Received: 18 April 2023 | Revised: 11 November 2023 | Accepted: 17 November 2023

DOI: 10.1002/nme.7406

RESEARCH ARTICLE

WILEY

RoeNet: Predicting discontinuity of hyperbolic systems from continuous data

Yunjin Tong[1] | **Shiying Xiong**[1,2] | **Xingzhe He**[1] | **Shuqi Yang**[1] | **Zhecheng Wang**[1] | **Rui Tao**[1] | **Runze Liu**[1] | **Bo Zhu**[1]

[1]Department of Computer Science, Dartmouth College, Hanover, New Hampshire, USA

[2]Department of Engineering Mechanics, School of Aeronautics and Astronautics, Zhejiang University, Hangzhou, China

Correspondence
Shiying Xiong, Department of Engineering Mechanics, School of Aeronautics and Astronautics, Zhejiang University, Hangzhou 310027, China.
Email: shiying.xiong@zju.edu.cn

Funding information
NSF, Grant/Award Numbers: 1919647, 2106733, 2144806, 2153560

Abstract

Predicting future discontinuous phenomena that are unobservable from training data sets has long been a challenging problem in scientific machine learning. We introduce a novel paradigm to predict the emergence and evolution of various discontinuities of hyperbolic partial differential equations (PDEs) based on given training data over a short window with limited discontinuity information. Our method is inspired by the classical Roe solver [P. L. Roe, J Comput Phys., vol. 43, 1981], a basic tool for simulating various hyperbolic PDEs in computational physics. By carefully designing the computing primitives, the data flow, and the novel pseudoinverse processing module, we enable our data-driven predictor to satisfy all the essential mathematical criteria of a Roe solver and hence deliver accurate predictions of hyperbolic PDEs. We demonstrate through various examples that our data-driven Roe predictor outperforms original human-designed Roe solver and deep neural networks with weak priors in terms of accuracy and robustness.

KEYWORDS

conservation law, machine learning, physics-informed neural network, Roe solver

1 | INTRODUCTION

Predicting the invisible future is the ultimate goal researchers strive to achieve in machine intelligence. To materialize this visionary goal into an algorithmic context, we propose to predict a dynamical system's future behavior patterns that are unseen in the training data set. Namely, the training and testing data sets exhibit salient differences with respect to their features and distributions in the solution space. Such examples are ubiquitous in natural and social sciences. For example, hydrodynamologists seek to predict the future occurrence of shock waves based on the observation of the current smooth fluid flow[1]; epidemiologists aim to predict the break-out time of an epidemic disease based on the collected data on an early stage.[2] More examples exist in weather forecast,[3] aircraft control,[4] traffic scheduling,[5] and economic crisis prediction.[6] From a mathematical perspective, we summarize these prediction problems as searching for the evolutionary solution of a partial differential equation (PDE) whose current observable status is significantly different from its targeted behaviors in the long future.

Hyperbolic PDE featured by discontinuities is commonly seen in many fields of physical science[7–9] and engineering.[10,11] Riemann problem,[12] as the simplest Hyperbolic PDE, is composed of a conservation equation with piecewise constant initial data which has only a single discontinuity in the domain of interest.[13] In scientific computing, Riemann solvers enable the hyperbolic PDEs to be seen as Riemann problems on local scales[14,15] by discretizing the hyperbolic PDEs

into grids.[16–18] Treating the hyperbolic PDE as multiple Riemann problems can be traced back to the work of Godunov[19] with many follow-ups works.[12,20] Roe solver, invented by Phil Roe in 1981, is a linearized approximate Riemann solver based on Godunov's method.[21]

In addition to the traditional numerical methods, using a deep learning network to approximate discontinuous functions has a theoretical foundation in various literature, such as studies on the Hölder spaces,[22] piecewise smooth functions,[23] linear estimators,[24] and higher adaptive and spatial anisotropic target functions.[25] With the above-mentioned theoretical cornerstone, physics-informed neural networks (PINNs) are proposed by Raissi et al. to provide data-driven solutions for nonlinear problems,[26] employing the well-known capacity of deep neural networks as universal function approximators.[27] Among its notable features, PINNs maintain symmetry, invariance, and conservation principles deriving from physical laws that govern observed data.[28] Michoski et al. show that without any regularization, irregular solutions to PDE can be captured.[29] Mao et al. further use PINNs to approximate solutions to high-speed flows by formulating the Euler equation with initial and boundary conditions into the loss function.[30] However, these works did not consider capturing the features that are not in the training set.

Devising a data-driven paradigm to address the prediction problem of unobservable pattern is difficult for two reasons. First, the target patterns do not exist in the training data set. In many scenarios, the training data merely covers a small region of the entire solution space.[31,32] Hence, the conventional use of deep neural networks falls short in the uncovering patterns. Second, there is a lack of accurate models to describe the evolution that connects the current, observable states, to the future, predictable targets. Mathematically, we typically need to use PDEs to model such an evolution.[33] However, in many cases, we do not have these PDEs in hand. Instead, only some first-principled inductive priors, such as conservation laws, are available to guide and evaluate the predictions on a high level. The limitation of observations and insufficiency of accurate models jointly make the prediction of unobservable patterns impossible.

We conduct a preliminary exploration to tackle the computational challenges of predicting unobservable patterns by proposing a novel approach that hybridizes model priors and learning paradigms on an architectural level. We describe our high-level design philosophy as "templaterizable prior embedding": analogous to a generic programming process, we use abstracted, inductive priors as a model template to uncover the unknown evolution mechanics that can be instantiated by an appropriate combination of data-driven primitive computing blocks. These computing primitives are embodied with specific mathematical roles in the template and designed as a set of neural network modules that can be trained in an end-to-end fashion to search for their optimal instantiations. Our learning paradigm which is facilitated with these prior-embedded modules can characterize dynamic evolution in which (i) the target patterns are invisible in training data sets, (ii) the observation window is partial and short, and (iii) no accurate PDE models connect the current and future.

We devise our "templaterizable priors" by creating a template model based on a classical Riemann solver, which is the Roe solver.[34] In the following sections, we will demonstrate how we take advantage of the mathematical principles to devise a novel learning model (named RoeNet) by embedding the templaterizable Roe modules as a set of data-driven computing primitives embodied with strong mathematical priors. There are two significant differences between RoeNet and Roe solver. First, RoeNet replaces the Roe matrix with a neural network that is mainly composed of ResBlocks. Second, RoeNet replaces the inverse matrix with the pseudoinverse matrix. Although RoeNet is designed based on an approximate Riemann solver, the introduction of the pseudoinverse gives it a higher parameter dimensions beyond the number of equation components. Therefore, the network has a different solution space from the conventional approximation or exact Riemann solver.

At the same time, navigating the solution space of a deep neural network can be particularly challenging when aiming to continuously track the evolution of intermittent solutions. Unlike these traditional networks, RoeNet offers a more constrained and hence manageable solution space. This advantageous feature obviates the need for employing exceedingly large neural networks and datasets to accurately delineate the solution space. Additionally, RoeNet incorporates an embedded Roe solver, which serves as a specialized mechanism for addressing the conservation challenges inherently associated with hyperbolic equations. The integration of this solver not only enhances the network's overall problem-solving capacity but also renders it more adept at tackling the specific nuances related to the conservation properties of hyperbolic systems. Consequently, RoeNet emerges as a highly efficacious computational tool for solving complex hyperbolic PDEs, offering both computational efficiency and solution accuracy.

RoeNet can be regarded as a neural network that predicts approximate continuous or discontinuous solutions of hyperbolic PDEs. The prediction of discontinuous solutions by RoeNet is manifested in two aspects. First, RoeNet can predict the evolution of discontinuous solutions, that is, it can predict the discontinuous jumps evolving from smooth regions in the dataset. In addition, RoeNet can predict the future discontinuities based only on short-term continuous training data. We examine RoeNet's ability by conducting long-term predictions of discontinuity for a broad range of hyperbolic

TABLE 1 Notations used. The outputs of $\boldsymbol{L}_\theta$ and $\boldsymbol{\Lambda}_\phi$ are a $N_h \times N_c$ matrix and a $N_h \times N_h$ diagonal matrix, respectively.

Notation	Meaning
N_c	Number of components of an evolutionary variable
N_d	Dimension of spatial coordinates
N_g	Number of spatial grid points
N_h	Hidden dimension in RoeNet
$\mathbf{u} = [u^{(1)}, u^{(2)}, \dots, u^{(N_c)}]^T$	Conserved quantity with N_c components
$\mathbf{F} = [F^{(1)}, F^{(2)}, \dots, F^{(N_c)}]^T$	Flux function with N_c components
$\mathbf{x} = (x_1, x_2, \dots, x_{N_d})$	N_d-dimensional spatial coordinates
$\mathbf{L}$	$N_c \times N_c$ invertible matrix
$\boldsymbol{\Lambda} = \mathrm{diag}(\Lambda_1, \dots, \Lambda_{N_c})$	$N_c \times N_c$ diagonal matrix
$\lvert\boldsymbol{\Lambda}\rvert = \mathrm{diag}(\lvert\Lambda_1\rvert, \dots, \lvert\Lambda_{N_c}\rvert)$	Absolute value of $\boldsymbol{\Lambda}$
$\tilde{\mathbf{A}} = \mathbf{L}^{-1}\boldsymbol{\Lambda}\mathbf{L}$	Diagonalization of Roe matrix
$\lvert\tilde{\mathbf{A}}\rvert = \mathbf{L}^{-1}\lvert\boldsymbol{\Lambda}\rvert\mathbf{L}$	Absolute value of $\tilde{\mathbf{A}}$
$\mathbf{L}_\theta$	Neural network counterpart of $\mathbf{L}$
$\boldsymbol{\Lambda}_\phi$	Neural network counterpart of $\boldsymbol{\Lambda}$
$\mathbf{L}_\theta^+ = (\mathbf{L}_\theta^T\mathbf{L}_\theta)^{-1}\mathbf{L}_\theta^T$	Neural network counterpart of $\mathbf{L}^{-1}$

PDEs. Outperforming both human-designed numerical schemes and state-of-the-art neural networks, the results demonstrate the accuracy, robustness, and outstanding predictive capability of our RoeNet framework in processing invisible phenomena.

The outline of this article is as follows. In Section 2 we introduce the hyperbolic conservation law and an approximate Riemann solver, the Roe solver. In Section 3, we introduce the design ideas and the implementation details of RoeNet based on the Roe solver and pseudoinverse embedding. Next, Section 4 describes the numerical results, including linear and nonlinear, one-dimensional and multidimensional, one-component and multi-component equations. In Section 5, we validate our design of RoeNet architecture and numerical settings in the training process by a series of ablation experiments. At the end of this section, we compare our method with PINNs using DeepXDE library.[35] Lastly, conclusions and potential directions of our future research are presented in Section 6. In addition, the main notations used in this article are summarized in Table 1.

2 | HYPERBOLIC CONSERVATION LAW AND ROE SOLVER

2.1 | Hyperbolic conservation law

In continuum mechanics, a one-dimensional hyperbolic conservation law is a first-order quasilinear hyperbolic PDE

$$\frac{\partial \boldsymbol{u}}{\partial t} + \frac{\partial \boldsymbol{F}(\boldsymbol{u})}{\partial x} = 0, \tag{1}$$

with an initial condition

$$\boldsymbol{u}(t = t_0, x) = \boldsymbol{u}_0(x), \tag{2}$$

and a proper boundary condition. Here the N_c-component vector $\boldsymbol{u} = [u^{(1)}, u^{(2)}, \dots, u^{(N_c)}]^T$ is the conserved quantity, $t \in T = [t_0, t_1]$ denotes the time variable, x denotes the spatial coordinate in a computational domain Ω, and $\boldsymbol{F} = [F^{(1)}, F^{(2)}, \dots, F^{(N_c)}]^T$ is a N_c-component flux function. The conservation laws described by (1) are fundamental in continuum mechanics, such as mass conservation, momentum conservation, and energy conservation in fluid mechanics.[36]

Equation (1) can also be expressed in a weak form, which extends the class of admissible solutions to include discontinuous solutions. Specifically, by defining an arbitrary test function $\phi(t,x)$ that is continuously differentiable both in time and space with compact support, and integrating (1)$\times\phi$ in the space-time domain $T\times\Omega$, the weak form of (1) is derived as follows:

$$\int_{T\times\Omega}\left(\boldsymbol{u}\frac{\partial\phi}{\partial t}+\boldsymbol{F}\frac{\partial\phi}{\partial x}\right)\mathrm{d}t\mathrm{d}x=0. \tag{3}$$

We remark that, with generalized Stokes theorem, all the partial derivatives of $\boldsymbol{u}$ and $\boldsymbol{F}$ in (1) have been passed on to the test function ϕ in (3), which with the former hypothesis is sufficiently smooth to admit these derivatives.[37] In the absence of ambiguity, we refer to the solution of (1) below as a weak solution that satisfies (3).

In addition, (1) can be written in a high-dimensional form

$$\frac{\partial\boldsymbol{u}}{\partial t}+\sum_{i=1}^{N_d}\frac{\partial\boldsymbol{F}_i(\boldsymbol{u})}{\partial x_i}=\boldsymbol{0}, \tag{4}$$

where $x_1, x_2, \ldots, x_{N_d}$ denote the N_d-dimensional spatial coordinates. Since every dimension in the second term of (4), namely $\partial\boldsymbol{F}_i(\boldsymbol{u})/\partial x_i$, has the same form $\partial\boldsymbol{F}(\boldsymbol{u})/\partial x$ as the second term of (1), (4) can be easily solved if given the solution of (1). Thus, in the rest of this article, we will only discuss the numerical method to solve (1).

2.2 | Roe solver

Philip L. Roe proposed an approximated Riemann solver based on the Godunov scheme[34] that constructs an estimation for the intercell numerical flux of $\boldsymbol{F}$ in (1) on the interface of two neighboring computational cells in a discretized space-time computational domain.[34] In particular, the Roe solver discretizes (1) as follows:

$$\boldsymbol{u}_j^{n+1}=\boldsymbol{u}_j^n-\lambda_r\left(\hat{\boldsymbol{F}}_{j+\frac{1}{2}}^n-\hat{\boldsymbol{F}}_{j-\frac{1}{2}}^n\right), \tag{5}$$

where $\lambda_r=\Delta t/\Delta x$ is the ratio of the temporal step size Δt to the spatial step size $\Delta x, j=1,\ldots,N_g$ is the grid node index, and

$$\hat{\boldsymbol{F}}_{j+\frac{1}{2}}^n=\hat{\boldsymbol{F}}(\boldsymbol{u}_j^n,\boldsymbol{u}_{j+1}^n) \tag{6}$$

with

$$\hat{\boldsymbol{F}}(\boldsymbol{u},\boldsymbol{v})=\frac{1}{2}\left[\boldsymbol{F}(\boldsymbol{u})+\boldsymbol{F}(\boldsymbol{v})-|\tilde{\boldsymbol{A}}(\boldsymbol{u},\boldsymbol{v})|(\boldsymbol{v}-\boldsymbol{u})\right]. \tag{7}$$

Here, Roe matrix $\tilde{\boldsymbol{A}}$ that is assumed constant between two cells and must obey the following Roe conditions:

1. Matrix $\tilde{\boldsymbol{A}}$ is a diagonalizable matrix with real eigenvalues, that is, matrix $\tilde{\boldsymbol{A}}(\boldsymbol{u},\boldsymbol{v})$ can be diagonalized as

$$\tilde{\boldsymbol{A}}=\boldsymbol{L}^{-1}\boldsymbol{\Lambda}\boldsymbol{L} \tag{8}$$

   with an invertible matrix $\boldsymbol{L}$ and a diagonal matrix $\boldsymbol{\Lambda}=\mathrm{diag}(\Lambda_1,\ldots,\Lambda_{N_c})$.
2. Matrix $\tilde{\boldsymbol{A}}$ is consistent with an exact Jacobian, that is,

$$\lim_{\boldsymbol{u}_j,\boldsymbol{u}_{j+1}\to\boldsymbol{u}}\tilde{\boldsymbol{A}}(\boldsymbol{u}_j,\boldsymbol{u}_{j+1})=\frac{\partial\boldsymbol{F}(\boldsymbol{u})}{\partial\boldsymbol{u}}. \tag{9}$$

3. Physical quantity $\boldsymbol{u}$ is conserved on the interface between two computational cells as follows:

$$\boldsymbol{F}_{j+1}-\boldsymbol{F}_j=\tilde{\boldsymbol{A}}(\boldsymbol{u}_{j+1}-\boldsymbol{u}_j). \tag{10}$$

We denote the absolute value of $\tilde{\boldsymbol{A}}(\boldsymbol{u},\boldsymbol{v})$ as

$$|\tilde{\boldsymbol{A}}| = \boldsymbol{L}^{-1}|\boldsymbol{\Lambda}|\boldsymbol{L}, \tag{11}$$

where $|\boldsymbol{\Lambda}| = \text{diag}(|\Lambda_1|, \dots, |\Lambda_{N_c}|)$ is the absolute value of $\boldsymbol{\Lambda}$. Substituting (6), (7) and (11) into (5) along with the third Roe condition (10) yields

$$\begin{aligned}\boldsymbol{u}_j^{n+1} = \boldsymbol{u}_j^n - \frac{1}{2}\lambda_r[&(\boldsymbol{L}^n_{j+\frac{1}{2}})^{-1}(\boldsymbol{\Lambda}^n_{j+\frac{1}{2}} - |\boldsymbol{\Lambda}^n_{j+\frac{1}{2}}|)\boldsymbol{L}^n_{j+\frac{1}{2}}(\boldsymbol{u}^n_{j+1} - \boldsymbol{u}^n_j)\\ &+ (\boldsymbol{L}^n_{j-\frac{1}{2}})^{-1}(\boldsymbol{\Lambda}^n_{j-\frac{1}{2}} + |\boldsymbol{\Lambda}^n_{j-\frac{1}{2}}|)\boldsymbol{L}^n_{j-\frac{1}{2}}(\boldsymbol{u}^n_j - \boldsymbol{u}^n_{j-1})]\end{aligned} \tag{12}$$

with

$$\boldsymbol{L}^n_{j+\frac{1}{2}} = \boldsymbol{L}(\boldsymbol{u}^n_j, \boldsymbol{u}^n_{j+1}), \quad \boldsymbol{\Lambda}^n_{j+\frac{1}{2}} = \boldsymbol{\Lambda}(\boldsymbol{u}^n_j, \boldsymbol{u}^n_{j+1}). \tag{13}$$

Equation (12) serves as a template of evolution from $\boldsymbol{u}_j^n$ to $\boldsymbol{u}_j^{n+1}$ in Roe solver.

The key to design an effective Roe solver is to find the Roe matrix $\tilde{\boldsymbol{A}}$ that satisfies the three Roe conditions. In order to construct a Roe matrix $\tilde{\boldsymbol{A}}$ in (8), Roe solver utilizes an analytical approach to solve $\boldsymbol{L}$ and $\boldsymbol{\Lambda}$ based on $\boldsymbol{F}(\boldsymbol{u})$. The Roe matrix is then plugged into (12) to ultimately solve for $\boldsymbol{u}$ in (1). The Roe solver linearizes Riemann problems, and such linearization recognizes the problem's nonlinear jumps, while remaining computationally efficient.

3 | ROENET

In this section, we introduce our design of the Roe template with pseudoinverse embedding, which accommodates the data processing and training over the entire learning pipeline. In particular, we present our basic ideas in Section 3.1, a detailed description of our network architecture in Section 3.2, and a summary of training settings in Section 3.3.

3.1 | Roe template with pseudoinverse embedding

Without a given $\boldsymbol{F}$, we learn the weak solution of (1) using a neural network that incorporates the framework of a Roe solver. For time integration of $\boldsymbol{u}$ in (12), we need to construct the matrix functions $\boldsymbol{L}$ and $\boldsymbol{\Lambda}$. Since learning a tiny parameter space is impractical, using neural networks to approximate $\boldsymbol{L}$ and $\boldsymbol{\Lambda}$ directly in (13) is ineffective given that the number of learnable parameters is limited by the number of components N_c of $\boldsymbol{u}$. To enhance the expressiveness of our model, we use neural network $\boldsymbol{L}_\theta$ and $\boldsymbol{\Lambda}_\phi$ to replace $\boldsymbol{L}$ and $\boldsymbol{\Lambda}$ in (13) respectively. Similar to (13), the inputs to $\boldsymbol{L}_\theta$ and $\boldsymbol{\Lambda}_\phi$ remains the same as $(\boldsymbol{u}^n_j, \boldsymbol{u}^n_{j+1})$. However, the outputs of $\boldsymbol{L}_\theta$ and $\boldsymbol{\Lambda}_\phi$ are now a $N_h \times N_c$ matrix and a $N_h \times N_h$ diagonal matrix respectively, where the positive integer N_h is a hidden dimension. Furthermore, we introduce the concept of pseudoinverses by replacing $\boldsymbol{L}^{-1}$ with

$$\boldsymbol{L}^+_\theta = (\boldsymbol{L}^T_\theta \boldsymbol{L}_\theta)^{-1}\boldsymbol{L}^T_\theta. \tag{14}$$

Here, the transpose and inverse operations are applied to the output matrix, that is

$$\boldsymbol{L}^+_\theta(\boldsymbol{u}^n_j, \boldsymbol{u}^n_{j+1}) = [\boldsymbol{L}_\theta(\boldsymbol{u}^n_j, \boldsymbol{u}^n_{j+1})^T \boldsymbol{L}_\theta(\boldsymbol{u}^n_j, \boldsymbol{u}^n_{j+1})]^{-1}\boldsymbol{L}^T_\theta(\boldsymbol{u}^n_j, \boldsymbol{u}^n_{j+1}). \tag{15}$$

Substituting $\boldsymbol{L}_\theta$, $\boldsymbol{\Lambda}_\phi$, and (14) into (12) and (13) yields

$$\begin{aligned}\boldsymbol{u}_j^{n+1} = \boldsymbol{u}_j^n &- \frac{1}{2}\lambda_r(\boldsymbol{L}^n_{j+\frac{1}{2},\theta})^+(\boldsymbol{\Lambda}^n_{j+\frac{1}{2},\phi} - |\boldsymbol{\Lambda}^n_{j+\frac{1}{2},\phi}|)\boldsymbol{L}^n_{j+\frac{1}{2},\theta}(\boldsymbol{u}^n_{j+1} - \boldsymbol{u}^n_j)\\ &- \frac{1}{2}\lambda_r(\boldsymbol{L}^n_{j-\frac{1}{2},\theta})^+(\boldsymbol{\Lambda}^n_{j-\frac{1}{2},\phi} + |\boldsymbol{\Lambda}^n_{j-\frac{1}{2},\phi}|)\boldsymbol{L}^n_{j-\frac{1}{2},\theta}(\boldsymbol{u}^n_j - \boldsymbol{u}^n_{j-1})\end{aligned} \tag{16}$$

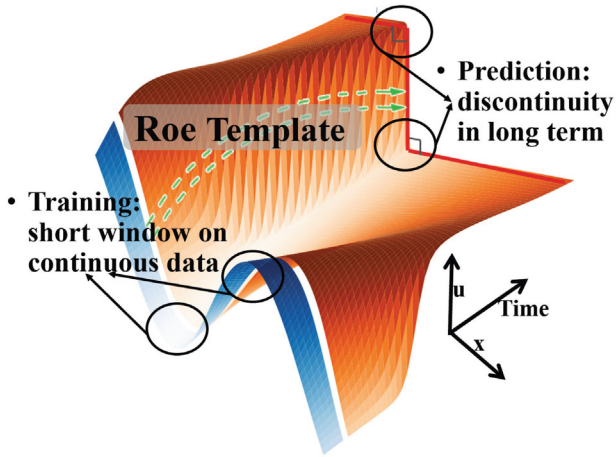

FIGURE 1 Schematic diagram of RoeNet to predict the future discontinuity from smooth observations. The blue band shows the distribution of the training set with respect to time, and the training set does not necessarily contain discontinuous solutions to the equations. Meanwhile, the orange band represents the solutions predicted with RoeNet, which may contain discontinuous solutions.

with

$$\boldsymbol{L}^n_{j+\frac{1}{2},\theta} = \boldsymbol{L}_\theta(\boldsymbol{u}^n_j, \boldsymbol{u}^n_{j+1}), \qquad \boldsymbol{\Lambda}^n_{j+\frac{1}{2},\phi} = \boldsymbol{\Lambda}_\phi(\boldsymbol{u}^n_j, \boldsymbol{u}^n_{j+1}). \tag{17}$$

Equation (16) serves as our template to evolve the system's states from $\boldsymbol{u}^n_j$ to $\boldsymbol{u}^{n+1}_j$ in RoeNet.

Figure 1 plots the schematic diagram of RoeNet to predict future discontinuity from smooth observations. We remark that for hyperbolic conversation law with discontinuous solutions, RoeNet can predict the long-term solutions that are fully or partially discontinuous when the given training data over a short window with limited information on discontinuity.

3.2 | Neural network architecture

Figure 2 shows an overview of our neural network architecture. In summary, RoeNet consists of $\boldsymbol{L}_\theta$ and $\boldsymbol{\Lambda}_\phi$, two networks embedded in (16) to serve as our template to evolve the system's states from $\boldsymbol{u}^n_j$ to $\boldsymbol{u}^{n+1}_j$. Specifically, the network in Figure 2 contains two parts, each consists of a $\boldsymbol{L}_\theta$ and a $\boldsymbol{\Lambda}_\phi$. The first part takes $\boldsymbol{u}^n_{j-1}$ and $\boldsymbol{u}^n_j$ as input of both $\boldsymbol{L}_\theta$ and $\boldsymbol{\Lambda}_\phi$ and outputs $\boldsymbol{L}_{j-\frac{1}{2},\theta}$ through $\boldsymbol{L}_\theta$ and $\boldsymbol{\Lambda}_{j-\frac{1}{2},\phi}$ through $\boldsymbol{\Lambda}_\phi$. The input $\boldsymbol{u}^n_{j-1}$ and $\boldsymbol{u}^n_j$ is a vector $[\boldsymbol{u}^{n,(1)}_{j-1}, \dots, \boldsymbol{u}^{n,(N_c)}_{j-1}; \boldsymbol{u}^{n,(1)}_j, \dots, \boldsymbol{u}^{n,(N_c)}_j]$ of length $2N_c$ with N_c components. The output matrix $L_{j-\frac{1}{2},\theta}$ is of size $(N_c \times N_h)$, and the other output matrix $\boldsymbol{\Lambda}_{j-\frac{1}{2},\phi}$ is a diagonal matrix of size $(N_h \times N_h)$. The second part takes $\boldsymbol{u}^n_j$ and $\boldsymbol{u}^n_{j+1}$ as the input for both $\boldsymbol{L}_\theta$ and $\boldsymbol{\Lambda}_\phi$ and outputs $\boldsymbol{L}_{j+\frac{1}{2},\theta}$ through $\boldsymbol{L}_\theta$ and $\boldsymbol{\Lambda}_{j+\frac{1}{2},\phi}$ through $\boldsymbol{\Lambda}_\phi$. The input $\boldsymbol{u}^n_j$ and $\boldsymbol{u}^n_{j+1}$ is a vector $[\boldsymbol{u}^{n,(1)}_j, \dots, \boldsymbol{u}^{n,(N_c)}_j; \boldsymbol{u}^{n,(1)}_{j+1}, \dots, \boldsymbol{u}^{n,(N_c)}_{j+1}]$ of length $2N_c$. The output matrices $\boldsymbol{L}_{j+\frac{1}{2},\theta}$ and $\boldsymbol{\Lambda}_{j+\frac{1}{2},\phi}$ take the same form as the output matrices in the first part. Given the four output matrices $\boldsymbol{L}_{j-\frac{1}{2},\theta}$, $\boldsymbol{\Lambda}_{j-\frac{1}{2},\phi}$, $\boldsymbol{L}_{j+\frac{1}{2},\theta}$, and $\boldsymbol{\Lambda}_{j+\frac{1}{2},\phi}$, we combine them through (16) and (17) to obtain $\boldsymbol{u}^{n+1}_j$.

Networks $\boldsymbol{L}_\theta$ and $\boldsymbol{\Lambda}_\phi$ both consist of a chain of ResBlocks[38] with a linear layer of size $N_h \times N_c$ and N_h at the end, respectively. The ResBlock architecture comprises two convolutional layers and one ReLU layer, meaning that the ResBlock we employ incorporates ReLU activation functions. In our future work, we will investigate the potential benefits of adaptive activation functions, which could enhance learning capabilities by enabling faster convergence rates during early training and leading to more accurate solutions.[39,40]

ResNet has been proven in numerous research studies to be a neural network architecture highly suitable for deep learning and computer vision. It offers distinctive advantages in mitigating problems like gradient vanishing during network training. Our RoeNet, tailored to predict the evolution of PDEs on a uniform grid, exhibits increasing network depth as the number of time iterations grows. This network design closely mirrors the deep neural networks commonly employed in computer vision, making ResBlock an ideal selection for building our network.

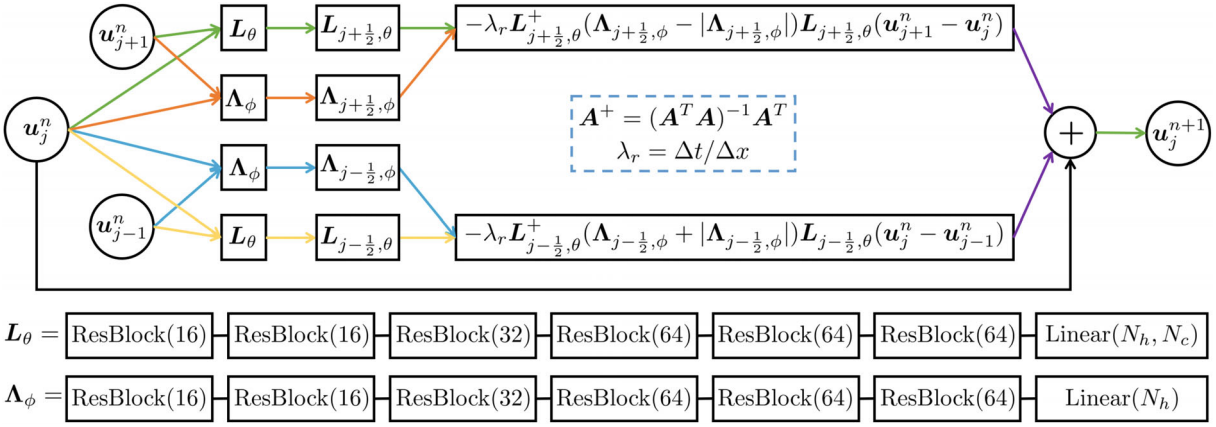

FIGURE 2 The architecture of the neural network that evolves the system's states from $\boldsymbol{u}_j^n$ to $\boldsymbol{u}_j^{n+1}$ in RoeNet. This network takes the current conserved quantity $\boldsymbol{u}_j^n$ and its direct neighbors, $\boldsymbol{u}_{j-1}^n$ and $\boldsymbol{u}_{j+1}^n$, as the inputs and outputs the conserved quantity $\boldsymbol{u}_j^{n+1}$ of the next time step. The ResBlock has the same architecture as in,[38] only with the 2D convolution layers replaced by linear layers. The number in the parentheses is the dimension of each Resblock output.

The N_h learned parameters by $\boldsymbol{\Lambda}_\phi$ is then transferred into a diagonal matrix of $N_h \times N_h$ with the learned parameters as its diagonal. The ResBlock has the same architecture as in,[38] only with the 2D convolution layers replaced by linear layers. Note that the number in the parentheses is the dimension of the output of each ResBlock, and the computation procedure for grid cell j is applied to all grid cells. Since the computation of each node is independent of other cells except the adjacent cells, we could train them in parallel to achieve high efficiency.

In addition, we implement two ways of padding to address different boundary conditions. For periodic boundary conditions, we use the periodic padding, e.g., if $j = 0$, then $\boldsymbol{u}_{j-1} = \boldsymbol{u}_{N_g}$, where N_g is the number of the grid node. For Neumann boundary conditions, we use the replicate padding, for example, if $j = 0$, then $\boldsymbol{u}_{j-1} = \boldsymbol{u}_0$.

By introducing a hidden dimension N_h, we have increased the number of network parameters and enhanced the network's expressive capacity. Simultaneously, due to the expansion of the parameter space, there may exist multiple numerical optimal solutions during the training process. However, by utilizing a regularized loss function for training, it is possible to make the network parameters converge to a local optimal solution. It is worth noting that our objective is to employ the network to fit the evolution of PDEs over time and space, without requiring the network parameters to have a unique solution.

3.3 | Training settings

Depending on the experiments, we construct data sets with analytical or numerical solutions calculated with a high-resolution finite difference method. We then split the training data sets and the validation data sets with a ratio of 9 : 1. The physical quantities solved in our experiments are of order $O(1)$ and do not require a normalization step. We train the network with a training set defined on a time span of T_{train} and then predict target values defined on a time span of $T_{predict}$, where $T_{predict} > T_{train}$ and $T_{predict}$ starts no earlier than the start of T_{train}. For all experiments, we use the Adam optimizer[41] with a learning rate of γ in Table 2, which decays with a multiplicative factor of 0.9 for every $5 - 20$ epochs. In all our experiments, we employ the Adam optimizer. This choice is motivated by its ability to adapt learning rates for each parameter by considering the gradient history, leading to faster and more accurate convergence when compared to fixed learning rate methods. With a batch size of $8 - 32$, all models are trained for 100 epochs to guarantee convergence. Note that we can trade training time for training accuracy, which means iteration of more epochs will result in higher accuracy.

Algorithm 1 summarizes the recursive relation from the input layer

$$\boldsymbol{u}(t = 0) = [\boldsymbol{u}_1(t = 0), \ \dots \ , \boldsymbol{u}_{N_g}(t = 0)] \tag{18}$$

to the output layer

$$\hat{\boldsymbol{u}}(t = T_{span}) = [\hat{\boldsymbol{u}}_1(t = T_{span}), \ \dots \ , \hat{\boldsymbol{u}}_{N_g}(t = T_{span})], \tag{19}$$

TABLE 2 Experimental set-up.

	1C Linear	Burgers	3C Linear	Sod Tube	2D Linear	2D Nonliear
Boundary condition	Periodic	Periodic	Neumann	Neumann	Periodic	Periodic
Time step Δt	0.02	0.001	0.001	0.001	0.02	0.02
Space step Δx	0.01	0.01	0.005	0.005	0.01	0.02
Training time span	0.04	0.002	0.06	0.06	0.1	0.02
Predicting time span >	2	0.5	0.3	0.1	1	1
Data set samples	500	100	1000	2000	200	500
Data set generation	Analytical	Numerical	Analytical	Analytical	Analytical	Numerical
Components number N_c	1	1	3	3	1	1
Hidden dimension N_h	1	4	16	64	8	8

Algorithm 1. Recursive relation from the input layer to the output layer in RoeNet. Here, $\boldsymbol{u}_j,\ j = 1, 2, \dots, N_g$ represents discretized points $\boldsymbol{u}$ in spatial coordinate

Input: $\boldsymbol{u}_j(t=0), j = 1, 2, \dots, N_g$, T_{span}, Δt, Δx, $\boldsymbol{L}_\theta$, $\boldsymbol{\Lambda}_\phi$
Output: $\hat{\boldsymbol{u}}_j(t = T_{span}) = \boldsymbol{u}_j^{N_t}$
$N_t = \text{floor}(T_{span}/\Delta t)$ $\lambda_r = \Delta t/\Delta x$ $\boldsymbol{u}_j^0 = \boldsymbol{u}_j(t=0), j = 1, 2, \dots, N_g$ **for** $n = 0 \to N_{t-1}$ **do**
  Calculate $\boldsymbol{L}^n_{j\pm\frac{1}{2},\theta}$, $\boldsymbol{\Lambda}^n_{j\pm\frac{1}{2},\phi}$ by substituting $\boldsymbol{u}_j^n, j = 1, 2, \dots, N_g$, $\boldsymbol{L}_\theta$, and $\boldsymbol{\Lambda}_\phi$ into (17) Calculate $\boldsymbol{u}_j^{n+1}, j = 1, 2, \dots, N_g$ by substituting $\boldsymbol{u}_j^n, \boldsymbol{L}^n_{j\pm\frac{1}{2},\theta}, \boldsymbol{\Lambda}^n_{j\pm\frac{1}{2},\phi}$, and λ_r into (16)
end

for each time step in RoeNet. Here N_g is the spatial grid size and T_{span} is the time span T_{train} or $T_{predict}$. As described in Alg. 1, feeding $\boldsymbol{u}(t=0)$, $T_{span} = T_{train}$, temporal step Δt, spatial step Δx, and the constructed networks $\boldsymbol{L}_\theta$ and $\boldsymbol{\Lambda}_\phi$ into RoeNet, we could get predicted $\hat{\boldsymbol{u}}(t = T_{train})$. Then, we choose the mean-squared error (MSE) as our loss function.

$$\mathcal{L}_{RoeNet} = \|\boldsymbol{u}(t = T_{train}) - \hat{\boldsymbol{u}}(t = T_{train})\|_{MSE}. \tag{20}$$

4 | EXAMPLES

We validate the ability of our model to solve different types of hyperbolic PDEs and predict the discontinuity in solutions with partially or entirely smooth training data. The details of parameters used and important quantities about hidden layers can be found in Table 2. Given that there is no analytical solution with explicit expression to the inviscid Burger's equation, we use the second-order central difference to generate data sets for the inviscid Burgers' equation. Since second-order central difference will cause numerical instability, we only use the data collected from a short time span at the beginning of the simulation, where the solution remains numerically correct, to train our model. For all the problems, we aim to solve spatial span from −0.5 to 0.5.

4.1 | A simple example

Taking a linear hyperbolic PDE with one component (1C Linear in Table 2)

$$\begin{cases} F = u, \\ u(t=0,x) = e^{-300x^2} \end{cases} \tag{21}$$

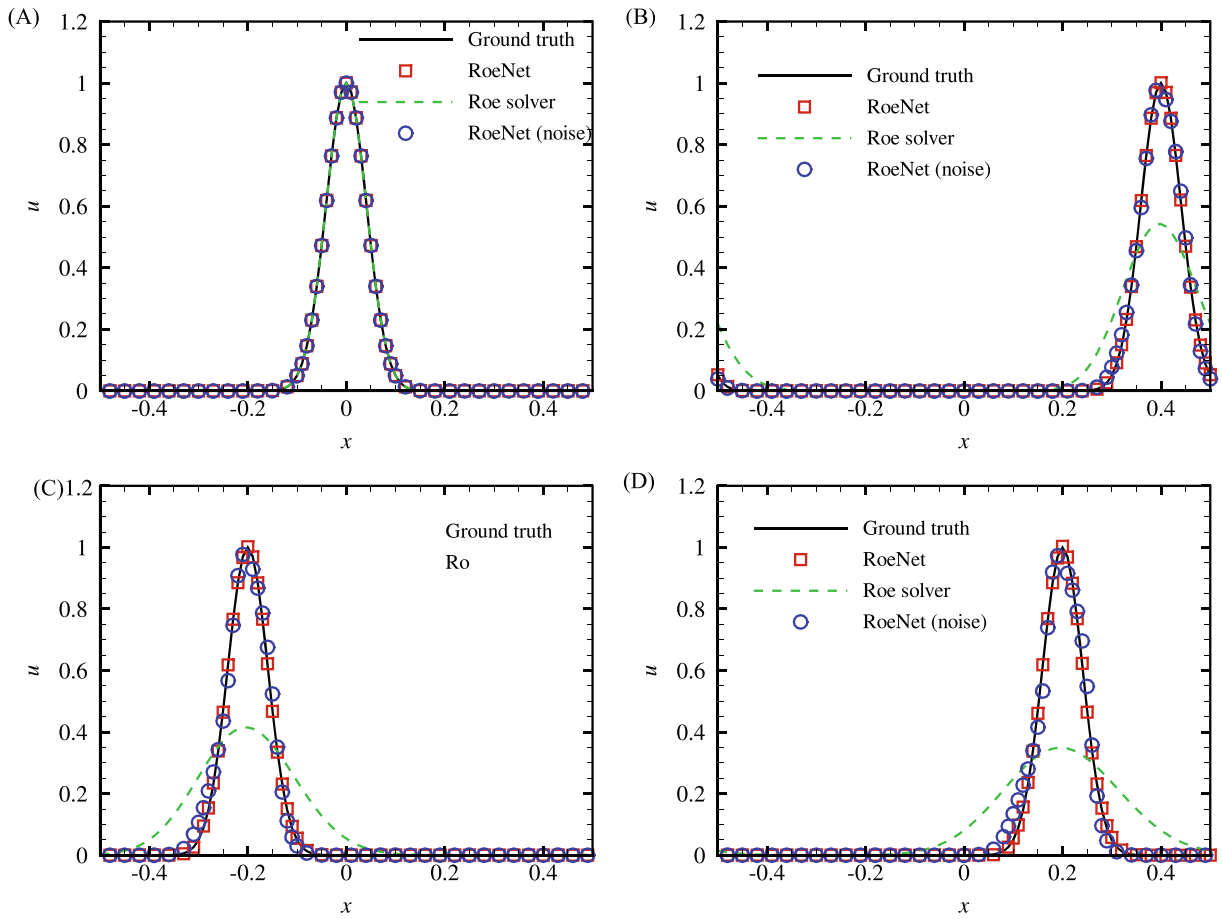

FIGURE 3 Comparison of RoeNet and Roe solver for solving a one component linear hyperbolic PDE (1C Linear in Table 2). (A) $t = 0$, (B) $t = 0.4$, (C) $t = 0.8$, (D) $t = 1.2$. The legend "RoeNet" and "RoeNet (noise)" denote the networks are trained by the clean dataset and the dataset with noise $\epsilon \sim \mathcal{N}(0, 0.1)$, respectively.

in (1) as an example, we carry out a performance test of RoeNet. This hyperbolic PDE describes a Gaussian wave traveling on a line at a constant speed. Figure 3 demonstrates the results of a Gaussian wave propagating over time using RoeNet with a clean or noisy training data set, the Roe solver, and the analytical solution. The predicting results of RoeNet with or without noise fit perfectly with the analytical results during the entire computational time domain. However, the simulation results with the Roe solver get flattened and dissipated quickly over time. We remark that the prediction error of RoeNet accumulates over time. However, such growth of numerical error is relatively slow compared to that using traditional numerical methods. Therefore, RoeNet outperforms Roe solver with its accurate prediction.

4.2 | Inviscid Burgers' equation

Given a short window of continuous training data, we aim to use our model to predict the long-term future discontinuity of a nonlinear hyperbolic PDE, the inviscid Burgers' equation (Burgers in Table 2). Burgers' equation is a fundamental PDE occurring in various areas, such as fluid mechanics[42] and traffic flows.[43] This problem describes an initial wave traveling in the one-dimensional domain at a speed that varies with time and space. Due to the compression of conversed quantity, it could develop discontinuities such as shock waves.[44] In this example, $\boldsymbol{F}$ in (1) and initial condition in (2) are defined as

$$\begin{cases} F = \frac{1}{2}u^2, \\ u(t=0,x) = \frac{1}{2} + \sin(2\pi x). \end{cases} \tag{22}$$

Figure 4 plots the predicting results of RoeNet solving inviscid Burgers' equation with (22). Note that our training data for u is an approximated solution generated with the numerical method and only defined on an extremely short time period of $t \in [0, 0.002]$, as shown in the thin blue curve in Figure 4A. From both Figure 4A,B, we can see that RoeNet successfully learns the future discontinuities of the problem based only on the short-term continuous training data. This is a significant improvement in solving prediction problems, as predicting long-term discontinuities from a short window of a smooth dataset is generally considered impossible using traditional machine-learning approaches. In addition, as shown in Figure 5, the predictions of RoeNet and the simulation results of Roe solver are in perfect agreement at the early stage of the PDE evolution. This proves that RoeNet predicts the correct solutions for the evolving PDEs based on the numerical dataset.

We remark that our use of training data relies on a time step span of 2 for predicting data spanning hundreds of time steps. The strong generalization capabilities of our network stem from the integration of physical information into its structure, ensuring the generation and preservation of discontinuities throughout the evolutionary process. From a different perspective, we have improved a numerical solver, even without prior knowledge of its equation, by leveraging data to achieve higher accuracy compared to traditional algorithms. This enhanced solver is suitable for computing the long-term evolution of hyperbolic PDEs.

We plot the results of RoeNet for solving the Burgers equation with different initial conditions in Figure 6. First, we train the neural network using the dataset generated by the initial condition in (22). Then we use the trained model to predict the evolution of one dimensional Burgers equation with different initial conditions: $0.5 + \cos(2\pi)$, $0.5 - \sin(2\pi)$, and $2\exp(-10x^2) - 0.5$. With this experiment, we showcase our trained RoeNet model can predict the evolution from unseen initial conditions.

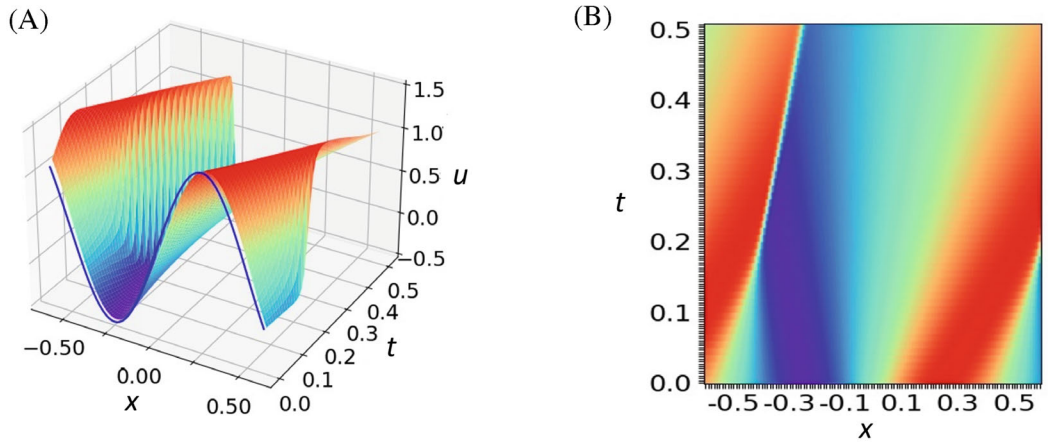

FIGURE 4 Results of RoeNet for solving a one component nonlinear hyperbolic PDE (Burgers in Table 2). (A) Blue line and rainbow-colored band denote the "short-term" continuous training data and the "long-term" predicting results of RoeNet, respectively. (B) Contour map to show the discontinuity on the evolution of conserved quantity u.

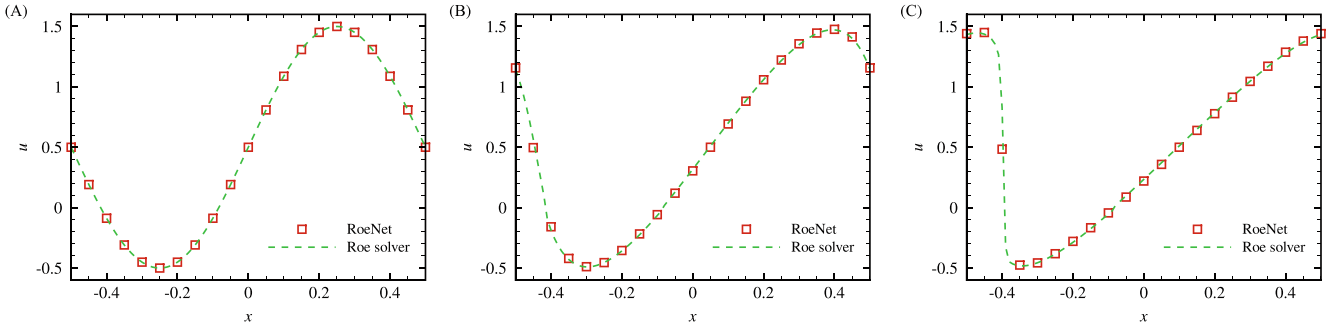

FIGURE 5 Comparison of RoeNet and Roe solver for solving a one component nonlinear hyperbolic PDE (Burgers in Table 2). (A) $t = 0$, (B) $t = 0.2$, and (C) $t = 0.4$.

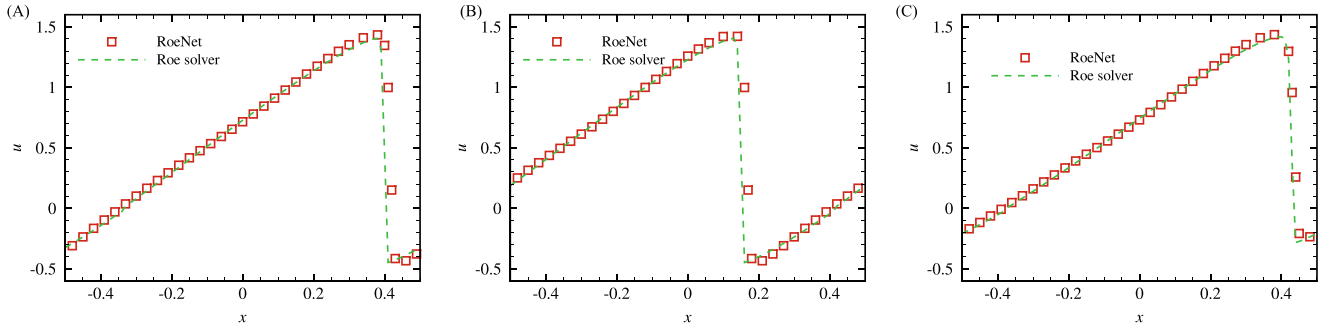

FIGURE 6 Results of RoeNet for solving an one component nonlinear hyperbolic PDE (Burgers in Table 2) with three different initial conditions (A) $0.5 + \cos(2\pi)$, (B) $0.5 - \sin(2\pi)$, and (C) $2\exp(-10x^2) - 0.5$. Here RoeNet model is trained with a dataset generated by the evolution from the initial condition in (22).

4.3 | Multi-component linear hyperbolic PDE

In addition, we apply RoeNet to solve a multi-component linear hyperbolic PDE, which shows how RoeNet successfully resolves coupled multiple waves traveling along different characteristic direction vectors. Specifically, we define the flux function $\boldsymbol{F}$ in (1) as a three-component linear function (3C Linear in Table 2) and set the initial condition in (2) as a piecewise constant Riemann condition:

$$\begin{cases} \boldsymbol{F}(\boldsymbol{u}) = \begin{bmatrix} 0.3237 & 2.705 & 5.4101 \\ 0.3597 & -0.4388 & -2.8777 \\ -0.0144 & 0.0576 & 1.1151 \end{bmatrix} \begin{bmatrix} u^{(1)} \\ u^{(2)} \\ u^{(3)} \end{bmatrix}, \\ \boldsymbol{u}(t=0, x \le -0.1) = (1.99, -0.46, 0.98), \\ \boldsymbol{u}(t=0, -0.1 < x < 0.1) = (4.69, -1.90, 1.04), \\ \boldsymbol{u}(t=0, x \ge -0.1) = (-1.99, 0.46, -0.98), \end{cases} \tag{23}$$

with $\boldsymbol{u} = [u^{(1)}, u^{(2)}, u^{(3)}]^T$. The linear propagation of the three components is coupled with the equation advancing since the Jacobian matrix of $\boldsymbol{F}$ is non-diagonal.

Figure 7 shows the initial conditions for $\boldsymbol{u}$ at $t = 0$ and the prediction results using RoeNet, numerical results using Roe solver, and analytical solutions at $t = 0.2$ of the three components $u^{(1)}$, $u^{(2)}$, and $u^{(3)}$, respectively. Figure 7B–D shows that the predictions with RoeNet match the analytical solutions perfectly, even with the discontinuous points traveling over time. However, obvious errors around the discontinuous points within $x \approx \pm 0.3$ occur with the simulations of Roe solver.

4.4 | Sod shock tube

We take the one-dimensional diatomic ideal gas problem to assess the performance of our model on solving multi-component Riemann problems with nonlinear flux functions (Sode Tube in Table 2). Specifically, the system is modeled by (1) with

$$\begin{cases} \boldsymbol{u} = (\rho, \rho v, e)^T, \\ \boldsymbol{F} = [\rho v, \rho v^2 + p, v(e+p)]^T, \end{cases} \tag{24}$$

where ρ is the density, p is the pressure, e is the energy, v is the velocity, and the pressure p is related to the conserved quantities through the equation of state $p = (\gamma - 1)\left(e - 0.5\rho v^2\right)$ with a heat capacity ratio $\gamma \approx 1.4$. We apply our model to the Sod shock tube problem,[45] a one-dimensional Riemann problem in the form of (1) with (24). The time evolution of

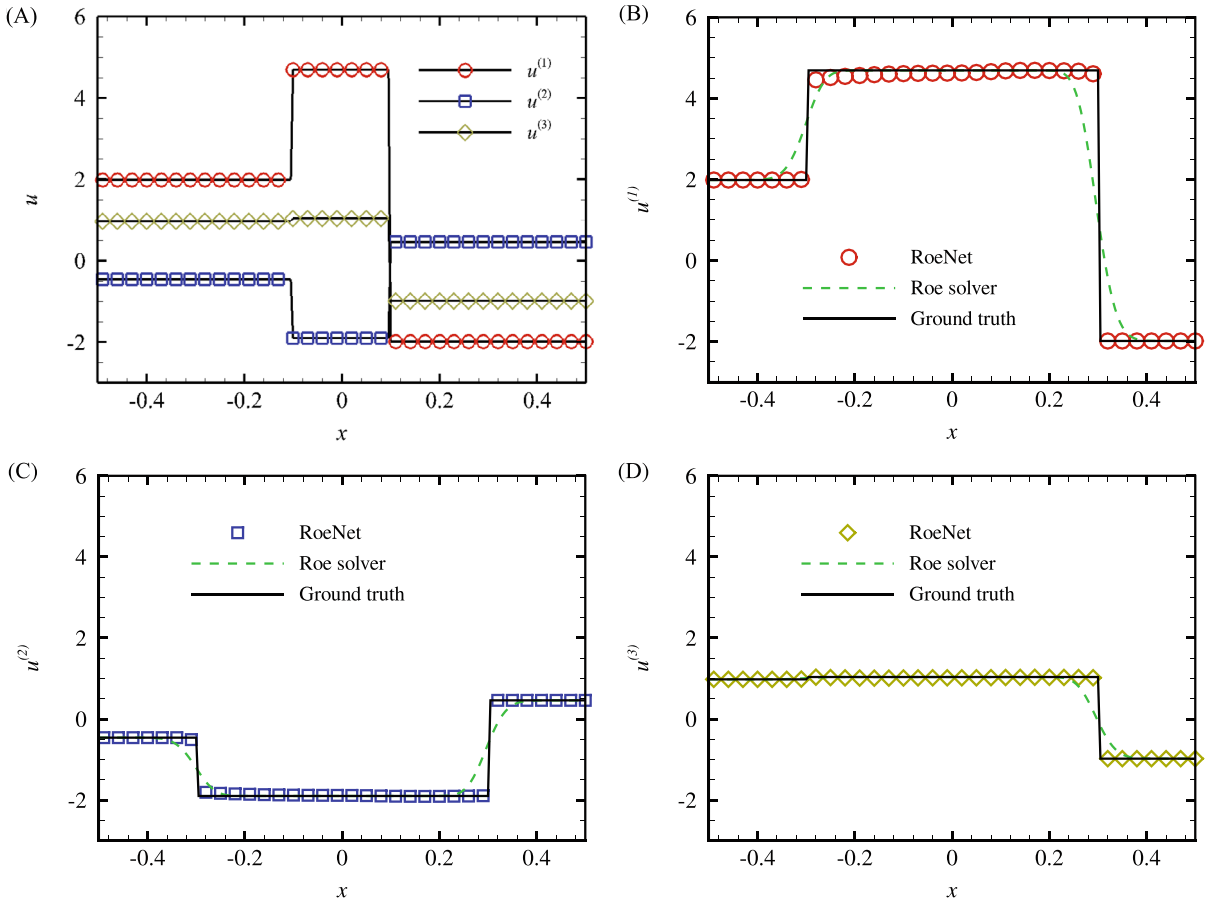

FIGURE 7 Comparison of RoeNet and Roe solver for solving a Riemann problem with three components and a linear flux function (3C Linear in Table 2). (A) Initial conditions for $\boldsymbol{u} = [u^{(1)}, u^{(2)}, u^{(3)}]$ at $t = 0$. (B), (C), and (D) plot the comparison of the prediction results using RoeNet, numerical results solved with Roe solver, and the analytical solutions at $t = 0.2$ of the three components $u^{(1)}$, $u^{(2)}$, and $u^{(3)}$, respectively.

this problem can be described by solving the mass, momentum, and energy conservation of ideal gas inside a slender tube, which leads to three characteristics, describing the propagation speed of various regions in the system.[45] In Figure 8, we plot the three components of the problem, at $t = 0.1$. Note that due to dissipation there is no sign of sonic glitch. Similar to the conclusion drawn from the previous Section 4.3, RoeNet exhibits higher accuracy in predicting the discontinuities of the nonlinear Riemann problem.

4.5 | Two-dimensional examples

We show our model's ability to predict higher-dimensional PDEs by learning and predicting the evolutionary behavior of a two-dimensional wave. To solve two-dimensional PDEs, we introduce another pair of $\boldsymbol{\Lambda}_\phi$ and $\boldsymbol{L}_\theta$ networks to learn those matrices in the second dimension and combine the vertical and horizontal increment of $\boldsymbol{u}^n$ together to predict $\boldsymbol{u}^{n+1}$. Specifically, the recursive relation of u^n in (16) is rewritten as follows:

$$\begin{aligned}
\boldsymbol{u}_{i,j}^{n+1} = \boldsymbol{u}_{i,j}^{n} &- \frac{1}{2}\lambda_r(\boldsymbol{L}^{1,n}_{i+\frac{1}{2},j,\theta})^{+}(\boldsymbol{\Lambda}^{1,n}_{i+\frac{1}{2},j,\phi} - |\boldsymbol{\Lambda}^{1,n}_{i+\frac{1}{2},j,\phi}|)\boldsymbol{L}^{1,n}_{i+\frac{1}{2},j,\theta}(\boldsymbol{u}^{n}_{i+1,j} - \boldsymbol{u}^{n}_{i,j}) \\
&- \frac{1}{2}\lambda_r(\boldsymbol{L}^{1,n}_{i-\frac{1}{2},j,\theta})^{+}(\boldsymbol{\Lambda}^{1,n}_{i-\frac{1}{2},j,\phi} + |\boldsymbol{\Lambda}^{1,n}_{i-\frac{1}{2},j,\phi}|)\boldsymbol{L}^{1,n}_{i-\frac{1}{2},j,\theta}(\boldsymbol{u}^{n}_{i,j} - \boldsymbol{u}^{n}_{i-1,j}) \\
&- \frac{1}{2}\lambda_r(\boldsymbol{L}^{2,n}_{i,j+\frac{1}{2},\theta})^{+}(\boldsymbol{\Lambda}^{2,n}_{i,j+\frac{1}{2},\phi} - |\boldsymbol{\Lambda}^{2,n}_{i,j+\frac{1}{2},\phi}|)\boldsymbol{L}^{2,n}_{i,j+\frac{1}{2},\theta}(\boldsymbol{u}^{n}_{i,j+1} - \boldsymbol{u}^{n}_{i,j}) \\
&- \frac{1}{2}\lambda_r(\boldsymbol{L}^{2,n}_{i,j-\frac{1}{2},\theta})^{+}(\boldsymbol{\Lambda}^{2,n}_{i,j-\frac{1}{2},\phi} + |\boldsymbol{\Lambda}^{2,n}_{i,j-\frac{1}{2},\phi}|)\boldsymbol{L}^{2,n}_{i,j-\frac{1}{2},\theta}(\boldsymbol{u}^{n}_{i,j} - \boldsymbol{u}^{n}_{i,j-1}),
\end{aligned} \tag{25}$$

with

$$\begin{cases} \boldsymbol{L}^{1,n}_{i+\frac{1}{2},j,\theta} = \boldsymbol{L}^{1}_{\theta}(\boldsymbol{u}^{n}_{i,j}, \boldsymbol{u}^{n}_{i+1,j}), & \boldsymbol{\Lambda}^{1,n}_{i+\frac{1}{2},j,\phi} = \boldsymbol{\Lambda}^{1}_{\phi}(\boldsymbol{u}^{n}_{i,j}, \boldsymbol{u}^{n}_{i+1,j}), \\ \boldsymbol{L}^{2,n}_{i,j+\frac{1}{2},\theta} = \boldsymbol{L}^{2}_{\theta}(\boldsymbol{u}^{n}_{i,j}, \boldsymbol{u}^{n}_{i,j+1}), & \boldsymbol{\Lambda}^{2,n}_{i,j+\frac{1}{2},\phi} = \boldsymbol{\Lambda}^{2}_{\phi}(\boldsymbol{u}^{n}_{i,j}, \boldsymbol{u}^{n}_{i,j+1}), \end{cases} \tag{26}$$

where L^1_θ and L^2_θ have the same network structure as L_θ in Figure 2; $\boldsymbol{\Lambda}^1_\phi$ and $\boldsymbol{\Lambda}^2_\phi$ have the same network structure as $\boldsymbol{\Lambda}_\phi$ in Figure 2. The rest training settings are identical to the one-dimensional examples.

For a linear example, we set $\boldsymbol{F}$ in (1) and the initial conditions in (4) as follows:

$$\begin{cases} \boldsymbol{F} = -\frac{1}{2}u, \\ u(t=0, \boldsymbol{x}) = \sin[2\pi(x_1 + x_2)], \end{cases} \tag{27}$$

where $\boldsymbol{x} = (x_1, x_2)$ is the two-dimensional coordinate. This system describes a trigonometrical function that travels on a two-dimensional plane with a constant speed.

Figure 9 plots the prediction results of RoeNet for solving the two dimensional linear wave (4) with (27). Similar to the one-dimensional examples, the two-dimensional RoeNet can learn from a limited amount of training data in a short time span to predict a long time evolution of $\boldsymbol{u}$ with high stability, as shown in Figure 9A. We also compare the prediction results of RoeNet with the ground truth solution in Figure 9B,C. The perfect match shows the high accuracy of RoeNet.

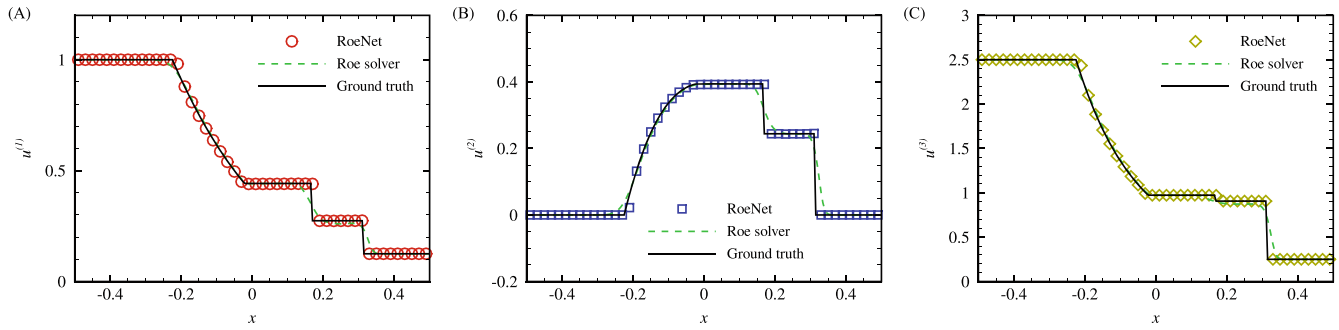

FIGURE 8 Comparison of RoeNet and Roe solver for solving a Riemann problem with three components and a nonlinear flux function (Sod Tube in Table 2). (A), (B), and (C) plot the comparison of the prediction results using RoeNet, numerical results solved with Roe solver, and the analytical solutions at $t = 0.1$ of the three components $u^{(1)}$, $u^{(2)}$, and $u^{(3)}$, respectively.

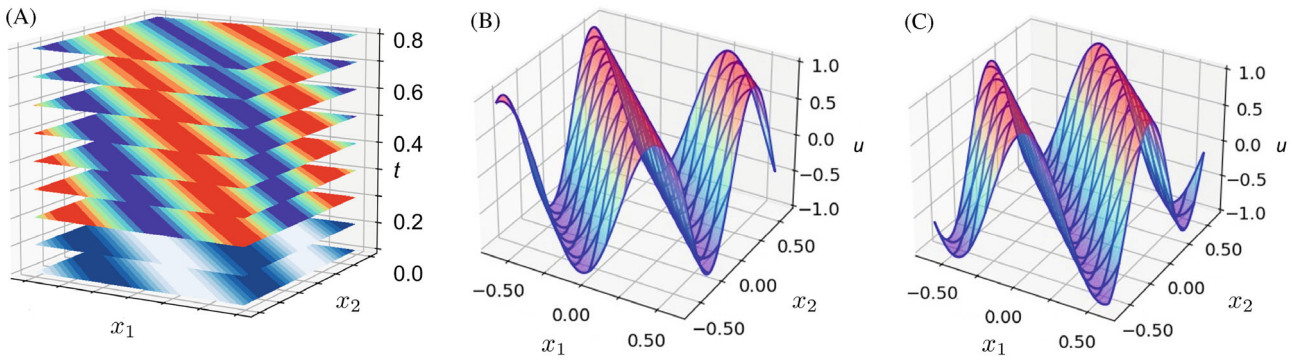

FIGURE 9 Prediction results of RoeNet for solving a two dimensional linear wave (2D Linear in Table 2). (A) Contour of $u(x_1, x_2)$ at different time. Here, the blue-white part denotes the time span of the training data set, while the rainbow-colored part at the top is the prediction by RoeNet. Predicted results (rainbow-colored contour map) and ground truth (blue mesh lines) are at (B) $t = 0.2$ and (C) $t = 0.4$, respectively.

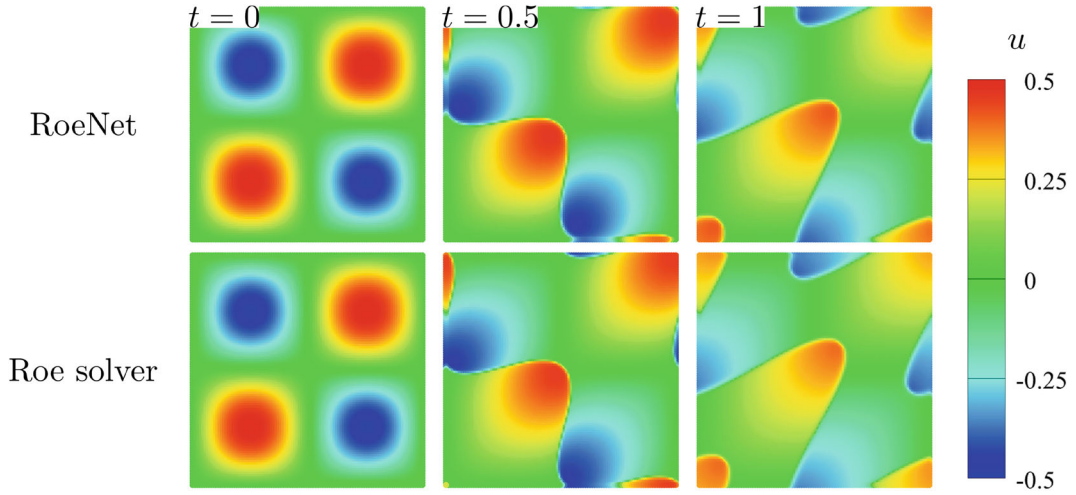

FIGURE 10 Comparison of RoeNet (top) and Roe solver (bottom) for solving a two dimensional nonlinear wave (2D Nonlinear in Table 2). The figures from left to right are $t = 0$, 0.5, and 1, respectively.

For a nonlinear example, we set $\boldsymbol{F}$ in (1) and the initial conditions in (4) as

$$\begin{cases} \boldsymbol{F} = \frac{1}{2}u^2, \\ u(t = 0, \boldsymbol{x}) = \frac{1}{2}\sin(2\pi x_1)\sin(2\pi x_2), \end{cases} \tag{28}$$

where $\boldsymbol{x} = (x_1, x_2)$ is the two-dimensional coordinate. This system describes a trigonometrical function that travels on a two-dimensional plane at varying speeds.

Figure 10 plots the comparison of RoeNet and Roe solver for solving a two-dimensional nonlinear hyperbolic PDE (4) with (28). This example demonstrates that RoeNet is able to predict multidimensional nonlinear PDEs and obtains results similar to those of Roe solver without the given governing equations.

5 | ABLATION TEST AND COMPARISON

In this section, we validate our design of the architecture of RoeNet and the numerical settings of the training process by comparing the effects of the usage of pseudoinverse, time-step size, and neural network size. The parameterization of RoeNet offers considerable flexibility, accommodating a broad spectrum of potential configurations. During the simulation process, special attention is devoted to the selection of an appropriate time step size. This selection aims to strike an optimal balance between minimizing validation loss and maximizing computational efficiency, thus yielding robust and reliable results within a reasonable time frame. In practical applications, the architecture of RoeNet can be further tailored to meet specific requirements through the adjustment of the number of ResBlocks. The quantity of ResBlocks can be modulated based on a multitude of factors, most notably the allowable error tolerance and the inherent complexity of the hyperbolic PDEs under investigation. By adjusting the composition of ResBlocks, RoeNet can be adeptly calibrated to offer a high level of precision while efficiently handling equations of varying complexity. We also compare our method with PINNs using DeepXDE library,[35] for solving hyperbolic PDEs.

5.1 | Ablation test

5.1.1 | Usage of pseudoinverse

We replace the inverse matrix in Roe solver with the pseudoinverse matrix in (14) to enhance the predictive model of RoeNet. Without the pseudoinverse, our network will lean toward learning an optimal Roe decomposition on a given

dataset. However, no rigorous mathematical theory proves such a Roe decomposition exists to ensure the Roe solver converges the hyperbolic PDEs' analytical solution. With the pseudoinverse, we could convert the Roe matrix, which is strictly a square matrix, to a high-dimensional matrix with hidden dimensions, thus greatly enhancing the expressiveness of the network. From the eigendecomposition perspective, RoeNet approximates the evolution of a low-dimensional hyperbolic PDE with a high-dimensional Riemann problem. Due to the high dimensionality of the approximated Riemann problem, we could achieve much higher accuracy than that of a traditional algorithm.

In this example, we show the importance of pseudoinverse in our network design. Figure 11 depicts the predicted results in the 3C Linear example in Section 4.3 using RoeNet with and without pseudoinverse, as well as the analytical solutions at a different time of the $u^{(2)}$ component. As the figure shows, predictions made by RoeNet without pseudoinverse fluctuate around the ground truth and fail to capture the exact moments of discontinuities that occur, while predictions made by RoeNet with pseudoinverse fit tightly to the ground truth. We remark that RoeNet with pseudoinverse outperforms RoeNet without pseudoinverse on accuracy, and the accuracy of prediction using RoeNet without pseudoinverse decreases over time.

5.1.2 | Time-step size

Figure 12 shows that in example 1C Linear in Section 4.1 with a fixed $T_{train} = 0.1$, the validation loss under the influence of different time step on clean data sets and noisy data sets. In this example, we conclude that the validation loss decreases

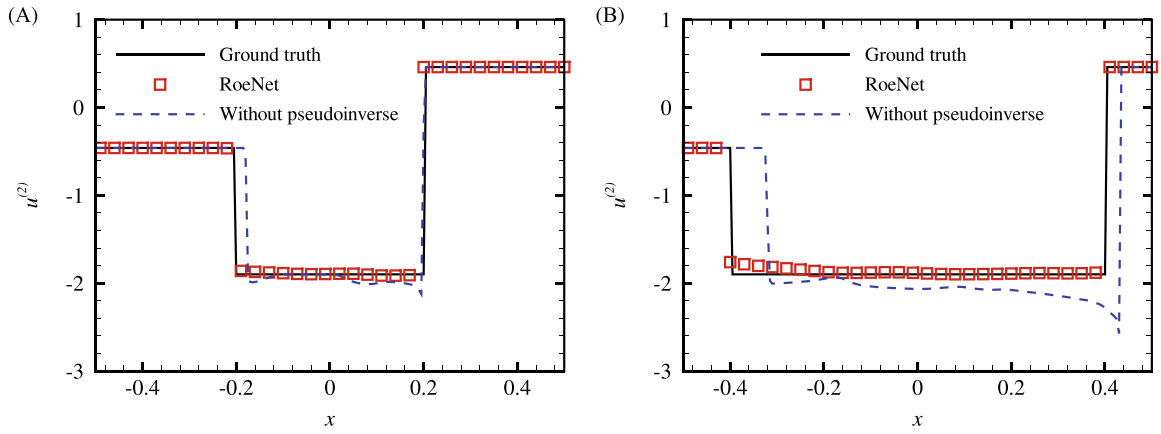

FIGURE 11 Comparison of prediction results of RoeNet with and without pseudoinverse, and analytical solutions for solving a Riemann problem with three components and a linear flux function (3C Linear in Table 2). The figures show the second component of $\boldsymbol{u}$ at (A) $t = 0.1$, (B) $t = 0.3$.

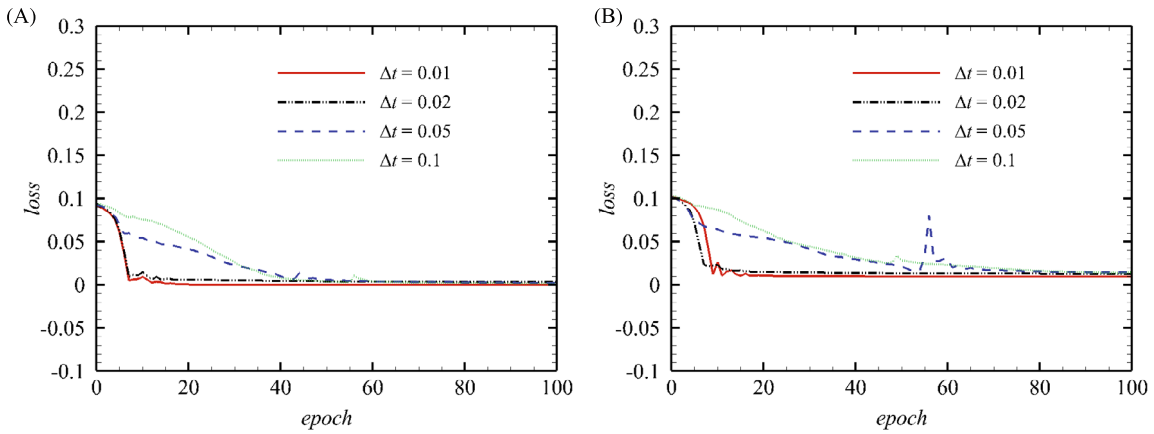

FIGURE 12 Comparisons of validation losses with the same $T_{train} = 0.1$ and different Δt in the training process of the problem 1C Linear in Table 2. The networks are trained by (A) the clean data set, (B) the data set with noise $\epsilon \sim \mathcal{N}(0, 0.1)$.

as Δt decreases and converges at $\Delta t \approx 0.01$. In general, a smaller Δt requires iterating more steps to reach the same time span T_{span}, and therefore requires a longer computation time. In the simulation, we choose a proper Δt to balance the small validation loss with the desired computational efficiency.

5.1.3 | Neural network size

We choose six-layer ResBloks as shown in Figure 2 to improve the expressiveness of the RoeNet. Since the RoeNet with six-layer ResBloks is more complex than the Roe matrix, the computational complexity of RoeNet is greater than that of the Roe solver. If we reduce the number of ResBlocks in RoeNet, the computation time of RoeNet can be significantly reduced. For this purpose, we test 6-layers, 4-layers, and 2-layers of ResBlock variations of RoeNets on the prediction of 1C Linear in Figure 13. We conclude that the reduction in layers increased the prediction error of the system to a small extent. However, it is still several orders of magnitude smaller than the Roe solver shown in Figure 3. In practice, we can adjust the number of ResBlocks in RoeNet according to the tolerance of error and the complexity of the hyperbolic PDEs. Overall, RoeNet has slightly greater computational requirements compared to the Roe solver, but both of them belong to the same order of computational complexity.

5.1.4 | Learning rate

Figure 14 presents an analysis of the error metrics associated with the utilization of RoeNet under varying learning rates, specifically within the context of Example 1C Linear, as detailed in Section 4.1. It is noteworthy that the error magnitudes are generally minimal when the learning rates are confined to a range between 0.001 and 0.05. However, caution is advised for learning rates exceeding 0.05, as these values exhibit a tendency to overshoot the optimal solution, thereby undermining the reliability of the model's predictions.

5.1.5 | Initial condition

In Figure 15, a comparative study is conducted to assess the predictive capabilities of RoeNet and the traditional Roe solver under differing initial conditions. For the purpose of this analysis, the initial condition employed during the training phase corresponds to that of Example 1C Linear, as specified in Section 4.1. Conversely, a sine function serves as the initial condition for the predictive phase. Figure 15A offers a visual representation that clearly highlights the superior accuracy of RoeNet's predictions in comparison to those generated by the Roe solver. Further insights can be observed from Figure 15B, which demonstrates that RoeNet consistently outperforms the Roe solver in terms of prediction error, across all observed time intervals.

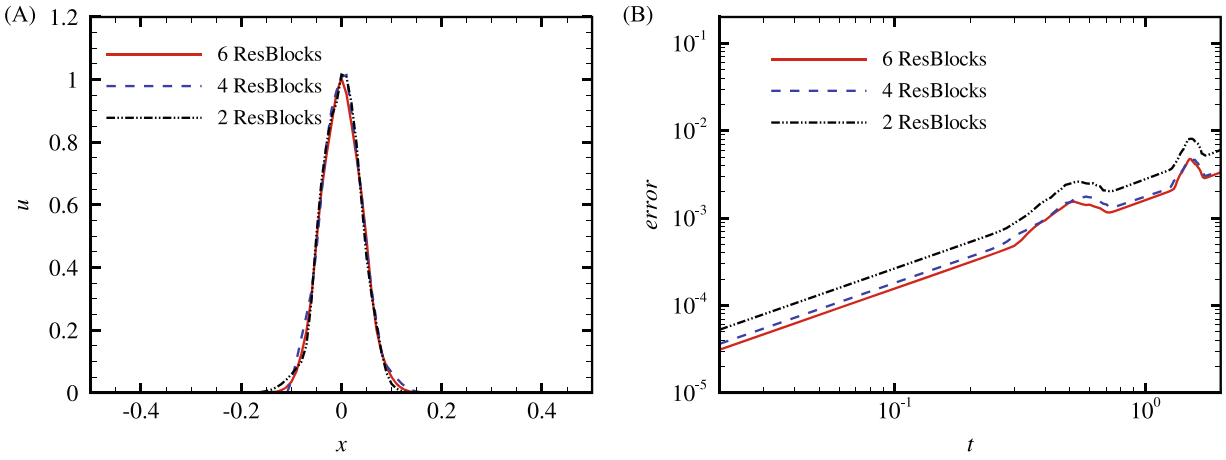

FIGURE 13 Comparisons of the prediction (A) results and (B) errors using RoeNet with different neural network sizes.

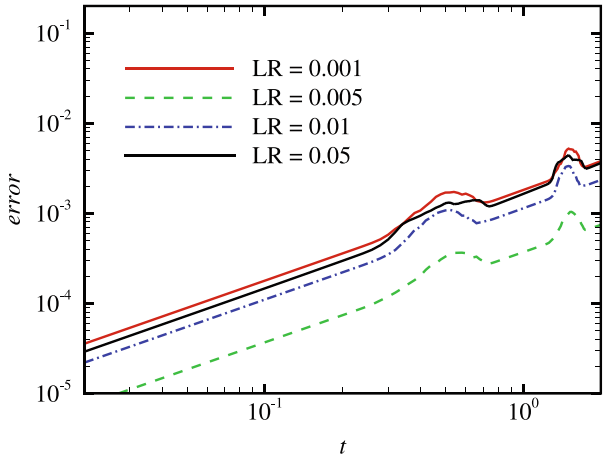

FIGURE 14 Errors using RoeNet with different learning rates.

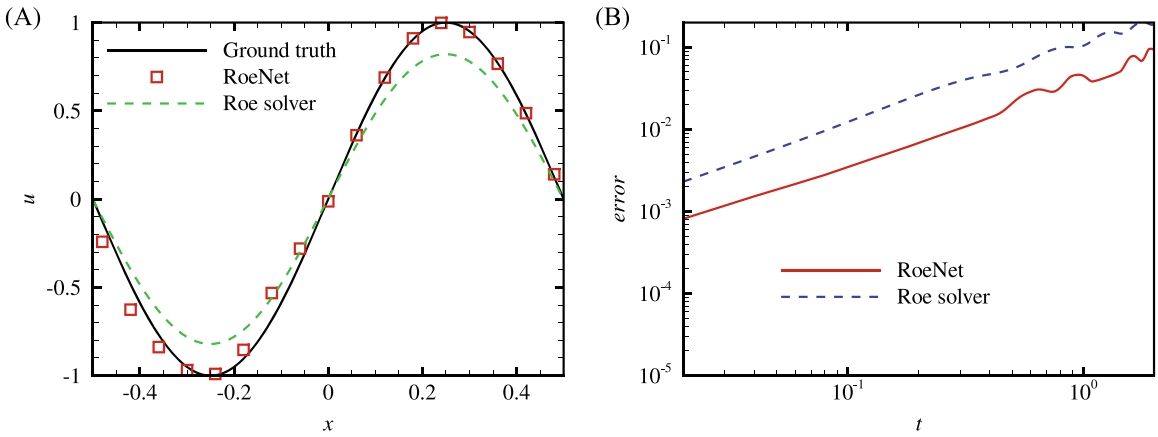

FIGURE 15 Comparisons of the prediction (A) results when changing the initial condition and (B) errors using RoeNet and Roe solver, respectively.

5.1.6 | Computational cost

To assess the computational efficiency and accuracy of RoeNet in comparison to the Roe solver, we performed an ablation test on the 1C Linear problem. In this test, RoeNet was evaluated using a grid size of 100, whereas the Roe solver underwent assessment at grid sizes of 100, 1000, and 2000. The data presented in Figure 16 provides valuable insights into the performance of both approaches. When working with the same grid size of 100, RoeNet exhibits a higher computational cost compared to Roe solver. However, despite this increased computational demand, RoeNet offers a significant advantage in predictive accuracy, with error rates that are several orders of magnitude lower than those of Roe solver.

Subsequently, as the grid size for the Roe solver increases, there is a proportional escalation in computational cost. Notably, when the grid size reaches the order of $O(1000)$, the computational cost of the Roe solver surpasses that of RoeNet with a grid size of 100. While there is an observable reduction in prediction error with the enlargement of the Roe solver's grid size, the achieved accuracy still falls short of outperforming RoeNet with a grid size of 100. Despite the enhanced resolution offered by the larger grid sizes in the Roe solver, the prediction error consistently remains higher than that of RoeNet, underscoring RoeNet's superior effectiveness in delivering precise predictions.

5.2 | Comparison with other methods

Generally, current methods based on neural networks could only solve a given PDE with the aid of a dataset or predict only continuous solutions of the PDEs. For the first class of neural networks, typical ones are PINNs.[26] Using PINNs

implies the networks need to be aware of the pre-established PDE model they are solving and require periodic feedback from the pre-established model on the loss, while RoeNet only requires training data sets without the knowledge of equations and feedback from the knowledge model. Moreover, for better convergence, the implementation of PINNs usually incorporates a Hessian-based optimizer like L-BFGS, which implies a longer training time and the usage of a closure function in optimization. While RoeNet mainly relies on gradient-based optimizers like SGD.

Without a given governing equation, conventional neural networks usually have difficulty in predicting the generation and evolution of discontinuous solutions. We especially showcase our model's unique ability to acomputationplish tasks that traditional machine learning approaches fail to complete. We show that our proposed RoeNet outperforms PINNs,[35] especially at larger t that is not included in training data. Figure 17 plots the comparison of the numerical results solved with Roe solver, as well as prediction results using RoeNet and PINNs at a different time of the problems 1C Linear in

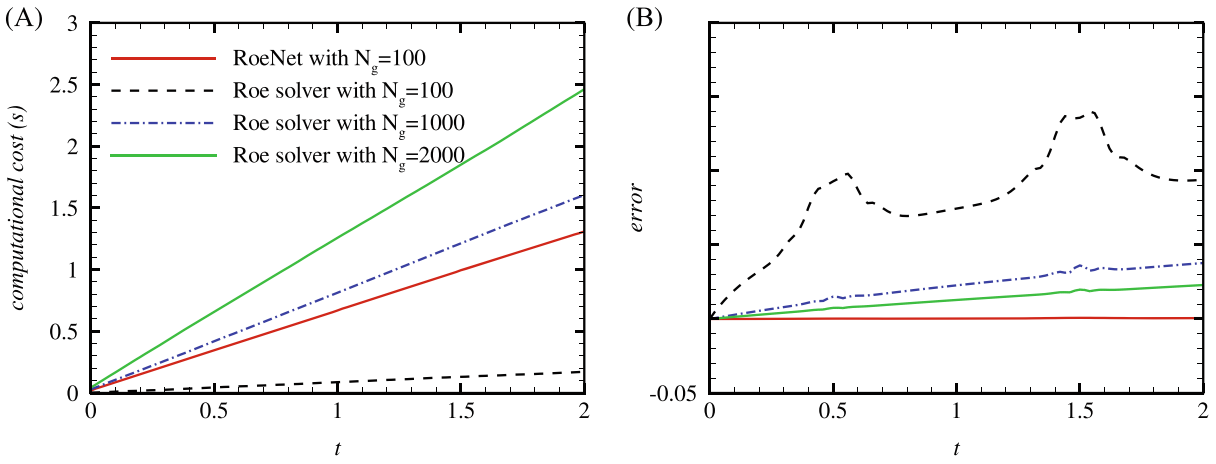

FIGURE 16 Comparisons of (A) computational costs and (B) prediction errors between RoeNet using a grid size of 100 and Roe solvers employing grid sizes of 100, 1000, and 2000 for the 1C Linear problem.

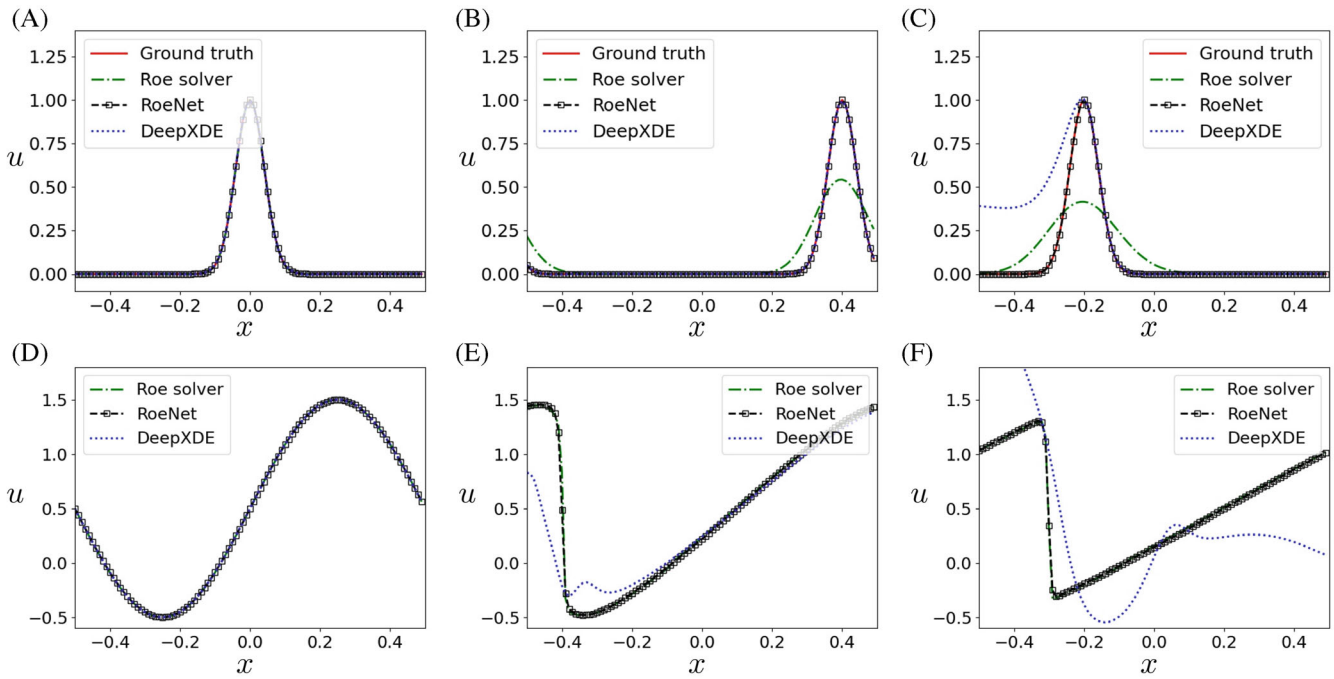

FIGURE 17 Comparion of the numerical results solved with Roe solver, prediction results using RoeNet and PINNs[35] at (A) $t = 0$, (B) $t = 0.4$, (C) $t = 0.8$ of the problem 1C Linear in Table 2 and (D) $t = 0$, (E) $t = 0.2$, (F) $t = 0.4$ of the problem Burgers in Table 2.

Section 4.1 and Burgers in Section 4.2. It shows that RoeNet, with no initial knowledge of the equations, outperforms PINNs even for the future t that is not included in the training data.

In contrast to conventional numerical approaches, RoeNet serves as a data-driven solver, eliminating the need for prior knowledge regarding the system's evolution equation. Moreover, RoeNet is designed with an optimization-based approach for constructing its numerical scheme, and the optimization space of its numerical scheme fully encompasses that of the Roe solver. Consequently, compared to traditional numerical methods, it can provide more precise simulations for PDE evolution.

6 | CONCLUSION

We present the RoeNet, a neural network analogous to a numerical Roe solver that could predict discontinuity in the future based on partially or entirely smooth data sets. Our numerical experiments show that RoeNet outperforms both the traditional Riemann-type numerical solver and the recent deep-learning solver in accuracy, robustness, and ability to predict unknown discontinuity in the future. Our ablation tests further demonstrate these computational merits drawn from our templatized prior embedding, whose scheme preserves the mathematical properties of the original Roe template. Our novel method using Roe template with pseudoinverse embedding is able to expand the parameter dimensions and conduct long-term predictions of discontinuity, making a significant progress in building prior-embedded machine learning methods to predict long-term future dynamic behaviors that are invisible in the given training data.

Compared to the Roe solver, RoeNet's primary limitation lies in its challenges when dealing with intricate computational domains and boundary conditions. The current network architecture is designed based on a deep convolutional neural network tailored for uniform grids, making it highly challenging to extend this network to complex scenarios. Furthermore, given its relatively substantial computational scale, it does not currently accommodate complex high-dimensional cases. Taking a broader perspective, RoeNet exhibits the following supplementary limitations. With the use of the templatized neural network architecture based on the traditional numerical Roe solver, we limit the scope of solvable systems to hyperbolic PDEs. Thus, it fails to address other physical systems, such as elliptic and parabolic equations. Also, our current RoeNet model discretizes the evolving variables on a regular orthogonal grid, leaving the systems defined an irregular domains out of consideration. Moreover, despite that our neural network takes advantage of the spatial discretization of the flux function in the traditional numerical Roe solver, temporal discretization is designed to work with the first-order Euler scheme. Similar to the traditional Roe solver, RoeNet requires a small time step in the training and the predicting process, and the preference for a small time step, as shown in the ablation test 5, further proves this. Moreover, the Roe solver is a discretization of the hyperbolic PDE in differential form. Discretization methods based on integral forms, such as HLLC,[46] may inspire us to construct more sophisticated neural network methods. Finally, we remark that many approximate Riemann solvers often find entropy-violating solutions to nonlinear hyperbolic PDEs. We did not observe significant entropy violation in the prediction results of RoeNet, which is likely because RoeNet introduces hidden dimensions to avoid matrix-based decomposition in Roe solver. In the future works, we plan to enhance RoeNet with the entropy fix technique to improve the robustness of the network.

A broad range of applications and extensions are yet to be explored. One of the interesting future directions to explore is to utilize the current model to predict real-world problems in fields like traffic flow, long-term meteorological prediction, and financial analysis. Moreover, the design of our templaterizale neural network architecture that combines modernized mathematical priors and data-driven paradigms opens up new possibilities for tackling complex physics problems by embedding structured priors. Broadly speaking, any prior template, including but not limited to numerical stencil generators and design principles, can empower high-performance learning pipelines. We envision a series of future works based on our templaterizable prior embedding scheme to bridge scientific computing and machine learning, such as templaterizing the high-resolution,[47] structure-preserved,[48–50] and energy-conserved[51] schemes in computational physics.

In forthcoming research, we aim to delve deeper into additional two-dimensional case studies, specifically those governed by the two-dimensional Euler equations. Recent scholarly work, notably by Mao et al. and Jagtap et al., attests to the promising potential of PINNs. The study by Mao et al. investigates the use of PINNs for approximating 2D Euler equations to model complex aerodynamic flows, including oblique shock waves.[30] The research demonstrates that PINNs are effective in capturing these 2D Euler flow phenomena with only a sparse set of data points. Jagtap et al. employs PINNs and their extended version, extended PINNs (XPINNs), to solve complex inverse problems related to 2D Euler equations

for modeling supersonic compressible flows in aerospace applications, including expansion waves and oblique and bow shock waves.[52] Traditional methods often struggle with these ill-posed problems, but PINNs and XPINNs succeed by incorporating domain decomposition, enforcing entropy and positivity conditions on density and pressure.

These studies have effectively showcased PINNs' capability in handling intricate two-dimensional problems. Concurrently, we foresee that RoeNet might offer a new perspective and deliver robust performance on these problems. A comprehensive comparison between PINNs and RoeNet is delineated in Section 5.2. It is crucial to note that the utility of PINNs necessitates their integration with a pre-established PDE model, demanding periodic updates to the loss function based on this model. Conversely, RoeNet operates independently of such equation-based feedback, requiring only training datasets for its functionality. Additionally, PINNs commonly employ Hessian-based optimizers such as L-BFGS, which tends to prolong training times and necessitates the use of closure functions in optimization processes. In contrast, RoeNet predominantly utilizes gradient-based optimizers like Stochastic SGD, streamlining the computational workload.

ACKNOWLEDGMENTS

We acknowledge the funding support from NSF-1919647, 2106733, 2144806, and 2153560. Shiying Xiong is supported by the 100 Researchers Program Start-up Grant of Zhejiang University. Our code is available at https://github.com/ShiyingXiong/RoeNet.

CONFLICT OF INTEREST STATEMENT

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

REFERENCES

1. White C, Ushizima D, Farhat C. Neural networks predict fluid dynamics solutions from tiny datasets. arXiv:1902.00091. 2019.
2. Philemon MD, Ismail Z, Dare J. A review of epidemic forecasting using artificial neural networks. *Int J Epidemiol Res*. 2019;6:132-143.
3. Methaprayoon K, Lee WJ, Rasmiddatta S, Liao JR, Ross RL. Multistage artificial neural network short-term load forecasting engine with front-end weather forecast. *IEEE Trans Ind Appl*. 2007;43(6):1410-1416.
4. Troudet T, Garg S, Merrill W. Neural network application to aircraft control system design. *National Aeronautics and Space Administration*; 1991:NASA TM-105151.
5. Yu A, Yang H, Yao Q, et al. Traffic scheduling based on spiking neural network in hybrid E/O switching intra-datacenter networks. *State Key Laboratory of Information Photonics and Optical Communication, Beijing University of Posts and Telecommunications, No.10 Xitucheng Road. Beijing University Press*; 2020:1-7.
6. Jahn M. Artificial neural network regression models in a panel setting: Predicting economic growth. *Econ. Model.* 2020;91:148-154.
7. Dam A, Zegeling PA. A robust moving mesh finite volume method applied to 1D hyperbolic conservation laws from magnetohydrodynamics. *J Comput Phys*. 2006;216:526-546.
8. Bressan A. *Hyperbolic Conservation Laws*. Springer; 2013.
9. Mao F, Kang L, Wu J, et al. A study of longitudinal processes and interactions in compressible viscous flows. *J Fluid Mech*. 2020;893:A23.
10. Scheid W, Muller H, Greiner W. Nuclear shock waves in heavy-ion collisions. *Phys Rev Lett*. 1974;32:741-745.
11. Terao K, Inagaki T. Interaction between combustion and shock waves. *Jpn J Appl Phys*. 1989;28:1226-1234.
12. Toro EF. *Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction*. Springer Science & Business Media; 2013.
13. Colella P, Glaz HM. Efficient solution algorithms for the Riemann problem for real gases. *J Comput Phys*. 1985;59:264-289.
14. Vilara F, Mairea P, Abgrall R. Cell-centered discontinuous Galerkin discretizations for two-dimensional scalar conservation laws on unstructured grids and for one-dimensional Lagrangian hydrodynamics. *Comput Fluids*. 2011;46:498-504.
15. Spekreijse S. Multigrid solution of monotone second-order discretizations of hyperbolic conservation laws. *Math Comput*. 1987;49:135-155.
16. McCorquodale P, Colella P. A high-order finite-volume method for conservation laws on locally refined grids. *Comm App Math Comput Sci*. 2011;6:1-25.
17. Griebel M, Zumbusch G. Adaptive sparse grids for hyperbolic conservation laws. *Institute for Applied Mathematics*. University Bonn; 1999:411-422.
18. Colella P, Graves DT, Benjamin JK, Modiano D. A Cartesian grid embedded boundary method for hyperbolic conservation laws. *J Comput Phys*. 2006;211:347-366.
19. Godunov SK. A difference method for numerical calculation of discontinuous solutions of the equations of hydrodynamics. *Mat. Sb. (N.S.)*. 1959;89:271-306.
20. Nishikawa H, Kitamura K. Very simple, carbuncle-free, boundary-layer-resolving, rotated-hybrid Riemann solvers. *J Comput Phys*. 2008;227:2560-2581.
21. Quirk JJ. *A Contribution to the Great Riemann Solver Debate*. Springer; 1997:550-569.
22. Yarosky D. Error bounds for approximations with deep ReLU networks. *Neural Netw*. 2017;94:103-114.

23. Petersen P, Voigtländer F. Optimal approximation of piecewise smooth functions using deep ReLU neural networks. *Neural Netw*. 2018;170:296-330.
24. Imaizumi M, Fukumizu K. Deep learning networks learn non-smooth functions effectively. *The 22nd International Conference on Artificial Intelligence and Statistics*; 2019:869-878.
25. Suzuki T. Adaptivity of deep RELU network for learning in Besov and mixed smooth Besov spaces: optimal rate and curse of dimensionality. *International Conference on Learning Representations*; 2019.
26. Raissi M, Perdikaris P, Karniadakis GE. Inferring solutions of differential equations using noisy multi-fidelity data. *J Comput Phys*. 2017;335:736-746.
27. Hornik K, Stinchcombe M, Halbert W. Multilayer feedforward networks are universal approximators. *Neural Netw*. 1989;2:359-366.
28. Zhang D, Guo L, Karniadakis GE. Learning in modal space: solving time-dependent stochastic PDEs using physics-informed neural networks. *SIAM J Sci Comput*. 2019;42:A639-A665.
29. Michoski C, Milosavljevic M, Oliver T, Hatch D. Solving differential equations using deep neural networks. *Neurocomputing*. 2019;399:193-212.
30. Mao Z, Jagtap AD, Karniadakis GE. Physics-informed neural networks for high-speed flows. *Comput Method Appl Methods*. 2020;360:112789.
31. Batista G, Prati RC, Monard MC. A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explor*. 2004;6:20-29.
32. Xiong S, He X, Tong Y, Deng Y, Zhu B. Neural vortex method: from finite Lagrangian particles to infinite dimensional Eulerian dynamics. *Comput Fluids*. 2023;258:105811.
33. Yang L, Meng X, Karniadakis GE. B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data. *J Comput Phys*. 2021;425:109913.
34. Roe PL. Approximate riemann solvers, parameter vectors and difference schemes. *J Comput Phys*. 1981;43:357-372.
35. Lu L, Meng X, Mao Z, Karniadakis GE. DeepXDE: A deep learning library for solving differential equations. *SIAM Rev Soc Ind Appl Math*. 2019;63:208-228.
36. Wu JZ, Ma HY, Zhou MD. *Vortical Flows*. Springer; 2015.
37. Evans LC. *Partial Differential Equations*. 2nd ed. American Mathematical Society; 2010.
38. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. *Proceedings of The IEEE Conference on Computer Vision and Pattern Recognition*; 2016:770-778.
39. Jagtap AD, Kawaguchi K, Em KG. Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks. *Proc R. Soc A*. 2020;476(2239):20200334.
40. Jagtap AD, Kawaguchi K, Karniadakis GE. Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. *J Comput Phys*. 2020;404:109136.
41. Kingma DP, Adam BJ. A method for stochastic optimization. arXiv:1412.6980. 2014.
42. Baker J, Armaou A, Christofides PD. Nonlinear control of incompressible fluid flow: Application to Burgers' equation and 2D channel flow. *J Math Anal Appl*. 2000;252:230-255.
43. Nagatani T, Emmerich H, Nakanishi K. Burgers equation for kinetic clustering in traffic flow. *Phys A: Stat Mech Appl*. 1998;255:158-162.
44. Burgers JM. A mathematical model illustrating the theory of turbulence. *Adv Appl Mech*. 1948;1:171-199.
45. Sod GA. A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws. *J Comput Phys*. 1978;27:1-31.
46. Harten A, Lax PD, Leer BV. On upstream differencing and Godunov-type schemes for hyperbolic conservation laws. *SIAM Rev*. 1983;25:35-61.
47. Wen X, Don W, Gao Z, Hesthaven JS. An edge detector based on artificial neural network with application to hybrid compact-WENO finite difference Scheme. *J Sci Comput*. 2020;83:49.
48. Tong Y, Xiong S, He X, Pan G, Zhu B. Symplectic neural networks in Taylor series form for Hamiltonian systems. *J Comput Phys*. 2021;437(110325):110325.
49. Xiong S, Tong Y, He X, Yang S, Yang C, Zhu B. Nonseparable symplectic neural networks. *International Conference on Learning Representations*; 2021.
50. DiPietro DM, Xiong S, Zhu B. Sparse symplectically integrated neural networks. *Advances in Neural Information Processing Systems*; 2020.
51. Greydanus S, Dzamba M, Yosinski J. Hamiltonian neural networks. *Conference on Neural Information Processing Systems*; 2019:15379-15389.
52. Jagtap AD, Mao Z, Adams N, Karniadakis GE. Physics-informed neural networks for inverse problems in supersonic flows. *J Comput Phys*. 2022;466:111402.

How to cite this article: Tong Y, Xiong S, He X, et al. RoeNet: Predicting discontinuity of hyperbolic systems from continuous data. *Int J Numer Methods Eng*. 2024;125(6):e7406. doi: 10.1002/nme.7406