describe TrajectoryOp for a one-dimensional problem with a single tracked shock, then state the changes needed in two dimensions. The method has four connected stages: encoding the initial field and predicting two regional states; predicting the shock trajectory and selecting the output state; enforcing conservation within each predicted region; and enforcing the Rankine–Hugoniot relation on the predicted interface. The first two stages define the solution operator, while the last two provide physical training constraints. Figure 2 summarizes this four-stage computation and the corresponding loss terms; the conceptual comparison with a conventional supervised operator is given in Figure 1. Evolution-state supervision is defined in the next subsection.

3.3.1 ENCODING AND REGIONAL STATE BRANCHES

Let $\mathcal{X}_h$ be the sensor grid and $\boldsymbol{V}_{0,h}(a) = \{V_{0,h}(x_i; a)\}_{x_i \in \mathcal{X}_h}$ the discrete initial field for an instance a. Define $\widehat{x} = (x - x_{\min})/L_x$ and $\widehat{t} = t/T$, where $L_x = x_{\max} - x_{\min}$. An FNO encoder maps the initial field and the normalized spatial coordinate channel to a grid-aligned latent field:

$$H_h(a) = \mathcal{E}_\theta(\mathrm{cat}[\boldsymbol{V}_{0,h}(a), \widehat{x}]) \,. \tag{3}$$

A Fourier layer in the encoder is $H^{\ell+1} = \varphi\big(H^\ell + \mathcal{F}^{-1}(\mathcal{R}_\ell \mathcal{F}(H^\ell)) + W_\ell H^\ell\big)$, where $\mathcal{R}_\ell$ acts on retained Fourier modes and W_ℓ is a pointwise linear map. At a query point (t, x), linear interpolation of $H_h(a)$ followed by a bounded nonlinearity gives $\boldsymbol{z}_\theta(a, x) = \tanh(\mathcal{I}_h[H_h(a)](x))$. We concatenate this feature with the normalized query coordinates to form $\boldsymbol{q}_\theta(a; t, x) = (\boldsymbol{z}_\theta(a, x), \widehat{t}, \widehat{x})$. Two independent Kolmogorov–Arnold network (KAN) decoders (Liu et al., 2025) then predict the primitive states on the negative ($x < \xi_\psi$) and positive ($x > \xi_\psi$) sides of the shock:

$$\widehat{V}^\sigma(a; t, x) = \mathcal{P}(\mathcal{K}_{\theta^\sigma}(\boldsymbol{q}_\theta(a; t, x))) \,, \qquad \sigma \in \{-, +\}. \tag{4}$$

The map $\mathcal{P}$ applies softplus to components that must be positive (water depth for shallow water; density and pressure for Euler). The two branches share the encoded initial field but represent the smooth states on opposite sides of the shock.

3.3.2 SHOCK TRAJECTORY AND INTERFACE SELECTION

We estimate the initial shock location $\xi_0(a)$ from the same discrete initial field by locating its largest jump between adjacent cells. This estimate uses no additional shock-trajectory label. The trajectory network uses $H_h(a)$ and $\widehat{t}$ to predict a dimensionless displacement Δ_ψ from the initial location:

$$\xi_\psi(a; t) = \xi_0(a) + \widehat{t} L_x \Delta_\psi(H_h(a), \widehat{t}). \tag{5}$$

The factor L_x restores the spatial scale, and multiplication by $\widehat{t}$ enforces $\xi_\psi(a; 0) = \xi_0(a)$. The predicted interface is $\Gamma_\psi(a) = \{(t, x) \in \mathcal{Q} : x = \xi_\psi(a; t)\}$. At each query point, the interface selects between the two regional states:

$$\widehat{V}(a; t, x) = \begin{cases} \widehat{V}^-(a; t, x), & x \le \xi_\psi(a; t), \\ \widehat{V}^+(a; t, x), & x > \xi_\psi(a; t). \end{cases} \tag{6}$$

Thus, the decoders specify the states on either side, while the trajectory network specifies where the interface lies. Together they define a piecewise solution, with conservative prediction $\widehat{U} = \mathcal{C}(\widehat{V})$.

3.3.3 REGIONAL CONSERVATION LOSS

The predicted interface partitions the space–time domain into $\mathcal{Q}_\psi^-(a) = \{(t,x) \in \mathcal{Q} : x < \xi_\psi(a;t)\}$ and $\mathcal{Q}_\psi^+(a) = \{(t,x) \in \mathcal{Q} : x > \xi_\psi(a;t)\}$. For each $\sigma \in \{-,+\}$, sample regional collocation points $\mathcal{S}_{\mathrm{PDE}}^\sigma \subset \mathcal{Q}_\psi^\sigma(a)$. The corresponding branch provides $\widehat{U}^\sigma = \mathcal{C}(\widehat{V}^\sigma)$ and $\widehat{\mathcal{F}}_x^\sigma = F_x(\widehat{U}^\sigma)$, yielding the one-dimensional conservation residual

$$R_{\mathrm{PDE}}^\sigma = \partial_t \widehat{U}^\sigma + \partial_x \widehat{\mathcal{F}}_x^\sigma, \qquad \sigma \in \{-,+\}.$$

The regional conservation loss is

$$\mathcal{L}_{\mathrm{PDE}} = \sum_{\sigma\in\{-,+\}} \left\langle \|R_{\mathrm{PDE}}^\sigma\|_2^2 \right\rangle_{\mathcal{S}_{\mathrm{PDE}}^\sigma}, \tag{7}$$

where $\langle f \rangle_{\mathcal{S}}$ is the empirical average over a finite set $\mathcal{S}$. Each strong-form residual is evaluated on its smooth regional branch, not across the discontinuity in the selected piecewise output. The governing equation therefore constrains regional collocation points without numerical state labels at those points.

3.3.4 INTERFACE CONSERVATION LOSS

Sample interface collocation points $\mathcal{S}_\Gamma \subset \Gamma_\psi(a)$ and evaluate both regional branches at each point. With $[\widehat{U}] = \widehat{U}^+ - \widehat{U}^-$ and $[\widehat{\mathcal{F}}_x] = \widehat{\mathcal{F}}_x^+ - \widehat{\mathcal{F}}_x^-$, the one-dimensional Rankine–Hugoniot residual is

$$R_{\mathrm{RH}} = -\xi_{\psi,t}[\widehat{U}] + [\widehat{\mathcal{F}}_x].$$

The interface loss is

$$\mathcal{L}_{\mathrm{RH}} = \left\langle \|R_{\mathrm{RH}}\|_2^2 \right\rangle_{\mathcal{S}_\Gamma}. \tag{8}$$

This relation couples the interface velocity to the jumps in conservative state and flux. It constrains interface motion without numerical state labels at the interface collocation points.

3.3.5 EXTENSION TO TWO DIMENSIONS

The same four stages apply in two dimensions. The encoder additionally receives $\widehat{y} = (y - y_{\min})/L_y$, where $L_y = y_{\max} - y_{\min}$, and the query feature includes $\widehat{y}$:

$$H_h(a) = \mathcal{E}_\theta(\mathrm{cat}[\boldsymbol{V}_{0,h}(a), \widehat{x}, \widehat{y}]),$$
$$\boldsymbol{q}_\theta(a;t,x,y) = (\boldsymbol{z}_\theta(a,x,y), \widehat{t}, \widehat{x}, \widehat{y}).$$

Here $\boldsymbol{z}_\theta$ uses bilinear interpolation of $H_h(a)$. The initial location $\xi_0(a;y)$ is estimated along each transverse column. Using a spatially pooled representation of $H_h(a)$, the trajectory network predicts

$$\xi_\psi(a;y,t) = \xi_0(a;y) + \widehat{t} L_x \Delta_\psi(H_h(a), \widehat{t}, \widehat{y}).$$

The interface is now the curve $x = \xi_\psi(a;y,t)$; at each query point, Eq. (6) uses $\xi_\psi(a;y,t)$ in place of $\xi_\psi(a;t)$. The regional and interface residuals gain transverse terms:

$$R_{\mathrm{PDE}}^\sigma = \partial_t \widehat{U}^\sigma + \partial_x \widehat{\mathcal{F}}_x^\sigma + \partial_y \widehat{\mathcal{F}}_y^\sigma,$$
$$R_{\mathrm{RH}} = -\xi_{\psi,t}[\widehat{U}] + [\widehat{\mathcal{F}}_x] - \xi_{\psi,y}[\widehat{\mathcal{F}}_y].$$

The loss definitions in Eqs. (7) and (8) remain unchanged, with regional points sampled on either side of the predicted curve and interface points sampled on it.

3.4 EVOLUTION-STATE SUPERVISION

Let $\mathcal{D}_{\mathrm{train}}$ be the set of training cases. Initial states are available for all cases and provide the initial-condition loss

$$\mathcal{L}_{\mathrm{IC}} = \frac{1}{|\mathcal{D}_{\mathrm{train}}|} \sum_{a\in\mathcal{D}_{\mathrm{train}}} \left\langle \left\| \widehat{V}(a;0,\boldsymbol{r}_i) - V_{0,h}(a;\boldsymbol{r}_i) \right\|_1 \right\rangle_{\boldsymbol{r}_i\in\mathcal{X}_h}. \tag{9}$$

Nonzero-time reference states are available only for a subset with evolution-state supervision labels, $\mathcal{D}_{\text{label}} \subseteq \mathcal{D}_{\text{train}}$. Let $\mathcal{I}_T(a) \subseteq \{1, \dots, N_t - 1\}$ be the nonzero time indices with evolution-state supervision labels for case a, and let $\mathcal{X}_{a,n}^{\text{sup}} \subseteq \mathcal{X}_h$ be the spatial points carrying those evolution-state supervision labels at time t_n. We use the relative ℓ_1 state loss

$$\mathcal{L}_{\text{state}} = \frac{1}{|\mathcal{D}_{\text{label}}|} \sum_{a \in \mathcal{D}_{\text{label}}} \frac{1}{|\mathcal{I}_T(a)|} \sum_{n \in \mathcal{I}_T(a)} \frac{\sum_{\boldsymbol{r}_i \in \mathcal{X}_{a,n}^{\text{sup}}} \left\| \widehat{V}(a; t_n, \boldsymbol{r}_i) - V_h(a; t_n, \boldsymbol{r}_i) \right\|_1}{\sum_{\boldsymbol{r}_i \in \mathcal{X}_{a,n}^{\text{sup}}} \left\| V_h(a; t_n, \boldsymbol{r}_i) \right\|_1}. \tag{10}$$

The evolution-state supervision-label budgets used in our experiments are specified in Section 4; solver details are given in Appendix A.

3.5 Training objective and procedure

We combine the initial-condition, state, regional conservation, and interface conservation losses into the objective

$$\mathcal{L}(\Theta) = w_{\text{IC}}\mathcal{L}_{\text{IC}} + w_{\text{state}}\mathcal{L}_{\text{state}} + w_{\text{PDE}}\mathcal{L}_{\text{PDE}} + w_{\text{RH}}\mathcal{L}_{\text{RH}}, \tag{11}$$

where the w's are loss weights. At each update, we sample one minibatch from $\mathcal{D}_{\text{train}}$ for the initial-condition and physical losses, and a minibatch of cases with evolution-state supervision labels from $\mathcal{D}_{\text{label}}$ for the state loss. We evaluate the PDE residual in the two predicted regions and the Rankine–Hugoniot residual on the predicted interface over $(0, T)$. The four terms are combined and one Adam step jointly updates the encoder, the two state decoders, and the trajectory network. Algorithm 1 summarizes one update; optimizer settings, batch sizes, and collocation counts are specified in the appendix. The same steps apply in two dimensions with the coordinate, trajectory, and residual forms given above.

Algorithm 1 One minibatch training step of TrajectoryOp.

Input: Training cases $\mathcal{D}_{\text{train}}$, cases with evolution-state supervision labels $\mathcal{D}_{\text{label}}$, sensor grid $\mathcal{X}_h$, loss weights $\{w_{\text{IC}}, w_{\text{state}}, w_{\text{PDE}}, w_{\text{RH}}\}$
Output: Updated parameters Θ

1: Sample initial/physics batch $B_{\text{phys}} \subset \mathcal{D}_{\text{train}}$ and labeled batch $B_{\text{label}} \subset \mathcal{D}_{\text{label}}$
2: **for** each case $a \in B_{\text{phys}}$ **do**
3: **Encoding:** Obtain $H_h(a)$ by Eq. (3) and the two branches by Eq. (4)
4: **Trajectory/output:** Estimate ξ_0; compute ξ_ψ by Eq. (5) and select $\widehat{V}$ at $t = 0$ by Eq. (6) for $\mathcal{L}_{\text{IC}}$ in Eq. (9)
5: **Regional conservation:** Sample $\mathcal{S}_{\text{PDE}}^{\pm} \subset \mathcal{Q}_\psi^{\pm}$; compute $\mathcal{L}_{\text{PDE}}$ from the matching branches by Eq. (7)
6: **Interface conservation:** Sample $\mathcal{S}_\Gamma \subset \Gamma_\psi$; compute $\mathcal{L}_{\text{RH}}$ from both branches and trajectory derivatives by Eq. (8)
7: **end for**
8: On B_{label}, repeat encoding and Eqs. (4)–(6) at labeled points to evaluate $\mathcal{L}_{\text{state}}$ by Eq. (10)
9: Average the minibatch losses, form $\mathcal{L}(\Theta)$ by Eq. (11), and update Θ with Adam.

4 Experiments

4.1 Setup

We consider four conservation-law problems: Burgers, the one-dimensional shallow-water equations, a one-dimensional Euler rarefaction–contact–shock problem, and a two-dimensional Euler shock–entropy interaction. The first three contain a one-dimensional moving shock. In the last problem, the shock deforms in the transverse direction as it interacts with an entropy perturbation.

We use the following primitive and conservative variables in the four benchmarks. For Burgers, $V = u$, $U = u$, and $\mathcal{C}(u) = u$. For the shallow-water equations, $V = (h, u)^\mathsf{T}$, $U = (h, hu)^\mathsf{T}$, and $\mathcal{C}(h, u)^\mathsf{T} = (h, hu)^\mathsf{T}$, with gravitational acceleration $g = 1$. For the one-dimensional Euler

Table 1: Space–time relative L_1 error over all nine test times. The sparse block compares TRAJECTORYOP with the three data-driven baselines under the same sparse evolution-state supervision-label budget; the dense block gives a high-budget baseline reference. Bold marks the lowest error in each block.

Method	Burgers	Shallow water	1-D Euler	2-D Euler
Sparse evolution-state supervision labels (3.125% of the dense budget)				
FNO	0.03454	0.04983	0.06446	0.00594
DeepONet	0.05747	0.05343	0.10881	0.01132
Shift-DeepONet	0.03851	0.02714	0.07325	0.00995
TRAJECTORYOP	**0.00789**	**0.01418**	**0.03789**	**0.00324**
Dense evolution-state supervision labels (100%): baseline reference				
FNO	**0.00817**	**0.00905**	**0.01649**	**0.00188**
DeepONet	0.05635	0.04071	0.07067	0.00906
Shift-DeepONet	0.02084	0.01591	0.02904	0.00949

equations, $V = (\rho, u, p)^\mathsf{T}$ and $U = (\rho, \rho u, E)^\mathsf{T}$, where

$$E = \frac{p}{\gamma - 1} + \frac{\rho u^2}{2}, \qquad \mathcal{C}(\rho, u, p)^\mathsf{T} = \left(\rho, \rho u, \frac{p}{\gamma - 1} + \frac{\rho u^2}{2}\right)^\mathsf{T}.$$

For the two-dimensional Euler equations, $V = (\rho, u, v, p)^\mathsf{T}$ and $U = (\rho, \rho u, \rho v, E)^\mathsf{T}$, where

$$E = \frac{p}{\gamma - 1} + \frac{\rho(u^2 + v^2)}{2}, \qquad \mathcal{C}(\rho, u, v, p)^\mathsf{T} = \left(\rho, \rho u, \rho v, \frac{p}{\gamma - 1} + \frac{\rho(u^2 + v^2)}{2}\right)^\mathsf{T}.$$

Thus, the networks predict primitive states, while the conservation laws and their physical fluxes are evaluated after applying $\mathcal{C}$. Throughout the Euler experiments, $\gamma = 1.4$.

The numerical solutions are stored at $t_n = nT/8$, $n = 0, \ldots, 8$. We report the relative L_1 error averaged over every test case, all nine times, all points on the numerical grid, and all state components. The test sets contain 1024 cases for each one-dimensional problem and 128 cases for the two-dimensional problem. The complete test trajectory is used in every experiment, regardless of which states are available during training.

Every method receives all training initial fields: 1024 in one dimension and 512 in two dimensions. Evolution-state supervision labels are provided only for selected cases, spatial points, and times. We write $nN/sM/t_i$–t_j for N cases with evolution-state supervision labels, M supervised points at each supervision-label time, and the time set $t_i, \ldots, t_j$.

The main comparison uses two supervision-label settings. The sparse-supervision setting is $n16/s64/t_1$–t_4 in one dimension and $n16/s4096/t_1$–t_4 in two dimensions. The dense-supervision setting is $n64/s256/t_1$–t_8 and $n64/s16384/t_1$–t_8, respectively. The sparse-supervision setting uses 25% of the cases with evolution-state supervision labels, 25% of the spatial points, and 50% of the supervision-label time levels of the dense-supervision setting. It therefore uses $\mathbf{3.125}\%$ of the dense evolution-state supervision-label budget; the supervision-label time coverage ends at $T/2$. The initial-condition, PDE, and Rankine–Hugoniot losses still use the complete initial-field set and sample the whole interval $[0, T]$.

Every run uses eight independent validation cases. Checkpoint selection uses the relative L_1 error on the same fixed spatial-point budget as training and on the training time coverage together with t_0. For example, the t_1–t_4 runs select checkpoints using $t_0, \ldots, t_4$, whereas the dense-supervision runs use all nine levels. Later times and spatial points without supervision labels therefore do not determine the selected checkpoint. The sparse and dense designations refer to evolution-state supervision labels used for parameter updates. Numerical solvers, network sizes, and optimization settings are given in Appendix A.

4.2 MAIN RESULTS

Table 1 compares sparse-supervision TRAJECTORYOP with the three data-driven baselines under the same sparse evolution-state supervision-label budget. The dense block reports the same baselines with dense evolution-state supervision labels and serves as a high-budget reference. The sparse block supervises only 3.125% of the evolution-state supervision-label budget used by the dense baselines.

Within the sparse block, TRAJECTORYOP attains the lowest error on all four benchmarks, reducing the error of the strongest sparse baseline by 77% on Burgers, 48% on shallow water, 41% on one-

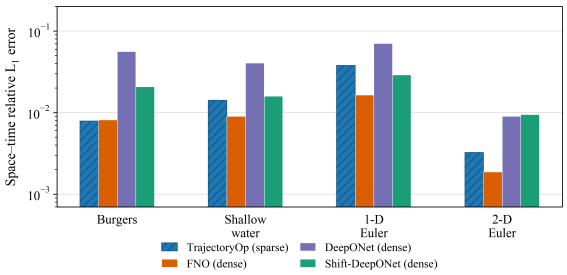


Figure 3: Space–time relative L_1 error; lower is better.

dimensional Euler, and 45% on the two-dimensional shock–entropy interaction. The improvement therefore reflects the physical constraints in addition to the reduced evolution-state supervision-label budget.

FNO gives the lowest error in the dense baseline block on all four benchmarks, which serves as a high-budget reference. Across the two blocks, sparse TRAJECTORYOP is more accurate than dense DeepONet on all four benchmarks, than dense Shift-DeepONet on three benchmarks, and than dense FNO on Burgers. Figure 3 plots these errors on a logarithmic scale. Temporal coverage is studied in Appendix A.4 with the labeled-case and spatial-point counts fixed.

4.3 NUMBER OF SUPERVISION-LABEL CASES AND SPATIAL POINTS

We vary the number of cases carrying evolution-state supervision labels and the number of supervised spatial points on the one- and two-dimensional Euler problems. All TRAJECTORYOP runs use evolution-state supervision labels at $t_1, \ldots, t_4$ and the same complete set of initial fields. FNO remains at $n64/s256/t_1$–t_8 in one dimension and $n64/s16384/t_1$–t_8 in two dimensions throughout this subsection.

For the case-count study, the spatial point count is fixed at 256 in one dimension and 16384 in two dimensions; for the spatial-sample study, the number of labeled cases is fixed at 64. The tested spatial counts are 64, 128, and 256 in one dimension and 4096, 8192, and 16384 in two dimensions. Table 2 shows a modest case-count effect and nearly unchanged spatial-sample results; the numerical grid is unchanged.

Table 2: Effect of the labeled-case count and supervised spatial-point count on TRAJECTORYOP. TRAJECTORYOP uses labels at $t_1, \ldots, t_4$; the fixed FNO column uses dense supervision labels through t_8. Errors use nine times; bold marks the lowest error in each row.

	TRAJECTORYOP			FNO
	$n16$	$n32$	$n64$	dense
(a) Cases with evolution-state supervision labels				
1-D Euler	0.03828	0.02947	0.02943	**0.01649**
2-D Euler	0.00320	0.00330	0.00321	**0.00188**
(b) Supervised spatial points (s, 1-D / 2-D)				
	$s64/4096$	$s128/8192$	$s256/16384$	dense
1-D Euler	0.02753	0.02742	0.02943	**0.01649**
2-D Euler	0.00307	0.00312	0.00321	**0.00188**

5 CONCLUSION AND FUTURE WORK

We introduce a novel operator learning framework called TRAJECTORYOP that integrates an explicitly shock trajectory for hyperbolic conservation laws. TRAJECTORYOP is composed by two regional state mappings and a trajectory network initialized at the shock location estimated from the initial field. Regional PDE residuals constrain the state evolution, while the Rankine–Hugoniot condition couples the one-sided states to interface motion throughout the prescribed time interval. Across Burgers, shallow-water, and one- and two-dimensional Euler benchmarks, TrajectoryOp achieves comparable performance with existing operator learning methods with only about 3% supervised information, demonstrating the potential of explicitly shock trajectories with physical constraints to reduce reliance on supervised solution data. Future work will extend this formulation to multiple interacting shocks and interfaces with changing topology, and investigate entropy-admissibility constraints for more general wave configurations.

REFERENCES

Simon Arridge, Peter Maass, Ozan Öktem, and Carola-Bibiane Schönlieb. Solving inverse problems using data-driven models. *Acta Numerica*, 28:1–174, 2019. doi: 10.1017/S0962492919000059.

Tariq D. Aslam. A level set algorithm for tracking discontinuities in hyperbolic conservation laws II: Systems of equations. *Journal of Scientific Computing*, 19(1-3):37–62, 2003.

Peter Benner, Serkan Gugercin, and Karen Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM Review*, 57(4):483–531, 2015. doi: 10.1137/130932715.

Ha Dang, Sebastian Schmidt, and Juergen Hesser. Smooth piecewise cutting for neural operator to handle discontinuities and sharp transitions. *arXiv preprint arXiv:2605.19823*, 2026.

J. Glimm, J. W. Grove, X. L. Li, K.-M. Shyue, Y. Zeng, and Q. Zhang. Conservative front tracking and level set algorithms. *Proceedings of the National Academy of Sciences*, 98(25):14198–14201, 2001. doi: 10.1073/pnas.251420998.

Jiarong Han, Junxi Chen, Fei Hu, and Liangcai Mei. A study on shock capturing in the Burgers' equation based on RH-piecewise PINNs. *Computers & Mathematics with Applications*, 199: 309–324, 2025. doi: 10.1016/j.camwa.2025.10.003.

Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to PDEs. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

Thorsten Kurth, Shashank Subramanian, Peter Harrington, Jaideep Pathak, Morteza Mardani, David Hall, Andrea Miele, Karthik Kashinath, and Anima Anandkumar. FourCastNet: Accelerating global high-resolution weather forecasting using adaptive fourier neural operators. In *Proceedings of the Platform for Advanced Scientific Computing Conference (PASC '23)*, pp. 1–11, New York, NY, USA, 2023. Association for Computing Machinery. doi: 10.1145/3592979.3593412.

Samuel Lanthaler, Roberto Molinaro, Patrik Hadorn, and Siddhartha Mishra. Nonlinear reconstruction for operator learning of PDEs with discontinuities. In *International Conference on Learning Representations*, 2023.

Peter D. Lax. *Hyperbolic Systems of Conservation Laws and the Mathematical Theory of Shock Waves*, volume 11 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. Society for Industrial and Applied Mathematics, 1973. doi: 10.1137/1.9781611970562.

Randall J. LeVeque. *Finite Volume Methods for Hyperbolic Problems*, volume 31 of *Cambridge Texts in Applied Mathematics*. Cambridge University Press, 2002. doi: 10.1017/CBO9780511791253.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021a.

Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *arXiv preprint arXiv:2111.03794*, 2021b.

Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y. Hou, and Max Tegmark. KAN: Kolmogorov–arnold networks. In *International Conference on Learning Representations*, 2025.

Emmanuel Lorin and Arian Novruzi. Non-diffusive neural network method for hyperbolic conservation laws. *Journal of Computational Physics*, 513:113161, 2024. doi: 10.1016/j.jcp.2024.113161.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3:218–229, 2021.

Benjamin Peherstorfer, Karen Willcox, and Max Gunzburger. Survey of multifidelity methods in uncertainty propagation, inference, and optimization. *SIAM Review*, 60(3):550–591, 2018. doi: 10.1137/16M1082469.

Ahmad Peyvan, Vivek Oommen, Ameya D. Jagtap, and George Em Karniadakis. RiemannONets: Interpretable neural operators for riemann problems. *Computer Methods in Applied Mechanics and Engineering*, 2024.

Hao Wang, Chi-Wang Shu, and Qi Tang. LGNO: A local–global neural operator for hyperbolic conservation laws. *arXiv preprint arXiv:2606.18221*, 2026.

Sifan Wang, Hanwen Wang, and Paris Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed DeepONets. *Science Advances*, 7(40):eabi8605, 2021. doi: 10.1126/sciadv.abi8605.

Michael J. Zahr and Per-Olof Persson. An optimization-based approach for high-order accurate discretization of conservation laws with discontinuities. *Journal of Computational Physics*, 365: 105–134, 2018. doi: 10.1016/j.jcp.2018.03.029.

Dimitrije Zdrale, Cassie An Jeng, Katie Wang, Sonia Vanier, Alexandre Bayen, and Hossein Nick Zinat Matin. HypNO: A graph-based neural operator with physics-informed message passing for hyperbolic conservation laws. *arXiv preprint arXiv:2607.20541*, 2026.

Yameng Zhu, Jingrun Chen, and Weibing Deng. R-adaptive DeepONet: Learning solution operators for PDEs with discontinuous solutions using an r-adaptive strategy. *arXiv preprint arXiv:2408.04157*, 2024.

Yinhao Zhu and Nicholas Zabaras. Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification. *Journal of Computational Physics*, 366:415–447, 2018. doi: 10.1016/j.jcp.2018.04.018.

AI USE STATEMENT

In this work, we used generative AI tools to draft and edit the manuscript for readability. We have not used generative AI tools to generate synthetic data, to develop the theoretical models or conceptual frameworks, to formulate mathematical claims, to design the research methodology or experiments, or to implement the methods. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

A EXPERIMENTAL DETAILS

A.1 INITIAL-STATE FAMILIES

The Burgers initial states are parameterized by $u_L \in [1.0, 2.0]$, $u_R \in [-0.5, 0.5]$, and $x_0 \in [-0.2, 0.2]$. The inequality $u_L > u_R$ gives a single shock. The shallow-water initial states use $h_L \in [1.0, 2.0]$, $h_R \in [0.5, 1.0]$, $u_L \in [0.2, 0.7]$, $u_R \in [-0.5, -0.1]$, and $x_0 \in [-0.2, 0.2]$, with $g = 1$.

The one-dimensional Euler states are sampled from the six-parameter family

$$(\rho_L, \rho_R, u_L, p_L, p_R, x_0),$$

with $u_R = 0$ and $\gamma = 1.4$. The component ranges are

$$\rho_L \in [0.30, 1.20], \quad \rho_R \in [0.10, 0.70], \quad u_L \in [0, 1],$$
$$p_L \in [0.90, 4.10], \quad p_R \in [0.05, 0.70], \quad x_0 \in [-0.5, 0.5].$$

The pressure-function Riemann test retains cases with the RCS pattern.

The two-dimensional initial condition follows the Aslam configuration. The upstream state is $(\rho, u, v, p) = (1, 1.4199, 0, 1)$ outside a circular entropy perturbation, and the post-shock state is $(1.3412, 1.058, 0, 1.5127)$. The perturbation amplitude, radius, and center are sampled from

$$A \in [0.20, 0.30], \quad R \in [0.22, 0.28], \quad x_c \in [0.27, 0.33], \quad y_c \in [0.47, 0.53].$$

Table 3: Domain, time horizon, finite-volume resolution, and boundary conditions for the four benchmark problems. The listed spacings are the cell widths used by the numerical solver.

Benchmark	Ω	T	Grid	Δx	Δy	Boundary
Burgers	$[-1, 1]$	0.5	512	$2/512$	–	Extrapolation
Shallow water	$[-1, 1]$	0.5	512	$2/512$	–	Extrapolation
1-D Euler RCS	$[-2, 2]$	0.3	512	$4/512$	–	Extrapolation
2-D Euler shock–entropy	$[0, 1.5] \times [0, 1]$	0.4	256×128	$3/512$	$1/128$	Extrapolation in x; reflection in y

The nominal initial shock location is $x_0 = 2/3$; this value is used only to describe the data family. For each discrete initial field, ξ_0 is estimated from the largest summed absolute jump between adjacent x-cells, using the midpoint of the two cell centers. In two dimensions, the estimate is made independently along each y column.

A.2 Optimization and numerical data

The numerical states are generated with a cell-centered finite-volume solver. At each cell face, the Rusanov flux is

$$\mathcal{H}(U_L, U_R) = \frac{\mathcal{F}(U_L) + \mathcal{F}(U_R)}{2} - \frac{a(U_L, U_R)}{2}(U_R - U_L),$$

where a is the largest absolute characteristic speed normal to that face. All four benchmarks use the same solver configuration: piecewise-constant spatial reconstruction, the Rusanov flux above, and SSP-RK3 time integration. The one-dimensional benchmarks use $\mathrm{CFL} = 0.45$, while the two-dimensional Euler benchmark uses $\mathrm{CFL} = 0.35$. The time step is adaptive: it is recomputed before each SSP-RK3 step from the maximum characteristic speed over the current batch of initial states being advanced and is clipped to the next requested output time. The step is therefore shared within a batch but changes as the numerical state evolves. The resulting domains, time horizons, and cell spacings are listed in Table 3. The batch size is 64 for the one-dimensional solvers and 16 for the two-dimensional solver. More precisely, for the one-dimensional problems we use

$$\Delta t = \min\left(\frac{\mathrm{CFL}\,\Delta x}{a_{\max}}, t_{\mathrm{target}} - t\right),$$

where $a_{\max}$ is the maximum characteristic speed in the current batch. For the two-dimensional problem, this becomes

$$\Delta t = \min\left(\frac{\mathrm{CFL}\,\Delta x}{a_{x,\max}}, \frac{\mathrm{CFL}\,\Delta y}{a_{y,\max}}, t_{\mathrm{target}} - t\right),$$

with the maximum normal characteristic speeds computed separately in the two coordinate directions. Extrapolation is used at the one-dimensional boundaries; the two-dimensional solver uses extrapolation in x and reflection in y. Nine states are stored at $t_n = nT/8$, $n = 0, \ldots, 8$.

The training initial-field sets contain 1024 cases in one dimension and 512 in two dimensions. The sparse evolution-state supervision labels use 16 cases and 64 spatial points at each supervision-label time in one dimension (4096 in two dimensions), with times $t_1, \ldots, t_4$. The dense evolution-state supervision labels use 64 cases, 256 spatial points (16384 in two dimensions), and $t_1, \ldots, t_8$. Spatial supervision samples the same numerical grids, with the point count measured at each supervision-label time. The initial shock-location estimate is stored with each initial field. It is computed directly from the discrete field by locating the largest summed absolute jump between adjacent cells and taking the corresponding cell-center midpoint; no independently supplied shock-location supervision label is used. In the two-dimensional benchmark, the resulting estimate is constant in y up to the spatial quantization of the grid.

All methods use 8 independent validation cases. For checkpoint selection, we use the same fixed spatial-point budget as training and the time levels covered by the training evolution-state supervision labels together with t_0; thus a run using supervision labels at $t_1, \ldots, t_4$ is selected on $t_0, \ldots, t_4$, while the dense run uses all nine levels. The checkpoint with the smallest validation relative L_1 error is used for testing. The test sets contain 1024 cases in one dimension and 128 in two dimensions. Test errors use the full spatial grid, every case, and all nine times.

Training uses Adam for 5000 epochs. The training and supervision-label-selection seeds are both 1234. The learning rate is 10^{-3} for TrajectoryOp and 5×10^{-4} for FNO, DeepONet, and Shift-

DeepONet. In the four-term objective, the fixed weights for the initial-condition, state, PDE, and Rankine–Hugoniot terms are

$$(w_{\mathrm{IC}}, w_{\mathrm{state}}, w_{\mathrm{PDE}}, w_{\mathrm{RH}}) = (10, 10, 1, 10).$$

The PDE and Rankine–Hugoniot residuals are averaged over their sampled points and residual components.

For TRAJECTORYOP, the initial/physical minibatch sizes are 16 in one dimension and 1 in two dimensions. The sparse-supervision-label minibatch sizes are 16 for Burgers and shallow water, 4 for one-dimensional Euler, and 1 for two-dimensional Euler. With dense evolution-state supervision labels they are 16 in one dimension and 1 in two dimensions. Each physical case uses 64 regional and 32 trajectory collocation points in one dimension, or 256 and 128 points in two dimensions. These points are resampled over the complete time interval at each update. The initial-condition loss uses the full initial grid.

The sparse-supervision baselines use initial and supervision-label minibatches of size 16 in one dimension and 4 in two dimensions. With dense evolution-state supervision labels, the one-dimensional initial and supervision-label minibatch sizes are 16 and 4, respectively; the two-dimensional sizes are both 1. Their state supervision uses the same case budget, spatial point budget, and time indices as the corresponding TRAJECTORYOP setting.

The TRAJECTORYOP encoder has three Fourier layers of width 48, retaining 16 modes in one dimension and 12×8 modes in two dimensions. Each state KAN has three hidden layers of width 33, with grid size 5 and spline order 3. The shock-trajectory readout uses four attention heads. In two dimensions, it reads from latent features pooled to a 32×16 grid.

For comparison, FNO, DeepONet, and Shift-DeepONet use the same discrete initial fields and the same evolution-state supervision-label cases, spatial points, and nonzero time indices as the corresponding TRAJECTORYOP run. The FNO receives the complete initial field together with normalized space and time coordinates. Its one-dimensional implementation has width 64, four Fourier layers, and retains 16 modes; its two-dimensional implementation uses 16×8 modes. Here the baseline FNO matches the one-dimensional mode count, uses a larger width and depth, and retains 16 x-modes in two dimensions in place of the 12 used by the TRAJECTORYOP encoder. The DeepONet and Shift-DeepONet use 128 fixed sensors of the initial field in one dimension. Their branch networks have four hidden layers of width 64 and output a 128-dimensional latent/basis representation, while their trunk networks also have four hidden layers of width 64. Shift-DeepONet additionally uses four-layer, width-64 scale and shift maps to transform the trunk coordinates from the input sensors before trunk evaluation.

The baseline objective contains a direct initial-field term and a numerical state term,

$$\mathcal{L}_{\mathrm{base}} = w_{\mathrm{IC}}\mathcal{L}_{\mathrm{IC}} + w_{\mathrm{state}}\mathcal{L}_{\mathrm{state}},$$

with $w_{\mathrm{IC}} = w_{\mathrm{state}} = 10$. The initial-field term is a mean absolute error. The numerical-state term is the same relative ℓ_1 error used by TRAJECTORYOP, and the same metric is used for validation, checkpoint selection, and testing. The baseline objective contains no PDE, Rankine–Hugoniot, or other physics term. The initial-field term is evaluated at $t = 0$ on every input minibatch, whereas the numerical-state term uses only the nonzero time layers carrying evolution-state supervision labels: $t_1, \ldots, t_4$ in the sparse setting and $t_1, \ldots, t_8$ in the dense setting. Thus, $t = 0$ is supplied to all methods through the initial field, but it is not counted as an evolution-state supervision label. Every method operates on the initial state field, and the explicit interface variable enters through the TRAJECTORYOP architecture: FNO uses the full initial field, DeepONet and Shift-DeepONet use the fixed initial-field sensors, and TRAJECTORYOP uses the initial shock location estimated from the discrete field to parameterize its trajectory. The estimate is a deterministic preprocessing operation on data already supplied to the encoder, rather than an additional shock-location supervision label. The evolution-state supervision labels and the validation protocol remain identical within each benchmark.

All baseline models are optimized with Adam for 5000 epochs at learning rate 5×10^{-4}. In the one-dimensional sparse runs, the input and supervision-label minibatches both have size 16; in the dense runs they have sizes 16 and 4, respectively. The corresponding two-dimensional sizes are 4 and 4 for sparse supervision labels and 1 and 1 for dense supervision labels. Table 4 summarizes the one-dimensional Euler instantiation. Parameter counts are obtained by summing `numel()` over each model's registered parameters.

Table 4: Architectural and optimization settings for the one-dimensional Euler comparison. The two widths and depths for TRAJECTORYOP are listed as encoder/KAN values. The two depths for the operator baselines are listed as branch/trunk values. The loss weights are $w_{\mathrm{IC}} = w_{\mathrm{state}} = w_{\mathrm{RH}} = 10$ for every method, and $w_{\mathrm{PDE}} = 1$ for TRAJECTORYOP.

Method	Parameters	Width	Layers	Latent/basis	Training loss	Learning rate
TRAJECTORYOP	382,537	$48/33$	$3/3$	–	$w_{\mathrm{IC}}\mathcal{L}_{\mathrm{IC}} + w_{\mathrm{state}}\mathcal{L}_{\mathrm{state}} + w_{\mathrm{PDE}}\mathcal{L}_{\mathrm{PDE}} + w_{\mathrm{RH}}\mathcal{L}_{\mathrm{RH}}$	10^{-3}
FNO	550,019	64	4	–	$w_{\mathrm{IC}}\mathcal{L}_{\mathrm{IC}} + w_{\mathrm{state}}\mathcal{L}_{\mathrm{state}}$	5×10^{-4}
DeepONet	83,075	64	$4/4$	128	$w_{\mathrm{IC}}\mathcal{L}_{\mathrm{IC}} + w_{\mathrm{state}}\mathcal{L}_{\mathrm{state}}$	5×10^{-4}
Shift-DeepONet	190,339	64	$4/4$	128	$w_{\mathrm{IC}}\mathcal{L}_{\mathrm{IC}} + w_{\mathrm{state}}\mathcal{L}_{\mathrm{state}}$	5×10^{-4}

Table 5: Effect of the evolution-state supervision-label time range on TRAJECTORYOP. The case and spatial point counts are fixed at $n64/s256$ in one dimension and $n64/s16384$ in two dimensions. FNO is shown once with its dense evolution-state supervision-label set and is not retrained for the individual columns. All errors use the nine test times. Bold identifies the TRAJECTORYOP settings below dense FNO; otherwise the FNO result is bold. All listed experiments are complete.

	TRAJECTORYOP: varying supervision-label times				FNO
Benchmark	t_1	t_1–t_2	t_1–t_4	t_1–t_8	Dense
Burgers	**0.00704**	**0.00679**	**0.00769**	**0.00738**	0.00817
Shallow water	0.01464	0.01349	0.01117	0.01049	**0.00905**
1-D Euler	0.04004	0.03258	0.02943	0.02176	**0.01649**
2-D Euler shock–entropy	0.00687	0.00532	0.00321	**0.00133**	0.00188

A.3 SOLUTION PROFILES

Figure 4 shows TRAJECTORYOP trained with sparse evolution-state supervision labels and FNO trained with dense evolution-state supervision labels on one test case from each equation. The one-dimensional panels use the middle test case. The two-dimensional panel uses the case with the median final-time shock-location deformation. These cases were fixed without inspecting the prediction errors.

The columns show $t = 0$, $T/8$, $T/2$, and T. Sparse-supervision TRAJECTORYOP receives evolution-state supervision labels only through $T/2$, whereas dense-supervision FNO receives evolution-state supervision labels at all eight nonzero times. Color identifies the primitive variable in one dimension, and the two-dimensional rows show density from the numerical solution, TRAJECTORYOP, and FNO. At T, TRAJECTORYOP still predicts shock locations near the numerical reference and recovers the deformed two-dimensional shock beyond the time interval covered by its supervision labels.

A.4 EVOLUTION-STATE SUPERVISION-LABEL TIME RANGE

To isolate temporal coverage, we fix the number of cases with evolution-state supervision labels at 64 and use 256 supervised spatial points in one dimension and 16384 in two dimensions. The TRAJECTORYOP state loss uses evolution-state supervision evolution-state supervision labels from one of four prefixes: $\{t_1\}$, $\{t_1, t_2\}$, $\{t_1, \ldots, t_4\}$, or $\{t_1, \ldots, t_8\}$. The PDE and Rankine–Hugoniot residuals in TRAJECTORYOP are sampled over $[0, T]$ in all four runs. FNO, the strongest external baseline in Table 1, is not retrained for these settings. It retains $n64/s256/t_1$–t_8 in one dimension and $n64/s16384/t_1$–t_8 in two dimensions.

The fixed dense FNO error is lower than TRAJECTORYOP for every one-dimensional Euler prefix and for every shallow-water prefix. The TRAJECTORYOP error is below the dense FNO error for Burgers at all four prefixes and for the full time prefix on the two-dimensional benchmark. For one-dimensional Euler, extending the prefix from t_1 to t_8 reduces the error from 0.04004 to 0.02176, while the shallow-water error decreases from 0.01464 to 0.01049. On the two-dimensional benchmark, the corresponding errors are 0.00687, 0.00532, 0.00321, and 0.00133.

A.5 LOSS-COMPONENT ABLATIONS UNDER SPARSE SUPERVISION

We examine the roles of the initial-condition, regional PDE, and Rankine–Hugoniot constraints on the one- and two-dimensional Euler benchmarks. The reference is the sparse-supervision TRAJECTORYOP model in Table 1, with weights $(w_{\mathrm{IC}}, w_{\mathrm{state}}, w_{\mathrm{PDE}}, w_{\mathrm{RH}}) = (10, 10, 1, 10)$. Each ablation sets one of w_{IC}, w_{PDE}, or w_{RH} to zero, without rescaling the remaining terms. The evolution-state

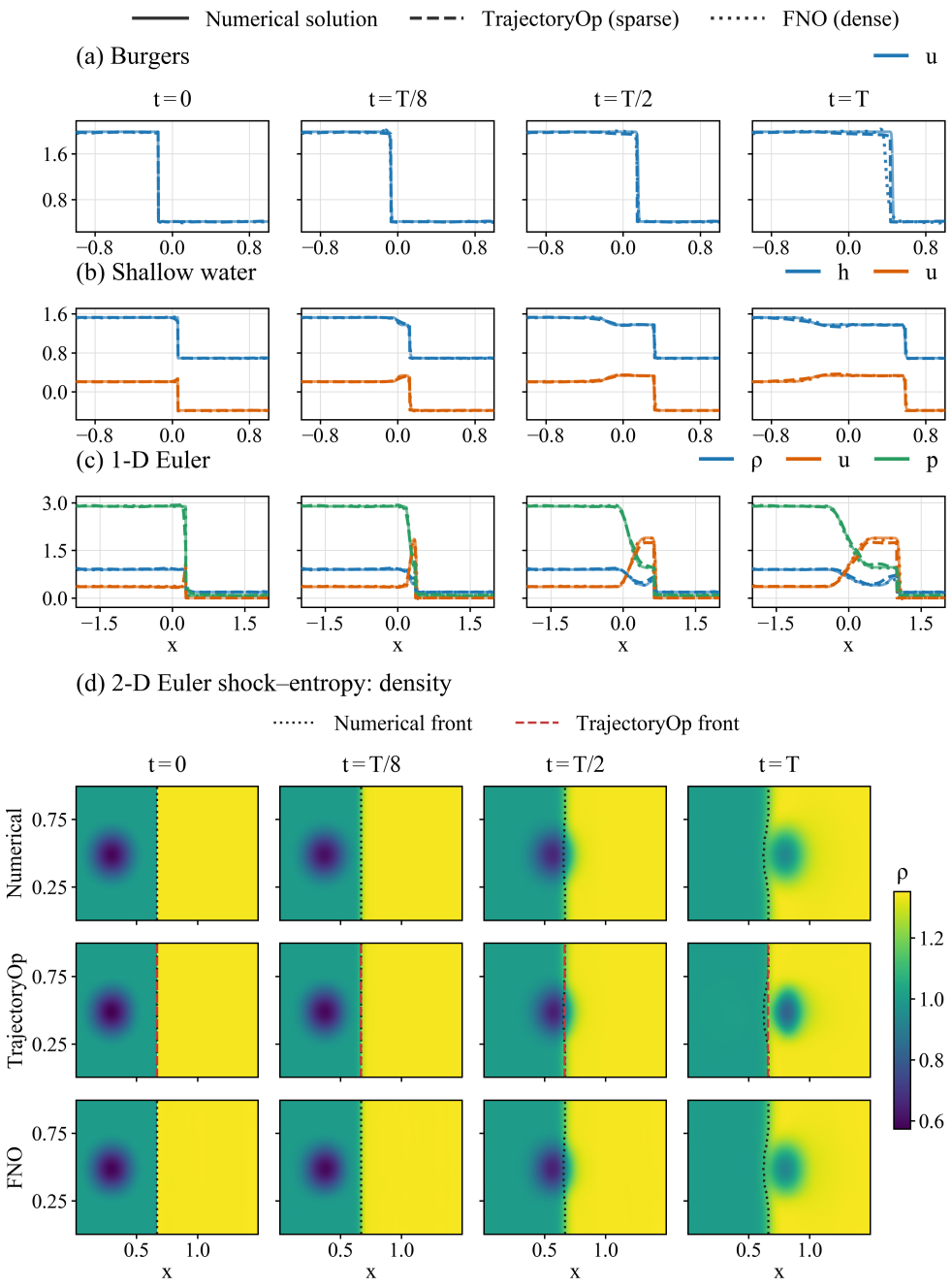


Figure 4: Numerical solutions and model predictions at four times. TRAJECTORYOP uses 3.125% of the evolution-state supervision-label budget used by FNO. In the two-dimensional panels, black dotted curves show the numerical shock and the red dashed curve shows the trajectory predicted by TRAJECTORYOP.

loss is retained in every run. Removing the initial-condition loss does not remove the initial field from the encoder or change the estimated initial shock location of the trajectory.

All variants use the same sparse evolution-state supervision labels as the reference: 16 cases, 64 spatial points per time in one dimension or 4096 in two dimensions, and times $t_1, \ldots, t_4$. The complete sets of 1024 and 512 training initial fields remain available. Network sizes, minibatch sizes, learning rates, and collocation budgets follow the settings above. Training lasts 5000 epochs with training and supervision-label-selection seeds both fixed to 1234; the cases carrying supervision labels, spatial indices, and validation cases are identical across variants within each benchmark. The

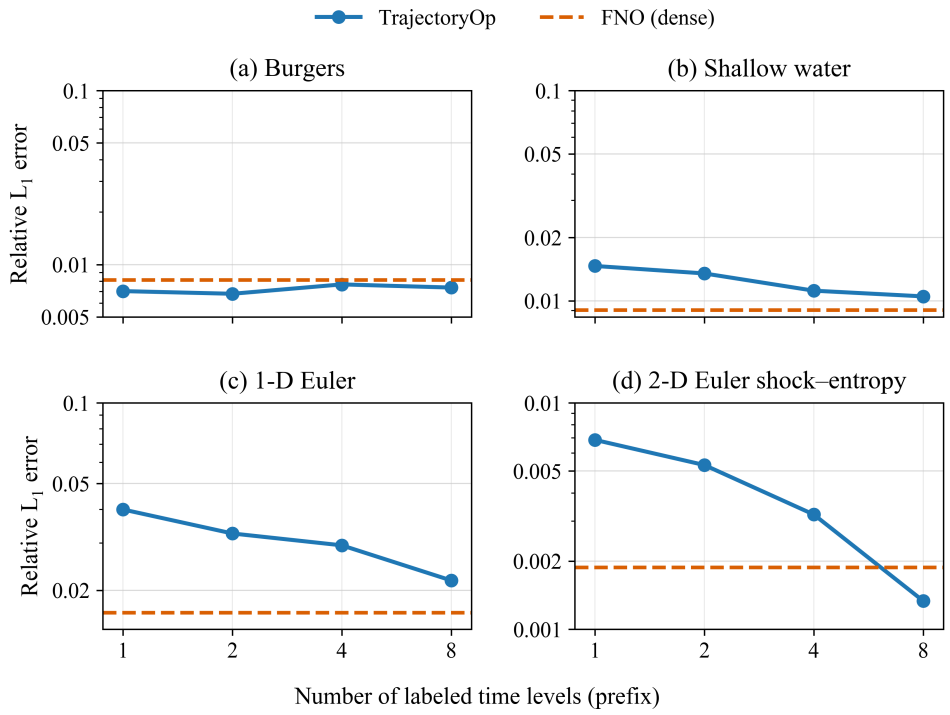


Figure 5: Space–time error as the evolution-state supervision-label prefix is extended from one to eight nonzero-time levels for TRAJECTORYOP. The dashed line is FNO trained with dense evolution-state supervision labels. The test error always includes all nine stored states.

Table 6: Space–time relative L_1 error for loss-component ablations with sparse evolution-state supervision labels, evaluated over all nine test times. Bold marks the lowest error in each column. All runs use seed 1234.

Training objective	1-D Euler	2-D Euler
All four terms	**0.03789**	**0.00324**
Without $\mathcal{L}_{\mathrm{IC}}$	0.15009	0.00375
Without $\mathcal{L}_{\mathrm{PDE}}$	0.10174	0.00417
Without $\mathcal{L}_{\mathrm{RH}}$	0.14267	0.00373

two-dimensional runs use query chunks of 32768 points for training and validation; this changes evaluation batching, not the supervision or collocation budgets. Checkpoint selection uses the same matched validation time coverage and spatial budget described above.

Table 6 reports the space–time relative L_1 error over all nine test times, the full numerical grid, and all test cases, using the same metric as the main experiments. Each entry comes from the checkpoint selected by the common validation criterion.

In one dimension, removing the initial-condition, PDE, or Rankine–Hugoniot term increases the space–time error to 3.96, 2.69, and 3.77 times the reference error, respectively. The initial-condition and interface terms have the largest effects in this single-seed comparison, while removing the regional PDE residual also substantially increases the error. Both the regional evolution residual and the interface jump constraint therefore contribute to the reported accuracy when evolution-state supervision labels are sparse.

In two dimensions, removing the PDE term raises the space–time error by 28.7%, while removing the initial-condition term increases it by 15.7%. Removing the Rankine–Hugoniot term increases the error by 15.1%. The relative impact of each term therefore differs between the benchmarks. In these single-seed runs, the four-term objective gives the lowest space–time error on both problems.

A.6 EXPLICIT-INTERFACE STRUCTURE ABLATION

We assess the explicit-interface representation by replacing the two interface-conditioned state decoders with a single state decoder and removing the interface-specific jump constraint. The initial-

Table 7: Joint ablation of the explicit-interface representation and its associated jump constraint under matched sparse supervision. Errors are space–time relative L_1 over all nine test times. The final column is the relative increase over TRAJECTORYOP.

Benchmark	TRAJECTORYOP	Single-state decoder	Increase
Burgers	**0.00789**	0.04750	502.2%
Shallow water	**0.01418**	0.02527	78.2%
1-D Euler	**0.03789**	0.04424	16.7%
2-D Euler shock–entropy	**0.00324**	0.00500	54.2%

field encoder, the initial-condition and relative state losses, the regional conservation residual, optimizer, and loss weights are otherwise retained. Because the comparison model has no predicted interface or left/right states, its PDE residual is evaluated at collocation points sampled over the full domain. We therefore treat this as a joint ablation of the interface representation and its associated jump constraint, rather than as an ablation of the decoder alone.

For each benchmark, both models use the same training and validation case indices, the same supervision-label spatial indices, and the same $n16/s64/t_1$–t_4 evolution-state supervision-label budget in one dimension or $n16/s4096/t_1$–t_4 in two dimensions. Checkpoint selection uses the matched validation supervision labels, and test error is computed over the complete test grid and all nine stored times. Table 7 shows that removing the interface representation and its associated jump constraint increases the space–time error on all four benchmarks. The increase is largest for Burgers, while it is more modest for one-dimensional Euler; the direction of the change is consistent across the tested systems.