ROENETS: PREDICTING DISCONTINUITY OF HYPERBOLIC SYSTEMS FROM CONTINUOUS DATA

Anonymous authors
Paper under double-blind review

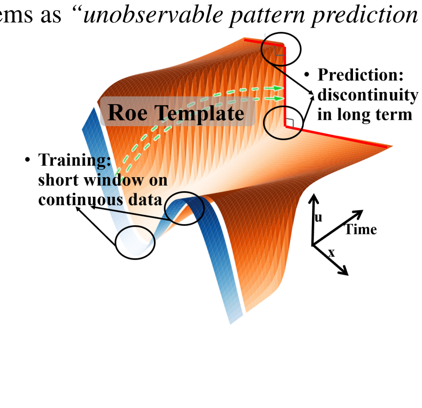
ABSTRACT

Predicting future discontinuous phenomena that are unobservable from the training data sets has been a challenging problem for scientific machine learning. We introduce a novel paradigm to predict the emergence and evolution of various kinds of discontinuities for hyperbolic dynamic systems based on smooth observation data. At the heart of our approach is a templaterizable and data-driven Riemann solver that functions as a strong inductive prior to tackle the potential discontinuities. The key design of our templaterized Riemann approximator is inspired by the classical Roe solver (P. L. Roe, J. Comput. Phys., vol. 43, 1981), which served as a fundamental mathematical tool for simulating various hyperbolic systems in computational physics. By carefully designing the computing primitives, data flow, and incorporating a novel pseudoinverse processing module, we enable our data-driven predictor to inherently satisfy all the essential mathematical criteria of a Roe hyperbolic solver derived from the first principles and hence deliver accurate predictions of hyperbolic dynamics. The most salient feature of our method is its capability in predicting future discontinuities accurately and robustly that are invisible from training data. We demonstrate by various examples that our data-driven Roe predictor can outperform both original human-designed Roe schemes and deep neural networks with weak priors, with respect to its accuracy, robustness, and minimum training data.

1 INTRODUCTION

Predicting the invisible future is an ultra ability humans aim to build into machine intelligence. To materialize this visionary goal into an algorithmic context, we propose to solve a new type of machine learning problems to predict a dynamic system's future behavior patterns that are invisible 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 (see the inset figure). Such examples are ubiquitous in natural and social sciences. For example, fluid mechanists want to predict the future occurrence of shock waves based on the observation of the current smooth fluid flow. Epidemiologists want to predict the break-out time of an epidemic disease based on the collected data on an early stage. More examples exist in weather forecast (Methaprayoon et al., 2007), aircraft control (Troudet et al., 1991), traffic scheduling (Yu et al., 2020), and economic crisis prediction (Jahn, 2020). From a mathematical perspective, we summarize these prediction problems as seeking the evolutionary solution of a dynamic system whose current observable status are significantly different from its targeted behaviors in the long future. We name these problems as *"unobservable pattern prediction"*.

Devising a data-driven paradigm to address the unobservable pattern prediction problem is challenging for two reasons. *First*, the target patterns to be predicted do not exist in the training data set. Hence, we cannot leverage the expressive power of the modern deep neural networks to uncover these patterns from abundant training data. In many scenarios, the training data merely covers a small region of the entire solution space (see the right inset figure). *Second*, there lack accurate models to describe the evolution that connects the current, observable states to the future, predictive targets. Mathematically, we typically



need to use partial differential equations (PDEs) to model such evolution. However, in many cases, we do not have these differential equations 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. These extremely partial observations and insufficiently accurate models jointly make the unobservable prediction problems seemingly infeasible to solve.

In this paper, we conduct a preliminary exploration on tackling the computational challenges of unobservable prediction problems by proposing a novel approach that hybridize model priors and learning paradigms on an architectural level. We describe our high-level design philosophy as "*templaterizable prior embedding*": Analog 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 facilitated with these prior-embedded modules can characterize dynamic evolution in which (i) *the target patterns are invisible in training data sets*, (ii) *observation window is partial and short*, and (iii) *no accurate PDE models connecting the current and future*.

As the first step toward the vision of unobservable dynamics prediction, we study a broad, representative category of problems—*hyperbolic dynamic systems*—which exist ubiquitously in many fields of physical science and engineering (see examples in Scheid et al., 1974; Terao & Inagaki, 1989; Dam & Zegeling, 2006; Bressan, 2013; Mao et al., 2020a). Mathematically, we narrow down our focus further onto *Riemann problem* (Toro, 2013), which is a specific type of hyperbolic PDEs with two constant initial states that will be separated by a single discontinuity in the (long) future. The data-driven prediction of Riemann problem faces the aforementioned two challenges of the partial observation and the lack of accurate model. To tackle these challenges, we devise our "templaterizable priors" by creating a template model based on a classical Riemann solver—*Roe solver* (Roe, 1981). Roe solver served as a fundamental mathematical tool in the history of scientific computing. It is not only a specific numerical scheme but also a set of mathematical design guidelines to create an effective numerical Riemann solver. In the following sections, we will demonstrate how we take advantage of these strong 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. We examine the RoeNet's ability by conducing long-term discontinuity predictions for a broad range of hyperbolic PDEs. The results demonstrate the accuracy, robustness, and the outstanding predictive capability on processing invisable phenomena of our RoeNet framework, which outperforms both human-designed numerical schemes and the state-of-the-art neural networks.

2 Roe template with Pseudoinverse Embedding

In this section, we will introduce the mathematical model of our RoeNet design. First, we briefly introduce the mathematical background of hyperbolic systems and the classical form of Roe solver (we refer the reader to the Appendix A for a detailed description). Next, we introduce our design of the Roe template with pesudoinverse embedding, which accommodate the data processing and training over the entire learning pipeline (see the details for network architecture in Section 3).

Hyperbolic Dynamic System A one dimensional hyperbolic problem can be described as a first-order partial differential equation (PDE) of the form

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

with a initial condition $\boldsymbol{u}(t = t_0, x) = \boldsymbol{u}_0(x)$ and a proper boundary condition. Here $\boldsymbol{u} = (u^{(1)}, u^{(2)}, \cdots, u^{(N_c)})$ with N_c components is called the conserved quantity, while $\boldsymbol{F} = (F^{(1)}, F^{(2)}, \cdots, F^{(N_c)})$ is the flux. The variable $t \in [t_0, t_T]$ denotes time, while $x \in \Omega$ is the space variable. The evolution of the PDE is discretized over a Cartersian grid.

We remark that for the discontinuous solution, (1) is interpreted as a integral form of weak solution (e.g. Eymard et al., 1995). In addition, (1) can be written in a high dimensional form

$$\frac{\partial \boldsymbol{u}}{\partial t} + \sum_{i=1}^{N_d} \frac{\partial \boldsymbol{F}_i(\boldsymbol{u})}{\partial x_i} = \boldsymbol{0}. \tag{2}$$

If we can successfully solve (1), (2) can be solved spontaneously by applying the method of approximating $\partial \boldsymbol{F}(\boldsymbol{u})/\partial x$ to approximate $\partial \boldsymbol{F}_i(\boldsymbol{u})/\partial x_i$.

Roe Solver The Roe solver involves finding an estimate for the intercell numerical flux at the interface between two computational cells, on some discretised space-time computational domain. In particular, the Roe solver Roe (1981) discretizes (1) as

$$\boldsymbol{u}_j^{n+1} = \boldsymbol{u}_j^n - \lambda_r \left(\hat{\boldsymbol{F}}(\boldsymbol{u}_j^n, \boldsymbol{u}_{j+1}^n) - \hat{\boldsymbol{F}}(\boldsymbol{u}_{j-1}^n, \boldsymbol{u}_j^n) \right), \tag{3}$$

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

$$\hat{\boldsymbol{F}}(\boldsymbol{u}, \boldsymbol{v}) = \frac{1}{2} \left[\boldsymbol{F}(\boldsymbol{u}) + \boldsymbol{F}(\boldsymbol{v}) - |\tilde{\boldsymbol{A}}(\boldsymbol{u}, \boldsymbol{v})|(\boldsymbol{v} - \boldsymbol{u}) \right]. \tag{4}$$

Here, Roe matrix $\tilde{\boldsymbol{A}}$ is assumed constant between two cells and must obey the three Roe conditions (termed property U in Appendix A) including diagonalizable with real eigenvalues, consistent with the exact Jacobian, and preserving conserved quantities. The key step to design an effective Roe solver consists in finding the Roe matrix $\tilde{\boldsymbol{A}}$ that is assumed constant between two cells by fulfilling the above "Property U."

Pseudoinverse Embedding We propose an effective approach to solve hyperbolic dynamic systems by applying Roe solver under a neural network perspective. Given the diagonalization of the Roe matrix

$$\tilde{\boldsymbol{A}} = \boldsymbol{L}^{-1} \boldsymbol{\Lambda} \boldsymbol{L}, \tag{5}$$

our model consists of two networks which learn $\boldsymbol{L}$ and $\boldsymbol{\Lambda}$ respectively.

Using neural networks to directly approximate $\boldsymbol{L}$ and $\boldsymbol{\Lambda}$ is ineffective, since the number of learnable parameters is limited by the number of components. To enhance the expressiveness of our model, we introduce the concept of pseudoinverses by replacing $\boldsymbol{L}^{-1}$ with

$$\boldsymbol{L}^{+} = (\boldsymbol{L}^T \boldsymbol{L})^{-1} \boldsymbol{L}^T \tag{6}$$

to enable a hidden dimension N_h so that we are flexible with the number of parameters.

Roe Template Substituting (18) and (6) into (15) along with the third Roe condition yields

$$\begin{aligned} \boldsymbol{u}_j^{n+1} = & \boldsymbol{u}_j^n - \frac{1}{2}\lambda_r \boldsymbol{L}^{+}_{j+\frac{1}{2}} (\boldsymbol{\Lambda}_{j+\frac{1}{2}} - |\boldsymbol{\Lambda}_{j+\frac{1}{2}}|) \boldsymbol{L}_{j+\frac{1}{2}} (\boldsymbol{u}_{j+1}^n - \boldsymbol{u}_j^n) \\ & - \frac{1}{2}\lambda_r \boldsymbol{L}^{+}_{j-\frac{1}{2}} (\boldsymbol{\Lambda}_{j-\frac{1}{2}} + |\boldsymbol{\Lambda}_{j-\frac{1}{2}}|) \boldsymbol{L}_{j-\frac{1}{2}} (\boldsymbol{u}_j^n - \boldsymbol{u}_{j-1}^n), \end{aligned} \tag{7}$$

with

$$\boldsymbol{L}_{j+\frac{1}{2}} = \boldsymbol{L}(\boldsymbol{u}_j^n, \boldsymbol{u}_{j+1}^n), \quad \boldsymbol{\Lambda}_{j+\frac{1}{2}} = \boldsymbol{\Lambda}(\boldsymbol{u}_j^n, \boldsymbol{u}_{j+1}^n). \tag{8}$$

Equation (7) serves as our template to evolve the system's states from $\boldsymbol{u}_j^n$ to $\boldsymbol{u}_j^{n+1}$.

3 Neural Network Architecture

Overall, RoeNet is constructed with two networks A_L and A_Λ , which learn $\boldsymbol{L}$ and $\boldsymbol{\Lambda}$ in (8) respectively. As shown in Figure 1, RoeNet takes $\boldsymbol{u}_j^n$ and its direct neighbors, $\boldsymbol{u}_{j-1}^n$ and $\boldsymbol{u}_{j+1}^n$, as the input, and outputs $\boldsymbol{u}_j^{n+1}$. Specifically, RoeNet contains two parts, each consists of a A_L and a A_Λ. The first part takes $\boldsymbol{u}_{j-1}^n$ and $\boldsymbol{u}_j^n$ as input of both A_L and A_Λ and outputs $\boldsymbol{L}_{j-\frac{1}{2}}$ through A_L and $\Lambda_{j-\frac{1}{2}}$ through A_Λ. The input $\boldsymbol{u}_{j-1}^n$ and $\boldsymbol{u}_j^n$ is a vector $(\boldsymbol{u}_{j-1}^{n,(1)}, \cdots, \boldsymbol{u}_{j-1}^{n,(N_c)}, \boldsymbol{u}_j^{n,(1)}, \cdots, \boldsymbol{u}_j^{n,(N_c)})$ of length $2N_c$. The output matrix $L_{j-\frac{1}{2}}$ is of size $(N_c \times N_h)$, and the other output matrix $\boldsymbol{\Lambda}_{j-\frac{1}{2}}$ is a diagonal matrix of size $(N_h \times N_h)$. The second part takes $\boldsymbol{u}_j^n$ and $\boldsymbol{u}_{j+1}^n$ as input of both A_L and A_Λ and outputs $\boldsymbol{L}_{j+\frac{1}{2}}$ through A_L and $\boldsymbol{\Lambda}_{j+\frac{1}{2}}$ through A_Λ. The input u_j^n and u_{j+1}^n is a vector $(\boldsymbol{u}_j^{n,(1)}, \cdots, \boldsymbol{u}_j^{n,(N_c)}, \boldsymbol{u}_{j+1}^{n,(1)}, \cdots, \boldsymbol{u}_{j+1}^{n,(N_c)})$ of length $2N_c$. The output matrices $\boldsymbol{L}_{j+\frac{1}{2}}$ and $\boldsymbol{\Lambda}_{j+\frac{1}{2}}$ take

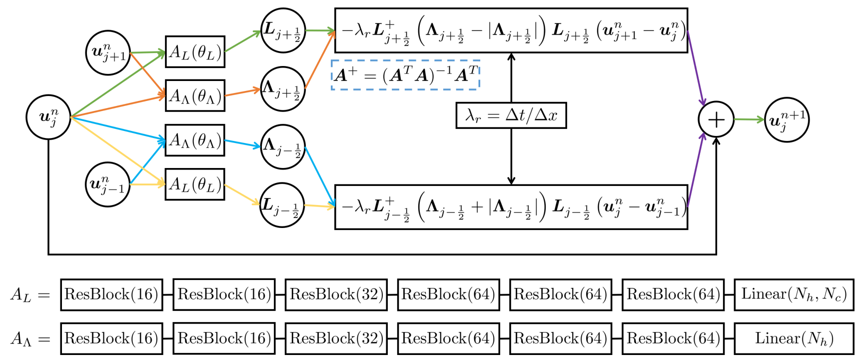


Figure 1: The architecture of RoeNet. RoeNet takes the current conserved quantity $\boldsymbol{u}_j^n$ and its direct neighbors, $\boldsymbol{u}_{j-1}^n$ and $\boldsymbol{u}_{j+1}^n$, as the input, and outputs the next conserved quantity $\boldsymbol{u}_j^{n+1}$. The ResBlock has the same architecture as in He et al. (2016) only with the 2D convolution layers replaced by linear layers. The numbers in the parentheses are output dimensions of each Resblock.

Table 1: Experimental set-up of four PDE problems.

	1C Linear	3C Linear	Sod tube	Inviscid Burgers
Boundary condition	periodic	Neumann	Neumann	periodic
Time step Δt	0.01	0.001	0.001	0.001
Space step Δx	0.01	0.005	0.005	0.01
Training time span	0.1	0.02	0.02	0.001
Predicting time span	2	0.2	0.1	0.3
Dataset samples	100	2000	2000	100
Dataset generation	analytical	analytical	analytical	2^{nd} central difference
Components number N_c	1	3	3	1
Hidden dimension N_h	1	16	32	64

the same form as the output matrices in the first part. Given the four output matrices $\boldsymbol{L}_{j-\frac{1}{2}}$, $\boldsymbol{\Lambda}_{j-\frac{1}{2}}$, $\boldsymbol{L}_{j+\frac{1}{2}}$, and $\boldsymbol{\Lambda}_{j+\frac{1}{2}}$, we combine them through (7) to obtain $\boldsymbol{u}_j^{n+1}$.

A_L and A_Λ both consist of a chain of ResBlock (He et al., 2016) with a linear layer at the end of size $N_h \times N_c$ and N_h, respectively. The N_h numbers learned by A_Λ is transferred into a diagonal matrix of $N_h \times N_h$ with the learned numbers as its diagonal. The ResBlock has the same architecture as in He et al. (2016) only with the 2D convolution layers replaced by linear layers. The numbers in the parentheses are output dimensions of each Resblock. Note that although we only show the calculation for grid cell j, the process is the same for grid cells. Since each node are calculated independent from the others except its closest neighbors, we train them in parallel to achieve high efficiency. In addition, to address different boundary conditions, we implement two ways of padding. For periodic boundary conditions, we use the periodic padding, e.g., if $j = 0$, $\boldsymbol{u}_{j-1} = \boldsymbol{u}_{N_g}$, where N_g is the number of grid node. For Neumann boundary conditions, we use the replicate padding, e.g., if $j = 0$, then we set $\boldsymbol{u}_{j-1}^n = \boldsymbol{u}_0^n$.

4 Experiments

We examine our model's ability of solving different kinds of hyperbolic PDEs. Most importantly, we show our model is capable of predicting discontinuity with smooth training data. The details of the parameters we set and important quantities about hidden layers can be found in Table 1. We use 2^{nd} central difference to generate dataset for the inviscid Burgers' equation, since there is no analytical solution for Burgers' equation. Note that 2^{nd} central difference will eventually lead to explosion. We only use the very first small time window, where the solution has not exploded yet, to train our model. For all the problems, the range of x we aim to solve are from -0.5 to 0.5.

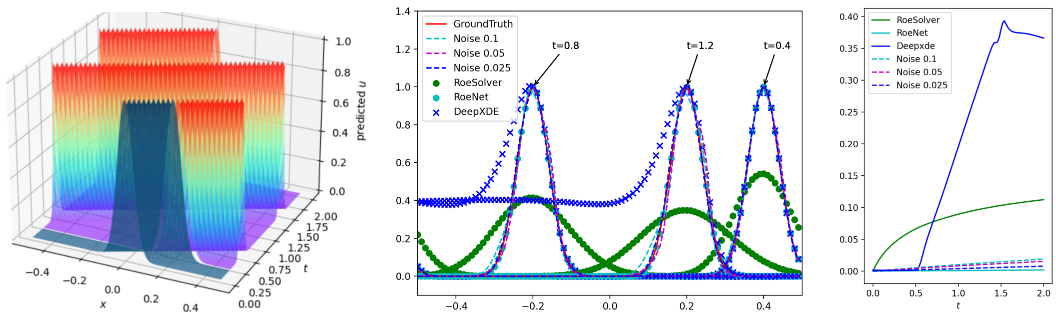


Figure 2: Results of solving an one component linear hyperbolic PDE. Left: Results of RoeNet; Middle: Comparisons ; Right: Errors compared with groundtruth. Notice the contrast between the small training region (dark blue) and the large future predictions (rainbow color).

For all experiments, we use the Adam optimizer Kingma & Ba (2014) with a learning rate 0.001. The learning rate decays with a ratio of 0.9 for every 5 epochs. We use a batch size of 16 for all experiments. We choose the Mean Squared Error as our loss function for all experiments. All the models are trained for 100 epochs and converge in less than 5 minutes in a single Nvidia RTX 2080Ti.

4.1 Hyperbolic PDEs with one component

We first show our model's ability of predicting the results of linear and non-linear hyperbolic PDEs with one components. The comparisons of our results with other PDE methods are also shown.

A simple example Figure 2 shows the predicting results of the linear hyperbolic PDE with one component (1C Linear)

$$\begin{cases} \boldsymbol{F} = x, \\ u(t=0,x) = e^{-300x^2}. \end{cases} \tag{9}$$

In Figure 2 (a), we plot the prediction results of RoeNet in 3D with time as an additional dimension. As show in this figure, the region of the wave peak of the predicted results (denoted by rainbow color) is completely different from the training data (denoted by blue color; $t \in [0.0, 0.2]$).

In Figure 2 (b), we plot the results using RoeNet, Roe solver, DeepXDE(Lu et al., 2019), as well as the exact solution. Note that for both Roe solver and DeepXDE, the euqations are known, and that we give DeepXDE training data of longer time period($t \in [0.0, 0.5]$). It is clear that RoeNet, even not knowing the equations, outperforms both these two traditional and deep learning PDE solvers, especially at larger t that is not included in training data.

Figure 2 (c) shows the averaged deviation $\lambda_u = \langle |u - u_{\text{exact}}| \rangle$ of the predicted solutions from the exact solution, where $\langle \cdot \rangle$ denotes the average over $[-0.5, 0.5]$. The averaged deviation of RoeNet is almost negligible, showing its high accuracy when making long time predictions.

Predict discontinuity with smooth training data In this example, we exhibit the unique ability of our model to accomplish tacks that traditional machine learning approaches fail to complete. Given a short window of continuous training data, we aim to use our model to predict long-term discontinuity of a nonlinear hyperbolic PDE, the inviscid Burgers' equation. Burgers' equation is a fundamental PDE occurring in various areas, such as fluid mechanics, nonlinear acoustics, gas dynamics, and traffic flow. The inviscid Burgers' equation is a conservation equation, more generally a first order quasilinear hyperbolic equation, which can develop discontinuities (shock waves) Burgers (1948). The set of equations is given by

$$\begin{cases} \boldsymbol{F} = \dfrac{1}{2}u^2, \\ u(t=0,x) = \dfrac{1}{2} + \sin(2\pi x). \end{cases} \tag{10}$$

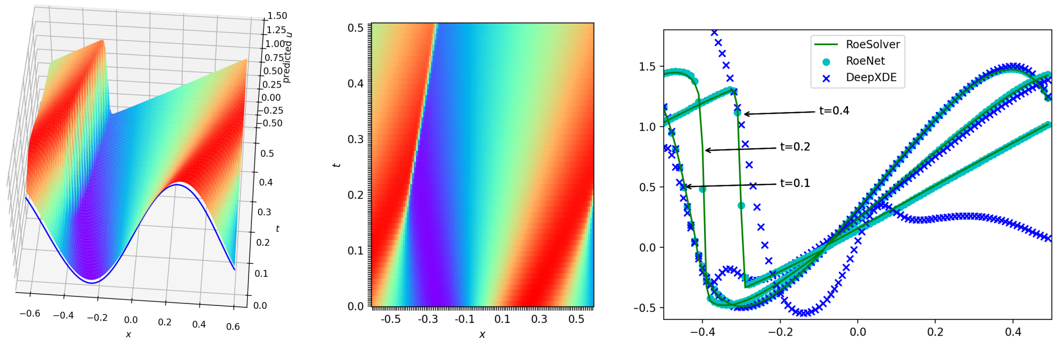


Figure 3: Results of solving an one component linear hyperbolic PDE. Left: Results of RoeNet; Middle: Heatmap to show the discontinuity ; Right: Comparisons.

Since there is no analytical solution for this problem, we plot only the prediction results made by RoeNet, Roe solver and DeepXDE(given training data of $t \in [0.0, 0.1]$) at $t = 0.1$, $t = 0.2$, and $t = 0.4$ in Figure 3 (c). Note that out training data for u is also an approximate solution generated by numerical method and only involves an extremely shorts time period of $t \in [0, 0.002]$, as shown in the thin blue curve in In Figure 3 (a). The perfect match of the predictions made by RoeNet with these made by Roe solver at all three time points shows that RoeNet successfully learn the future discontinuities of the problem based only on short-term continuous training data. This is a breakthrough improvement in solving prediction problems, as predicting long-term discontinuities from a short window of smooth training data is in general considered impossible using traditional machine learning approaches.

4.2 Hyperbolic PDEs with more component

In addition, we apply RoeNet to solve a linear hyperbolic PDE with three components (3C Linear)

$$\begin{cases} \boldsymbol{F} = \begin{bmatrix} 0.3237 & 2.705 & 5.4101 \\ 0.3597 & -0.4388 & -2.8777 \\ -0.0144 & 0.0576 & 1.1151 \end{bmatrix} \boldsymbol{x}, \\ \boldsymbol{u}(t=0, x \leq 0) = (0.4, 0.4, 0.4), \quad \boldsymbol{u}(t=0, x > 0) = (-0.4, -0.4, -0.4). \end{cases} \tag{11}$$

Figure 4 shows the the exact solutions and the prediction results of the three components $u^{(1)}$, $u^{(2)}$, and $u^{(3)}$ of a Riemann problem with linear flux function, at$t = 0.2$. From all three plots in Figure 4, we can observe that the predictions made by RoeNet match the exact solutions perfectly, while these of Roe solver have obvious errors around the discontinuous points (at $x \approx \pm 0.3$).

We then access the performance of our model on solving Riemann problems with nonlinear flux functions, which is in the form of equation 1 with $\boldsymbol{F}(\boldsymbol{u}) = \boldsymbol{A}(\boldsymbol{u})\boldsymbol{u}$. Specifically, we apply our model to the Sod shock tube problem Sod (1978), which is a one-dimensional Riemann problem in the following form

$$\begin{cases} \boldsymbol{u} = (\rho, \rho v, E)^T \\ \boldsymbol{F} = [\rho v, \rho v^2 + p, v(E+p)]^T, \\ (\rho, p, v)|_{t=0, x \leq 0} = (1, 1, 0), \quad (\rho, p, v)|_{t=0, x>0} = (0.125, 0.1, 0), \end{cases} \tag{12}$$

where ρ is the density, p is the pressure, E is the energy, and v is the velocity. The pressure, p, is related to the conserved quantities through the equation of state

$$p = (\gamma - 1)\left(e - \frac{1}{2}\rho v^2\right) \tag{13}$$

with $\gamma = 1.4$. The time evolution of this problem can be described by solving the Euler equations, which leads to three characteristics, describing the propagation speed of the various regions of the system. Namely the rarefaction wave, the contact discontinuity and the shock discontinuity Sod

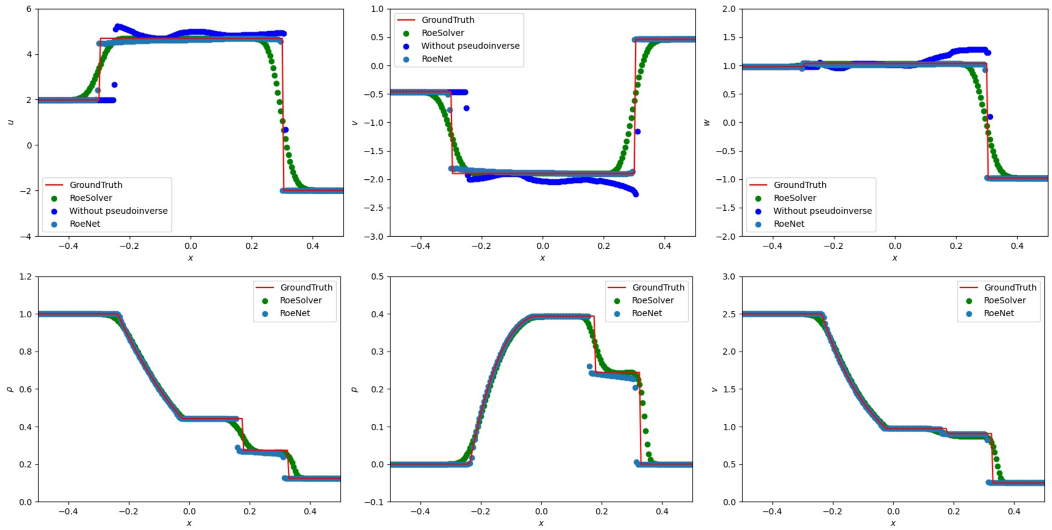


Figure 4: Riemann problem with three components and a linear (row 1) / non-linear (row 2) flux function. (a), (b), and (c) plot the prediction results using RoeNet and Roe solver, and exact solutions of the three components $u^{(1)}$, $u^{(2)}$, and $u^{(3)}$ respectively.

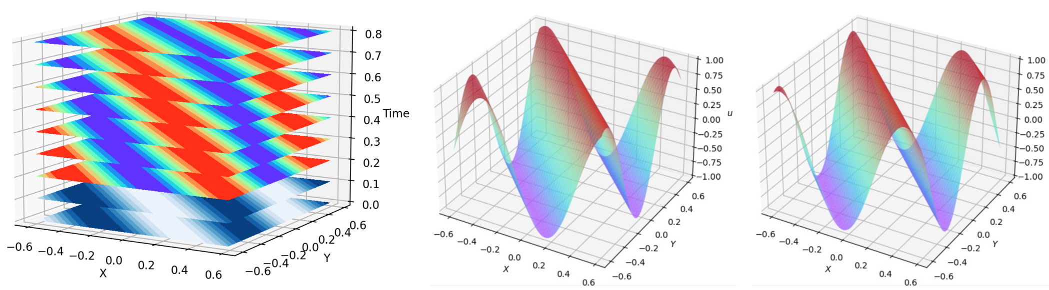


Figure 5: Results of 2D wave. Left: Heatmap of the 2D wave in different. Note that the blue part is the time range of training data, while the larger rainbow part are predicted by RoeNet; Middle and right: predicted results (rainbow) and ground truth(blue) at $t = 0.2$ and $t = 0.4$ respectively. The perfect match shows high accuracy of RoeNet.

(1978). In Figure 4, we plot the three components of the problem, at $t = 0.1$. Similar to the conclusion drawn from the previous section 4.1, RoeNet exhibits higher accuracy at predicting the discontinuities of the nonlinear Riemann problem.

4.3 2D Example

We show our model's ability of predicting higher dimensional PDEs by learning and predicting the behavior of a 2D wave, defined the following set of equations:

$$\begin{cases} \boldsymbol{F} = \boldsymbol{x}, \\ u(t = 0, \boldsymbol{x}) = \sin(2\pi(\boldsymbol{x}_1 + \boldsymbol{x}_2 + t)) \end{cases} \tag{14}$$

To solve 2 dimensional PDEs, we add another A_L and A_Λ networks to learn those matrix in the second dimension, and combine the vertical and horizontal Δu together to predict u^{t+1}. Similarly as our 1D examples, the 2D ReoNet can learn from a limited number and time range of training data and predict for a long time with high accuracy, as shown in Figure 5.

4.4 Ablation Study

Robustness to noisy data To test out model's robustness to noisy data, we use the 1D wave example and add different levels of noise to the training data. As shown in Figure 2 (b-c), our model can have higher accuracy than the compared methods even with noisy training data.

Influence of pseudoinverse We now examine the influence of pseudoinverse using the three component example. During training the three component models, the network can easily get NaN results if directly using the inverse matrix. We plot the comparison of inverse and pseudoinverse in Figure 4.

5 Related work

Riemann solvers In scientific computing, numerical methods discretize the hyperbolic problems into grids, which can be seen approximately as Riemann problems on a local scale (Griebel & Zumbusch, 1999; Colella et al., 2006; Vilara et al., 2011; McCorquodale & Colella, 2011; Spekreijse, 1987). In particular, the Riemann problem is a hyperbolic partial differential equation (PDE), with initial data comprised of two constant states, separated by a single discontinuity (Colella & Glaz, 1985). Integrating the Riemann problem into hyperbolic system simulation can be traced back to the work of Godunov Godunov (1959) with many follow-up works (see Toro, 1999). One of the most famous ones is the Roe solver, invented by Phil Roe in 1981, which is a linearized Riemann solver to improve the performance of Godunov's method regarding its performance and dissipation (see Quirk, 1997). More recently, Rotated-hybrid Riemann solvers were introduced by Hiroaki Nishikawa and Kitamura, in order to overcome the carbuncle problems of the Roe solver and the excessive diffusion of the HLLE solver at the same time Nishikawa & Kitamura (2008).

Deep learning solvers Approximating discontinuous functions with deep learning network has theoretical foundation in various literature, e.g. Yarosky (2017) work on the Hölder space, Petersen & Voigtländer (2017) on piece-wise smooth functions, Imaizumi & Fukumizu (2019) on DNN outperforming linear estimators and Suzuki (2019) study on deep learning's higher adaptivity to spatial inhomogeneity of the target function. With above-mentioned theoretical cornerstone, a Physics Informed Neural Network (PINN) is proposed by Raissi et al. (2017) to provide data-driven solutions to nonlinear problems, employing the well-known capacity of Deep Neural Networks (DNN) as universal function approximators (Hornik et al., 1989). Among its notable features, PINN maintains symmetry, invariance and conservation principles deriving from physical laws that governs observed data (Zhang et al., 2019). Michoski et al. (2019) show that without any regularization, irregular solutions to PDE can be captured. Mao et al. (2020b) used PINN to approximate solutions to high-speed flows by formulating the Euler equation and initial/boundary conditions into the loss function. However, in Mao's setting, PINN does not solve the forward problems as accurately as the traditional numerical methods. By incorporating invariants and data *a priori* known to loss functions, such DNNs are also less adaptive to different kinds of problems.

6 Conclusion

We presented Roe Neural Networks (RoeNets) to predict long-term discontinuity based on partial and smooth observation. Our numerical experiments showed that RoeNet outperforms both the traditional Riemann solver and modern deep learning solvers regarding its accuracy, robustness, and ability to predict invisible discontinuity in the future. We further demonstrated in our ablation tests that these computational merits draw from our templaterized prior embedding whose scheme preserves the essential mathematical properties of the original Roe template. To the best of our knowledge, our method is the first step toward building prior-embedded machine learning methods to predict long-term future dynamic behaviors that are invisible in the current training data. A broad range of applications outside physical prediction remains to be explored. On another hand, the design of our templaterized architecture that hybrids modulerized mathematical priors and data-driven paradigms opens up a new door to embed structured priors into networks to tackle physical problems. More broadly speaking, any numerical stencil generators or design principles can serve as a potential prior template that can be used to empower high-performance learning pipelines. We envision a series of future work based on our templaterizable prior embedding scheme to bridge scientific computing and machine learning, such as templaterizing the high-order advection schemes in CFD.

REFERENCES

A. Bressan. *Hyperbolic Conservation Laws*, pp. 157–245. Springer, Berlin, Heidelberg, 2013.

J. M. Burgers. A mathematical model illustrating the theory of turbulence. *Adv. Appl. Mech.*, 1: 171–199, 1948.

P. Colella, D. T. Graves, B. J. Keen, and D. Modiano. A Cartesian grid embedded boundary method for hyperbolic conservation laws. *J. Comput. Phys.*, 211:347 – 366, 2006.

Phillip Colella and Harland M Glaz. Efficient solution algorithms for the riemann problem for real gases. *Journal of Computational Physics*, 59(2):264–289, 1985.

A. Dam and P. A. Zegeling. A robust moving mesh finite volume method applied to 1D hyperbolic conservation laws from magnetohydrodynamics. *J. Comput. Phys.*, 216:526–546, 2006.

Robert Eymard, Thierry Gallouët, and Raphaèle Herbin. Existence and uniqueness of the entropy solution to a nonlinear hyperbolic equation. 1995.

S. K. Godunov. A difference method for numerical calculation of discontinuous solutions of the equations of hydrodynamics. *Matematicheskii Sbornik*, 89:271–306, 1959.

M. Griebel and G. Zumbusch. Adaptive sparse grids for hyperbolic conservation laws. In *Hyperbolic Problems: Theory, Numerics, Applications*, pp. 411–422, 1999.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of The IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

K. Hornik, M. Stinchcombe, and W. Halbert. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359 – 366, 1989.

M. Imaizumi and K. Fukumizu. Deep learning networks learn non-smooth functions effectively. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 869–878, 2019.

Malte Jahn. Artificial neural network regression models in a panel setting: Predicting economic growth. *Economic Modelling*, 91:148–154, 2020.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

Lu Lu, Xuhui Meng, Zhiping Mao, and George E. Karniadakis. DeepXDE: A deep learning library for solving differential equations. *arXiv preprint arXiv:1907.04502*, 2019.

F. Mao, L. L. Kang, J. Z. Wu, J.-L. Yu, A. K. Gao, W. D. Su, and X.-Y. Lu. A study of longitudinal processes and interactions in compressible viscous flows. *J. Fluid Mech.*, 893:A23, 2020a.

Z. Mao, A. D. Jagtap, and G. E. Karniadakis. Physics-informed neural networks for high-speed flows. *Comput. Method. Appl. M.*, pp. 112789, 2020b.

P. McCorquodale and P. Colella. A high-order finite-volume method for conservation laws on locally refined grids. *Comm. App. Math. Com. Sc.*, 6:1–25, 2011.

Kittipong Methaprayoon, Wei-Jen Lee, Sothaya Rasmiddatta, James R Liao, and Richard J Ross. Multistage artificial neural network short-term load forecasting engine with front-end weather forecast. *IEEE Transactions on Industry Applications*, 43(6):1410–1416, 2007.

C. Michoski, M. Milosavljevic, T. Oliver, and D. Hatch. Solving irregular and data-enriched differential equations using deep neural networks. *Xiv:1905.04351*, 2019.

H. Nishikawa and K. Kitamura. Very simple, carbuncle-free, boundary-layer-resolving, rotated-hybrid riemann solvers. *J. Comput. Phys.*, 227:2560–2581, 2008.

P. Petersen and F. Voigtländer. Optimal approximation of piecewise smooth functions using deep relu neural networks. *Neural Networks*, 09 2017. doi: 10.1016/j.neunet.2018.08.019.

James J Quirk. A contribution to the great riemann solver debate. In *Upwind and High-Resolution Schemes*, pp. 550–569. Springer, 1997.

M. Raissi, P. Perdikaris, and G. E. Karniadakis. Inferring solutions of differential equations using noisy multi-fidelity data. *J. Comput. Phys.*, 335:736–746, 2017.

P. L. Roe. Approximate riemann solvers, parameter vectors and difference schemes. *J. Comput. Phys.*, 43:357–372, 1981.

W. Scheid, H. Muller, and W. Greiner. Nuclear shock waves in heavy-ion collisions. *Phys. Rev. Lett.*, 32:741–745, 1974.

G. A. Sod. A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws. *J. Comput. Phys.*, 27:1–31, 1978.

S. Spekreijse. Multigrid solution of monotone second-order discretizations of hyperbolic conservation laws. *Math. Comput.*, 49:135–155, 1987.

T. Suzuki. Adaptivity of deep relu network for learning in besov and mixed smooth besov spaces: Optimal rate and curse of dimensionality. In *International Conference on Learning Representations*, 2019.

K. Terao and T. Inagaki. Interaction between combustion and shock waves. *Jpn. J. Appl. Phys.*, 28: 1226 – 1234, 1989.

E. F. Toro, M. Spruce, and W. Speares. Restoration of the contact surface in the HLL-Riemann solver. *Shock waves*, 4:25–34, 1994.

EF Toro. Riemann solvers and numerical methods for fluid dynamics–springer. *Heidelberg, 624pp*, 1999.

Eleuterio F Toro. *Riemann solvers and numerical methods for fluid dynamics: a practical introduction*. Springer Science & Business Media, 2013.

Terry Troudet, Sanjay Garg, and Walter Merrill. Neural network application to aircraft control system design. In *Navigation and Control Conference*, pp. 2715, 1991.

F. Vilara, P. Mairea, and R. Abgrall. Cell-centered discontinuous Galerkin discretizations for two-dimensional scalar conservation laws on unstructured grids and for one-dimensional Lagrangian hydrodynamics. *Comput. Fluids*, 46:498–504, 2011.

D. Yarosky. Error bounds for approximations with deep relu networks. *Neural Networks*, 94:103–114, 2017.

Ao Yu, Hui Yang, Qiuyan Yao, Kaixuan Zhan, Bowen Bao, Zhengjie Sun, and Jie Zhang. Traffic scheduling based on spiking neural network in hybrid e/o switching intra-datacenter networks. In *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pp. 1–7. IEEE, 2020.

D. Zhang, L. Guo, and G. E. Karniadakis. Learning in modal space: solving time-dependent stochastic PDEs using physics-informed neural networks. *SIAM J. Sci. Comput.*, 42:A639–A665, 2019.

A DETAILED INTRODUCTION OF ROE SOLVER

The Roe solver Roe (1981) discretizes (1) as

$$\boldsymbol{u}_j^{n+1} = \boldsymbol{u}_j^n - \lambda_r \left(\hat{\boldsymbol{F}}_{j+\frac{1}{2}}^n - \hat{\boldsymbol{F}}_{j-\frac{1}{2}}^n \right), \tag{15}$$

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

$$\hat{\boldsymbol{F}}_{j+\frac{1}{2}}^n = \hat{\boldsymbol{F}}(\boldsymbol{u}_j^n, \boldsymbol{u}_{j+1}^n) \tag{16}$$

with

$$\hat{\boldsymbol{F}}(\boldsymbol{u}, \boldsymbol{v}) = \frac{1}{2} \left[\boldsymbol{F}(\boldsymbol{u}) + \boldsymbol{F}(\boldsymbol{v}) - |\tilde{\boldsymbol{A}}(\boldsymbol{u}, \boldsymbol{v})|(\boldsymbol{v} - \boldsymbol{u}) \right]. \tag{17}$$

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

1. Diagonalizable with real eigenvalues: ensures that the new linear system is truly hyperbolic.
2. Consistency with the exact Jacobian: when $\boldsymbol{u}_j, \boldsymbol{u}_{j+1} \to \boldsymbol{u}$, we demand that $\tilde{\boldsymbol{A}}(\boldsymbol{u}_j, \boldsymbol{u}_{j+1}) = \partial \boldsymbol{F}(\boldsymbol{u})/\partial \boldsymbol{u}$.
3. Conserving $\boldsymbol{F}_{j+1} - \boldsymbol{F}_j = \tilde{\boldsymbol{A}}(\boldsymbol{u}_{j+1} - \boldsymbol{u}_j)$.

From the first Roe condition, matrix $\tilde{\boldsymbol{A}}$ can be diagonalized as

$$\tilde{\boldsymbol{A}} = \boldsymbol{L}^{-1} \boldsymbol{\Lambda} \boldsymbol{L}. \tag{18}$$

Therefore, $|\tilde{\boldsymbol{A}}(\boldsymbol{u}, \boldsymbol{v})|$ can be interpreted as

$$|\tilde{\boldsymbol{A}}| = \boldsymbol{L}^{-1} |\boldsymbol{\Lambda}| \boldsymbol{L}. \tag{19}$$

Substituting (16), (17) and (19) into (15) along with the third Roe condition yields

$$\begin{aligned} \boldsymbol{u}_j^{n+1} = & \boldsymbol{u}_j^n - \frac{1}{2}\lambda_r [\boldsymbol{L}_{j+\frac{1}{2}}^{-1} (\boldsymbol{\Lambda}_{j+\frac{1}{2}} - |\boldsymbol{\Lambda}_{j+\frac{1}{2}}|) \boldsymbol{L}_{j+\frac{1}{2}} (\boldsymbol{u}_{j+1}^n - \boldsymbol{u}_j^n) \\ & + \boldsymbol{L}_{j-\frac{1}{2}}^{-1} (\boldsymbol{\Lambda}_{j-\frac{1}{2}} + |\boldsymbol{\Lambda}_{j-\frac{1}{2}}|) \boldsymbol{L}_{j-\frac{1}{2}} (\boldsymbol{u}_j^n - \boldsymbol{u}_{j-1}^n)], \end{aligned} \tag{20}$$

with

$$\boldsymbol{L}_{j+\frac{1}{2}}^n = \boldsymbol{L}(\boldsymbol{u}_j^n, \boldsymbol{u}_{j+1}^n), \quad \boldsymbol{\Lambda}_{j+\frac{1}{2}}^n = \boldsymbol{\Lambda}(\boldsymbol{u}_j^n, \boldsymbol{u}_{j+1}^n). \tag{21}$$

(20) serves as a template of evolution from $\boldsymbol{u}_j^n$ to $\boldsymbol{u}_j^{n+1}$.

In order to construct a Roe matrix $\tilde{\boldsymbol{A}}$ that follows the Roe conditions, 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 (20) to ultimately solve for $\boldsymbol{u}$ in (1). The Roe solver made a 'smart' linearization of the Riemann problem, which is computationally efficient while still recognizing the non-linear jumps in the problem. Compared with the other Riemann solvers, e.g. Godunov's method Godunov (1959) and HLLC solver Toro et al. (1994), it performs with less cost and less dissipation.