Proceedings of Machine Learning Research vol 145:1–17, 2021 2nd Annual Conference on Mathematical and Scientific Machine Learning

Implicit Form Neural Network for Learning Scalar Hyperbolic Conservation Laws

Xiaoping Zhang XPZHANG.MATH@WHU.EDU.CN
Tao Cheng TAOCHENG77@WHU.EDU.CN
School of Mathematics and Statistics, Wuhan University, Wuhan, Hubei 430072, China.

Lili Ju JU@MATH.SC.EDU
Department of Mathematics, University of South Carolina, Columbia, SC 29208, USA

Editors: Joan Bruna, Jan S Hesthaven, Lenka Zdeborova

Abstract

Conservation laws are critical to our understanding of the physical world. Numerical solution of hyperbolic conservation laws has been a very important and challenging task with some difficulties such as nonphysical solutions, solution discontinuities and shock wave capturing, and a large amount of conventional numerical solvers of finite difference, finite volume and discontinuous Galerkin types have been developed in the past decades. Along with the booming and great success of deep learning in computer vision and natural language processing, many excellent works also have emerged for their application to PDE related scientific problems in recent years. In this paper, we propose a deep learning method for solving scalar hyperbolic conservation laws. More specifically, we design a neural network model in the unsupervised fashion, called "IFNN", based on a special implicit form for the solution of the problem. The proposed IFNN does not directly process the target PDEs, instead it deals with the nonlinear equations characterizing implicitly the solution of the PDEs. Numerical experiments and performance comparisons of our IFNN with the well-known PINN are performed on two well-known problems under various boundary conditions, the inviscid Burgers' equation and the Lighthill-Whitham-Richards (LWR) model for traffic flow, and the results show that IFNN can significantly outperform PINN in capturing the shock waves.

Keywords: Scalar conservation laws, implicit form, deep learning, neural network

1. Introduction

Along with the fast advance of computational power (especially GPUs) and the explosive growth of usable data, deep learning has achieved great success in the past decades in many fields (Krizhevsky et al., 2012; He et al., 2016; Ioffe and Szegedy, 2015; Devlin et al., 2019; Sutskever et al., 2014), such as computer vision, natural language processing, gaming, fraud detection and data mining. In recent years, the deep learning techniques also have been applied to mathematical and scientific computation problems, and a large number of remarkable works have emerged on learning and predicting solutions of partial differential equations (E and Yu, 2018; Han et al., 2018; Raissi and Karniadakis, 2018; Raissi et al., 2018; Sirignano and Spiliopoulos, 2018; Berg and Nystrom, 2018; Raissi et al., 2019; Sun et al., 2020). Although there exist a lot of mature traditional methods for solving PDEs, such as finite element method, finite difference method, and finite volume method, the deep learning based methods provide a completely new perspective through the use of neural networks. The first attempt of numericallly solving PDEs using neural networks can be traced back at least to the late last century (Lee and Kang, 1990), where the approximate solution is modeled

through the Hopfield neural network. However, due to the limitation of computational resource back that time, it unfortunately did not attract much attention of researchers. Currently there are two major ways to use deep learning for numerical solutions of PDEs.

The first way is to convert the PDEs into their equivalent variational forms and then solve them using deep learning approaches. The pioneering work is the deep Ritz method proposed by E and Yu (2018), followed by a series of researches (Liao and Ming, 2019; Müller and Zeinhofer, 2020; Wang and Zhang, 2020; Chen et al., 2020). The deep Nitsche method (Liao and Ming, 2019) extended the deep Ritz method to deal with essential boundary conditions without extra computational costs. (Chen et al., 2020) carried out the comparison study for elliptic problems with different boundary conditions, including Dirichlet, Neumann, Robin, and periodic boundary conditions by using the deep Ritz and the deep Galerkin methods (Sirignano and Spiliopoulos, 2018), and observed that the deep Galerkin method works better for problems with smooth solutions while the deep Ritz method works better for problems with low-regularity solutions. Based on the deep Ritz Method, Wang and Zhang (2020) proposed a mesh-free method to solve the interface problems, where two challenging interface problems, including an elliptic PDE with a discontinuous and heterogeneous diffusion coefficient and a linear elasticity equation with discontinous strees tensor, were investigated. In terms of theoretical analysis, Müller and Zeinhofer (2020) obtained the convergence of the deep Ritz method with respect to a regularized Dirichlet energy towards the solution of the Poisson problem, in which ReLU is used as the activation function,

Another way is to use neural networks to directly deal with the original PDEs, and the most representative work is the physical informed neural network (PINN) proposed by Raissi et al. (2019), where the PDE residual is incorporated into the loss function as a regularizer, thereby constraining the space of admissible solutions. PINNs have been successfully applied to solve various PDE problems, such as classic time-independent or time-dependent PDEs, fractional PDEs (Pang et al., 2019), and stochastic PDEs (Zhang et al., 2020). It should be noted that the original PINN uses the residual of PDEs as the loss function, which is called the "soft constraint". Different from the soft constraints, (Berg and Nystrom, 2018; Sun et al., 2020) proposed some "hard constraints" to make the output of the neural networks satisfy the boundary and initial conditions precisely. However, for high-dimensional problems, especially for time-dependent problems, the amount of data sampling tends to become huge, which makes the training of PINN difficult, even intractable. In order to overcome this issue, the parallel PINNs have been proposed by Jagtap et al. (2020) and Meng et al. (2020), where the solution area was firstly divided into several subregions, and then respective neural networks for different subregions were trained in parallel. In addition, a series of excellent works (Yang and Paris, 2019; Liu et al., 2020, 2021) about uncertainty quantifications on PDEs by using PINNs emerged recently.

Although the deep Ritz method and PINN have achieved great success, they still exhibit poor performance in handling problems with strong discontinuous solutions, such as hyperbolic conservation laws. For example, when they are used to tackle Burgers' equation, an additional viscous term is usually added to ensure certain smoothness of the solution (Raissi et al., 2019), i.e., produce a viscous solution as the approximation of the original soltuion. On the other hand, if the viscous term vanishes or the viscosity coefficient is very small, both the deep Ritz method and PINN become very hard to train and fail to capture the shock waves. A deep reinforcement learning method was recently proposed by Wang et al. (2020) to solve 1D scalar conservation laws. Other than these, no much work has been done on solving the conservation law using deep learning techniques.

In this paper, we propose a deep learning method for solving scalar conservation laws, called "IFNN", which is based on the fact the solution of scalar conservation laws can be expressed alternately in an implicit form. Through extensive numerical tests and comarisons, we demonstrate that the proposed IFNN can significantly outperform PINN in capturing the shock waves. The rest of our paper is organized as follows. The problem setting, the architecture, the data sampling strategy and the loss function for our IFNN are illustrated in Section 2, and numerical experiments in 1D and 2D are conducted and discussed in Section 3, followed by concluding remarks in Section 4.

2. Implicit form neural network for solving scalar hyperbolic conservation laws

2.1. Problem settings

Let $\boldsymbol{x} = (x_1, \cdots, x_d)^T \in \mathbb{R}^d$ and $\boldsymbol{f} = (f_1, \cdots, f_d)^T \in \mathbb{R}^d$, the scalar hyperbolic conservation law can be formulated as follows :

$$\begin{cases} u_t + \nabla \cdot \boldsymbol{f}(u) = 0, \quad (\boldsymbol{x}, t) \in \mathbb{R}^d \times (0, T], & \text{(1a)} \\ u(\boldsymbol{x}, 0) = \phi(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^d. & \text{(1b)} \end{cases}$$

where u is unknown function defined in $\mathbb{R}^d$. There may not exist a classical continuous solution for the above equation even if the initial value is smooth, which often causes difficulty to its numerical solution. An implicit form for the solution of (1) also can be formulated as

$$u = \phi(\boldsymbol{x} - \boldsymbol{f}'(u)t), \tag{2}$$

where $\boldsymbol{f}'$ denotes the velocity

$$\boldsymbol{f}'(u) = (f_1'(u), \cdots, f_d'(u))^T. \tag{3}$$

In practice, we are often interested in the equation (1) defined in a finite domain $\Omega \subset \mathbb{R}^d$, and assume that the boundary condition is imposed by either the Dirichlet type as

$$u(\boldsymbol{x}, t) = g_{db}(\boldsymbol{x}, t), \quad (\boldsymbol{x}, t) \in \partial\Omega \times (0, T], \tag{4}$$

or the Neumann type as

$$\frac{\partial u}{\partial n}(\boldsymbol{x}, t) = g_{nb}(\boldsymbol{x}, t), \quad (\boldsymbol{x}, t) \in \partial\Omega \times (0, T], \tag{5}$$

or the periodic type (for rectangular domains only). We will construct a deep feedforward network, "**IFNN**" (implicit form neural network), to learn the solution of (1) in the unsupervised fashion. In particular, the loss function for training of the proposed IFNN is based on the implicit form (2) instead of the original equation (1).

2.2. Network architecture

The architecture of IFNN is shown in Figure 1 by using a two-dimensional problem setting for illustration. The vector $(\boldsymbol{x}, t)$ is fed as the input to the network, and the layer is connected to $D-1$ hidden layers and an output layer, which form a fully-connected neural network of depth D. Letting

n_k be the number of neurons in the k-th layer, each hidden layer recieves an input $\boldsymbol{v}^{k-1} \in \mathbb{R}^{n_{k-1}}$ from the previous layer output and transforms it to

$$\ell_k(\boldsymbol{v}^{k-1}) = \boldsymbol{W}^k \boldsymbol{v}^{k-1} + \boldsymbol{b}^k, \tag{6}$$

where $\boldsymbol{W}^k \in \mathbb{R}^{n_k \times n_{k-1}}$ and $\boldsymbol{b} \in \mathbb{R}^{n_k}$. The nonlinear activation function $\sigma(\cdot)$ is applied to each component of the transformed vector before sending it to the next layer, except the last hidden layer. The network thus is a composite of sequence of nonlinear functions:

$$u(\boldsymbol{x}, t; \Theta) = (\ell_D \circ \sigma \circ \ell_{D-1} \cdots \circ \sigma \circ \ell_1)(\boldsymbol{x}, t), \tag{7}$$

where the operator $\circ$ denotes the composition and $\Theta = \{\boldsymbol{W}^k, \boldsymbol{b}^k\}_{k=1}^D$ represents the trainable parameters in the network. In addition, to make the network easier to train, a skip connection is added between each pair of two adjacent hidden layers of our IFNN, which is shown in Fig. 2.

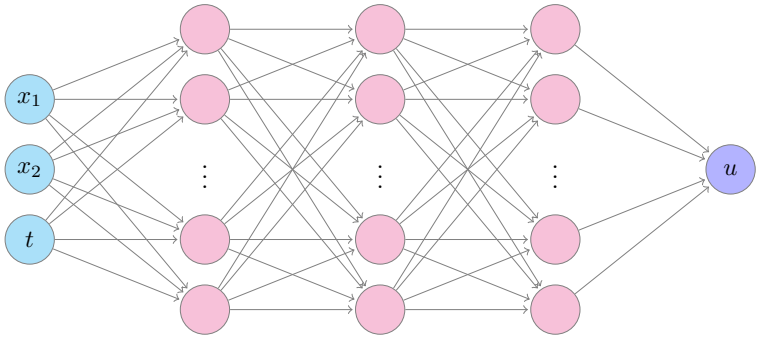


Figure 1: The architecture of the proposed IFNN in a two-dimensional problem setting.

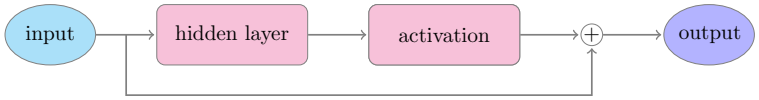


Figure 2: The skip connection between two adjacent hidden layers in the proposed IFNN.

2.3. The sampling strategy and loss function

To train the network (7), we first need a set of sampling points with respect to $(\boldsymbol{x}, t)$. In this paper, we use the Latin Hypercubic Sampling (LHS) (Raissi et al., 2019) to generate the point set $\mathcal{D}$, and then split it into three parts $\mathcal{D}_{\rm IC} = \mathcal{D} \cap (\Omega \times \{0\})$, $\mathcal{D}_{\rm BC} = \mathcal{D} \cap (\partial\Omega \times (0, T])$, and $\mathcal{D}_{\rm IM} = \mathcal{D} \backslash (\mathcal{D}_{\rm IC} \cup \mathcal{D}_{BC})$. The proposed IFNN is then trained by minimizing the following loss function:

$$\mathcal{L}(\Theta) = \mathcal{L}_{\rm IM}(\Theta) + \lambda_1 \mathcal{L}_{\rm IC}(\Theta) + \lambda_2 \mathcal{L}_{\rm BC}(\Theta), \tag{8}$$

where

$$\mathcal{L}_{\rm IM}(\Theta) = \frac{1}{|\mathcal{D}_{\rm IM}|} \sum_{(\boldsymbol{x},t) \in \mathcal{D}_{\rm IM}} \left[u(\boldsymbol{x}, t; \Theta) - \phi(\boldsymbol{x} - \boldsymbol{f}'(u)t)\right]^2, \tag{9}$$

$$\mathcal{L}_{\rm IC}(\Theta) = \frac{1}{|\mathcal{D}_{\rm IC}|} \sum_{(\boldsymbol{x},t) \in \mathcal{D}_{\rm IC}} [u(\boldsymbol{x}, t; \Theta) - \phi(\boldsymbol{x})]^2. \tag{10}$$

Regarding the loss from the boundary conditions, we define

$$\mathcal{L}_{\mathrm{BC}}(\Theta) = \frac{1}{|\mathcal{D}_{\mathrm{BC}}|} \sum_{(\boldsymbol{x},t)\in\mathcal{D}_{\mathrm{BC}}} \left[u(\boldsymbol{x},t;\Theta) - g_{db}(\boldsymbol{x},t)\right]^2 \tag{11}$$

for the cased of Dirichlet boundary condition (4) and

$$\mathcal{L}_{\mathrm{BC}}(\Theta) = \frac{1}{|\mathcal{D}_{\mathrm{BC}}|} \sum_{(\boldsymbol{x},t)\in\mathcal{D}_{\mathrm{BC}}} \left[\frac{\partial u}{\partial n}(\boldsymbol{x},t;\Theta) - g_{nb}(\boldsymbol{x},t)\right]^2 \tag{12}$$

for the case of Neumann boundary condition (5). For the case of periodic boundary condition over a rectangular domain Ω, we set

$$\mathcal{L}_{\mathrm{BC}}(\Theta) = \frac{1}{|\mathcal{D}_{\mathrm{BC}}|} \sum_{(\boldsymbol{x},t)\in\mathcal{D}_{\mathrm{BC}}} \left[u(\boldsymbol{x},t;\Theta) - u(\boldsymbol{x}',t;\Theta)\right]^2, \tag{13}$$

where $\boldsymbol{x}'$ is the symmetric point of $\boldsymbol{x}$ in the opposite boundary side of Ω. Note that $\mathcal{L}_{\mathrm{IM}}(\Theta)$ denotes the residual computed based on the implicit form (2), $\mathcal{L}_{\mathrm{IC}}$ and $\mathcal{L}_{\mathrm{BC}}$ measure the initial and boundary errors, respectively, λ_1 and λ_2 are two hyperparameters for balancing the three terms.

3. Numerical experiments

Our IFNN is implemented using PyTorch and we use 6 hidden layers with 20 neurons per each hidden layer for all tests if not stated otherwise. Adam optimizer is used to train the network and the number of epochs is set as 25000. The learning rate is initially set to 0.001, and then adjusted with a decay rate of 0.7 per 3000 epochs. The hyperparamters λ_1 and λ_2 in (8) are both chosen as 1. Several scalar conservation laws examples, including the inviscid Burgers' equation and the LWR model for traffic flow problem, are investigated to demonstrate the performance of IFNN. For the purpose of comparison, we also train and test the PINN with the same network architecture.

3.1. 1D inviscid Burgers' equation

By choosing $d = 1$ and $f(u) = u^2/2$ in (1), we have the 1D inviscid Burgers' equation:

$$\begin{cases} u_t + uu_x = 0, & x \in \mathbb{R}, t > 0, \\ u(x,0) = \phi(x), & x \in \mathbb{R}. \end{cases} \tag{14}$$

It is one of the most important PDEs in the theory of conservation laws and often used to model the momentum transfer in fluids with uniform density and pressure. However, the nonlinear hyperbolic nature of the equation (14) may produce shock over time, even if the initial condition is smooth.

Shock wave (I) In this case, we choose $\Omega = [-1, 1]$, the initial value $\phi(x)$ as a step function:

$$\phi(x) = \begin{cases} 1, & x \leqslant 0, \\ 0, & x > 0, \end{cases} \tag{15}$$

and the Dirichlet boundary condition $u(-1,t)=2$ for $t\in(0,1]$. The solution u is constant along the projected characteristic curves given by

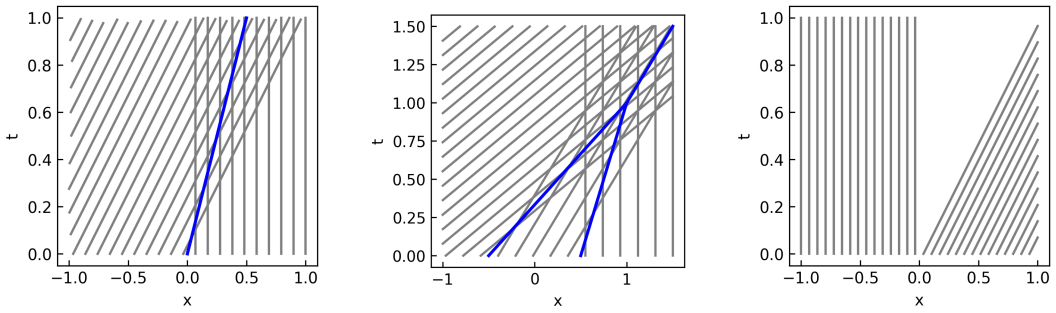
$$x=\phi(r)t+r=\begin{cases} t+r, & r<0,\\ r, & r>0, \end{cases} \tag{16}$$

which is shown in Fig. 3-(a). Obviously, the weak solution of (14) is discontinuous across certain curve $x=\xi(t)$ but smooth on either side of this curve , and it satisfies the Rankine-Hugoniot jump condition

$$\xi'(t)=\frac{f(u^-)-f(u^+)}{u^--u^+}=\frac{u^-+u^+}{2}=\frac{1}{2}. \tag{17}$$

Therefore, the curve of discontinuity is given by $x=t/2$, and the weak soution of (14) in this case can be formulated as follows:

$$u(x,t)=\begin{cases} 1, & x<t/2,\\ 0, & x>t/2. \end{cases} \tag{18}$$



(a) shock wave (I) (b) shock wave(II) (c) rarefaction wave

Figure 3: Projected characteristic curve (gray line) and the curve of discontinuity (blue line) of the inviscid Burgers' equation (14) with different initial conditions.

It is known that the training of PINN (without adding an extra viscous term) is quite difficult when the solution has strong discontinuities. We first consider the effect of the network's architecture on the prediction accuracy of IFNN, and the corresponding results are listed in Table 1 (IFNN with skip connection is used), from which we see that with the increase of network's depth and width, the prediction accuracy of IFNN gets better. To compromise the network's complexity with the prediction performance, we choose 6×20 as our network architecture as stated at the beginning of this section. Next, we consider the effect of the size of the training set, and we find from Fig. 4 that with the increase of the size of the training set, the prediction error decreases until the size of the set $\mathcal{D}_{\mathrm{IM}}$ reaches about 10000, and then the continuous increase of the data set does not improve the prediction accuracy. Then, we take the hyperbolic tangent ($\tanh$) as the activation function for both the proposed IFNN and PINN and the corresponding loss curves from the training process are presented in Fig. 5, which shows that the training of IFNN, with or without the skip connection, is much easier than that of PINN, and the skip connection used in our IFNN does further help to achieve even smaller loss. Therefore, we use IFNN with the skip connection (abbreviated as IFNN)

to conduct all the following tests. The relative L_2 errors of the solutions predicted by IFNN and PINN with different activation function are reported in Table 2, from which we can see that no matter what kind of activation function is selected, the performance of IFNN is always better than PINN, and moreover, $\tanh$ works significantly better than the other two for both IFNN and PINN in this case. A more detailed assessment of the predicted solutions is shown in Fig. 6. From the top panel of Fig. 6 we see that the predicted solution by IFNN is essentially in agreement with the reference solution in the whole spatial-temporal domain, and the comparison results from the bottom panel of Fig. 6 show our IFNN can capture the shock very well for all times, while the PINN does not especially when $t = 0.25, 0.5$.

Table 1: Relative L_2 errors of the predicted solutions by IFNN with skip connection with different number of hidden layers ($\#l$) and neurons per each layer ($\#n$).

$\#l$ \ $\#n$	10	20	30
2	5.76×10^{-2}	2.33×10^{-2}	9.23×10^{-3}
4	8.91×10^{-3}	2.72×10^{-3}	6.71×10^{-4}
6	2.97×10^{-3}	2.50×10^{-4}	2.32×10^{-4}

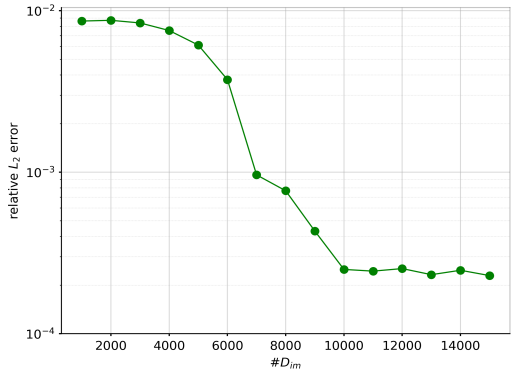
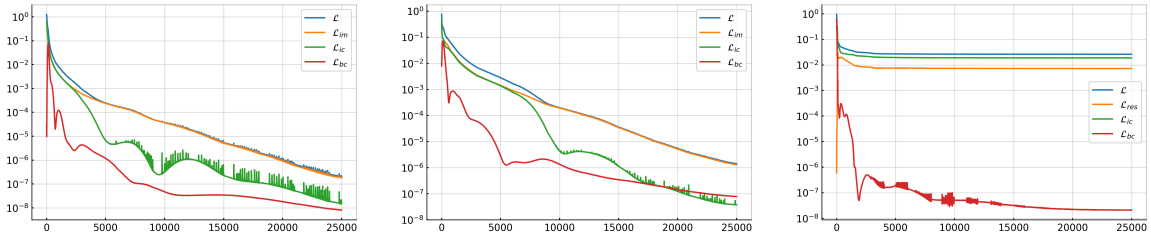
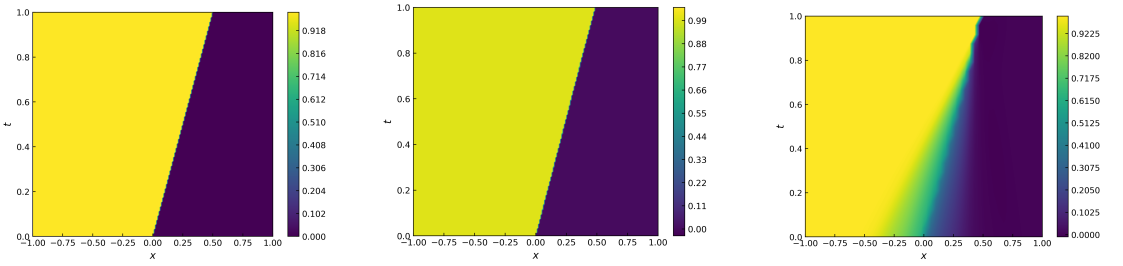


Figure 4: The effect of the size of training set on the prediction accuracy of IFNN.

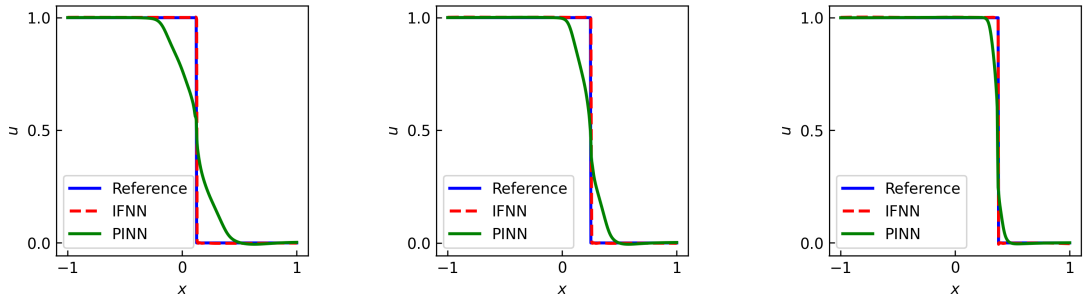


(a) IFNN w skip (b) IFNN w/o skip (c) PINN

Figure 5: The loss curves of IFNN, including the total loss and each of its terms during the training for the 1D inviscid Burgers' equation (14) – Shock wave (I).



(a) reference solution (b) predicted solution by IFNN (c) predicted solution by PINN



(d) $t = 0.25$ (e) $t = 0.5$ (f) $t = 0.75$

Figure 6: Comparison results between the reference solution and the predicted solutions by IFNN and PINN for the 1D inviscid Burgers' equation (14) – Shock wave (I).

Table 2: Relative L_2 errors of the predicted solutions by IFNN and PINN with different activation functions for solving the 1D inviscid Burgers' equation (14) – Shock wave (I).

Model \ Activation	tanh	sin	ReLU
PINN	1.17×10^{-1}	1.54×10^{-1}	1.05×10^{0}
IFNN	2.53×10^{-4}	6.63×10^{-2}	8.01×10^{-2}

Shock wave (II) Next we choose $\Omega = [-1, 3/2]$, the initial condition $\phi(x)$

$$\phi(x) = \begin{cases} 2, & x \leqslant -1/2, \\ 1, & -1/2 < x \leqslant 1/2, \\ 0, & x > 1/2, \end{cases} \tag{19}$$

and the Neumann boundary condition $u'(-1, t) = u'(3/2, t) = 0$ for $t \in (0, 3/2]$. The exact solution u is constant along the projected characteristic curves

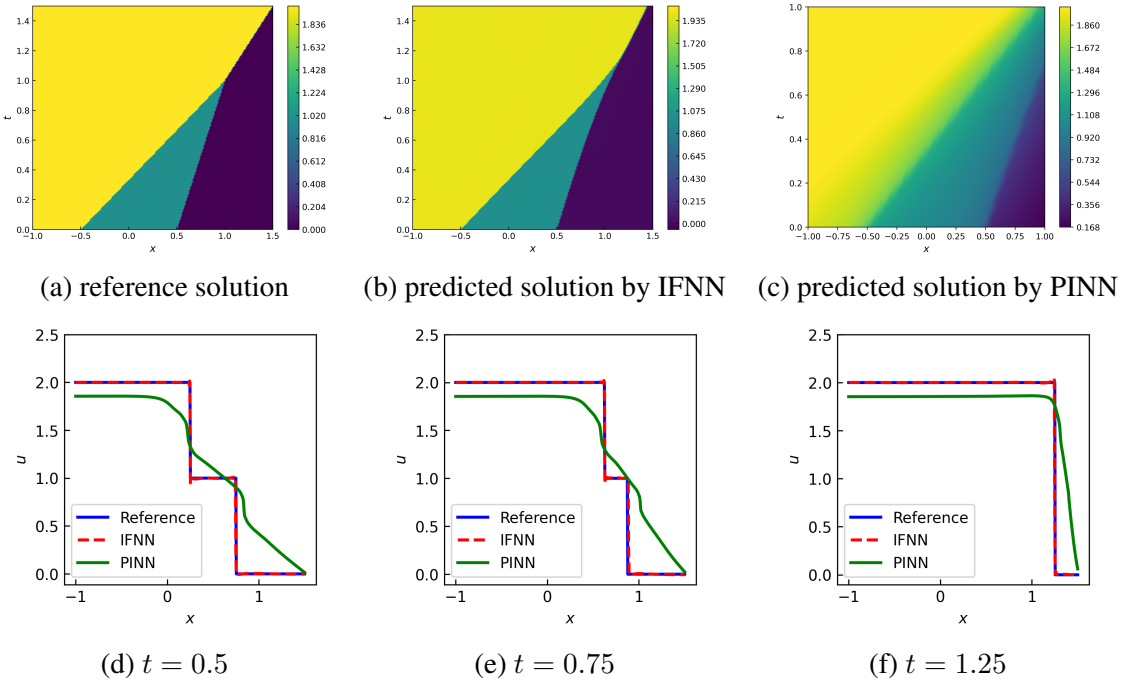
$$x = \phi(r)t + r = \begin{cases} 2t + r, & r < -1/2, \\ t + r, & -1/2 < r < 1/2, \\ r, & r > 1/2, \end{cases}$$

which is shown in Fig. 3-(b). By using Rankine-Hugoniot jump condition, there exist two curves of discontinuity $x = (3t - 1)/2$ and $x = (t + 1)/2$ when $t < 1$, and after which these two curves will

merge into one $x = t$. Therefore, the weak solution of (14) in this case can be formulated as:

$$u(x,t) = \begin{cases} 2, & t < 1,\ x < (3t-1)/2, \\ 1, & t < 1,\ (3t-1)/2 < x < (t+1)/2, \\ 0, & t < 1,\ x > (t+1)/2, \\ 2, & t > 1,\ x < t, \\ 0, & t > 1,\ x > t. \end{cases} \tag{20}$$

The comparison results shown in Fig. 7 still exhibit the superiority of IFNN over PINN in dealing with the solutions with multiple discontinuities. Again the predicted solution by IFNN is essentially in agreement with the reference solution in the whole spatial-temporal domain while the PINN does not capture the shock well. The relative L_2 errors of the predicted solutions by IFNN and PINN using different activation functions are reported in Table 3. Again $\tanh$ is still preferred for both IFNN and PINN among the three activation functions.



(a) reference solution (b) predicted solution by IFNN (c) predicted solution by PINN

(d) $t = 0.5$ (e) $t = 0.75$ (f) $t = 1.25$

Figure 7: Comparison results between the reference solution and the predicted solutions by IFNN and PINN for the 1D inviscid Burgers' equation (14) – Shock wave (II).

Rarefaction wave In this case we take $\Omega = [-1, 1]$, the initial condition

$$\phi(x) = \begin{cases} 0, & x \leqslant 0, \\ 1, & x > 0. \end{cases} \tag{21}$$

Table 3: Relative L_2 errors of the predicted solutions by IFNN and PINN with different activation functions for solving the 1D inviscid Burgers' equation (14) – Shock wave (II).

Model \ Activation	tanh	sin	ReLU
PINN	0.73	1.04	1.12
IFNN	5.41×10^{-3}	7.69×10^{-3}	2.43×10^{-2}

and the boundary condition $u(-1,t) = 2$ for $t \in (0, 5/2]$. The corresponding projected characteristic curve is

$$x = \begin{cases} t, & r < 0, \\ t + r, & r > 0, \end{cases} \tag{22}$$

It is shown in Fig.3-(c), from which we find there is a region where the information is not enough to define the solution. If no additional conditions is provided, the equation (14) may have multiple solutions. To determine a physical solution, the entropy condition

$$f'(u^-) > \xi'(t) > f'(u^+),$$

must be satisfied, which leads to

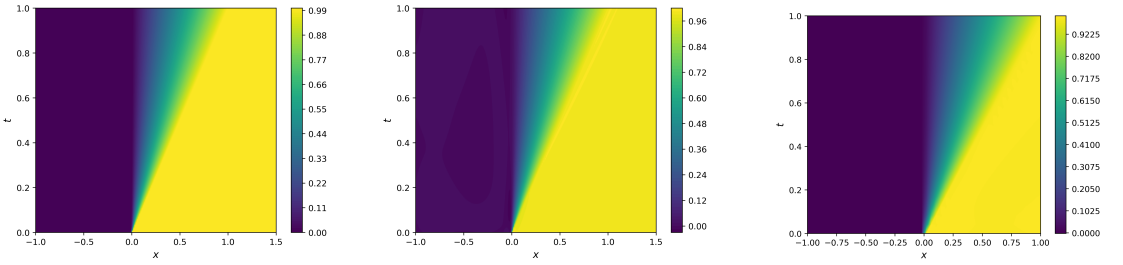
$$u(x,t) = \begin{cases} 0, & \text{for } x < 0, \\ x/t, & \text{for } 0 \leqslant x \leqslant t, \\ 1, & \text{for } x > t. \end{cases} \tag{23}$$

Such type of solution is called a rarefaction wave. The comparison results of IFNN and PINN are presented in Fig. 8 and Table 4. It is clear that both IFNN and PINN can produce the accurate results meeting the entropy conditions automatically without any additional constraints, implying that both of them can capture the continuous solutions very well.

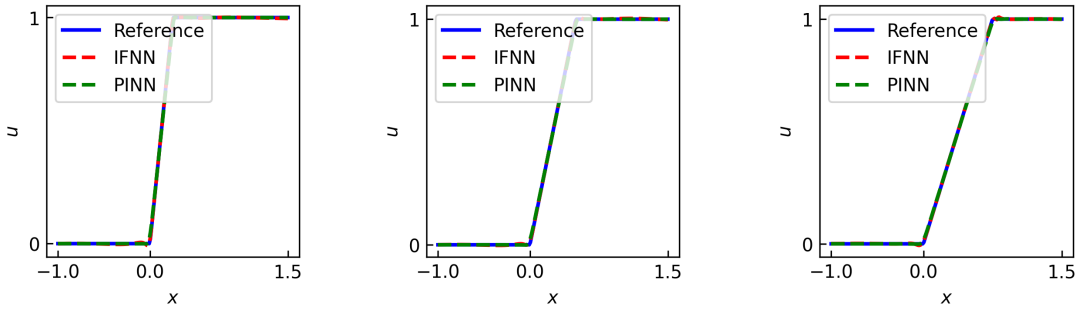
Table 4: Relative L_2 errors of the predicted solutions by IFNN and PINN with different activation functions for solving the 1D inviscid Burgers' equation (14) -Rarefaction wave.

Model \ Activation	tanh	sin	ReLU
PINN	2.89×10^{-3}	7.56×10^{-3}	9.10×10^{-3}
IFNN	4.13×10^{-3}	6.65×10^{-3}	7.32×10^{-3}

With sinusoidal initial data Now we consider the 1D inviscid Burgers' equation (14) with $\Omega = [0, 1]$, the smooth initial condition $\phi(x) = \sin(2\pi x)$ and the periodic boundary condition $u(0,t) = u(1,t)$ for $t \in (0, 1]$. In this case, the exact solution can not be expressed explicitly, and we use a traditional finite volume method ("Godunov" scheme) with fine grid to obtain the reference solution on a fined-enough mesh. The comparison results of IFNN and PINN are presented in Fig. 9. We observe that the IFNN produces the satisfactory predictions which are in excellent agreement with the finite volume solution, while the PINN fails to capture the shock completely. The relative L_2 errors of the predicted solutions by IFNN and PINN using different activation functions are reported in Table 5. In this case, ReLU is preferred for IFNN among the three activation functions.



(a) reference solution (b) predicted solution by IFNN (c) predicted solution by PINN



(d) $t = 0.25$ (e) $t = 0.5$ (f) $t = 0.75$

Figure 8: Comparison results between the reference solution and the predicted solutions by IFNN and PINN for the 1D inviscid Burgers' equation (14) – Rarefaction wave.

Table 5: Relative L_2 errors of the predicted solutions by IFNN and PINN with different activation functions for solving the 1D inviscid Burgers' equation (14) with sinusoidal initial data.

Model \ Activation	tanh	sin	ReLU
PINN	0.42	0.74	1.23
IFNN	3.72×10^{-3}	3.54×10^{-3}	1.02×10^{-3}

3.2. 2D inviscid Burgers' equation

By choosing $d = 2$ and $\boldsymbol{f} = (u^2/2, u^2/2)^T$ in (1), we obtain the 2D inviscid Burgers' equation

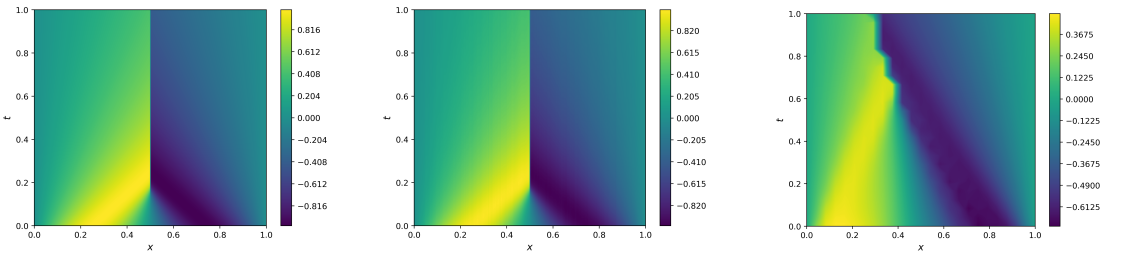
$$\begin{cases} u_t + u(u_{x_1} + u_{x_2}) = 0, & \boldsymbol{x} \in \mathbb{R}^2, t > 0, \\ u(\boldsymbol{x}, 0) = \phi(\boldsymbol{x}), & \boldsymbol{x} \in \mathbb{R}^2. \end{cases} \tag{24}$$

We choose $\Omega = [0, 1] \times [0, 1]$, the initial condition

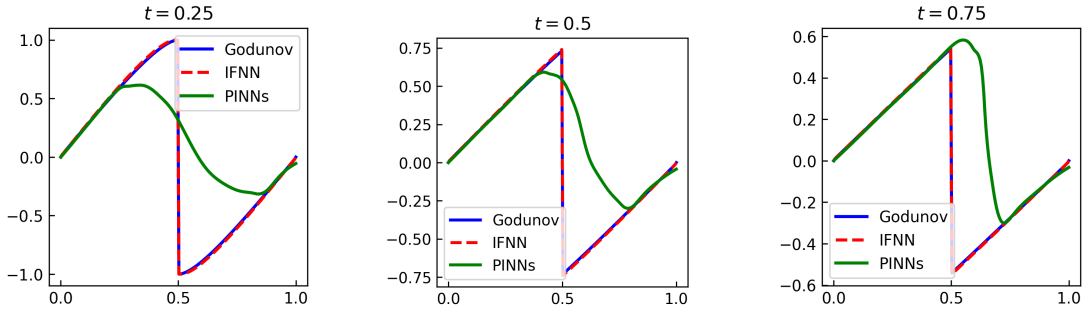
$$\phi(\boldsymbol{x}) = \begin{cases} 1, & \boldsymbol{x} \in [0, 1/2]^2, \\ 0, & \text{otherwise}, \end{cases} \tag{25}$$

and the boundary condition

$$u(0, x_2, t) = \begin{cases} 1, & x_2 < (t+1)/2, \ t \in [0, 1], \\ 0, & \text{otherwise}, \end{cases} \tag{26}$$



(a) reference solution (b) predicted solution by IFNN (c) predicted solution by PINN



(d) $t = 0.25$ (e) $t = 0.5$ (f) $t = 0.75$

Figure 9: Comparison results between the reference solution and the predicted solutions by IFNN and PINN for the 1D inviscid Burgers' equation (14) with sinusoidal initial data.

$$u(x_1, 0, t) = \begin{cases} 1, & x_1 < (t+1)/2,\ t \in [0,1], \\ 0, & \text{otherwise}, \end{cases} \tag{27}$$

Then the exact solution is given by

$$u(\boldsymbol{x}, t) = \begin{cases} 1, & \boldsymbol{x} \in [0, (t+1)/2]^2,\ t \in (0,1], \\ 0, & \text{otherwise}. \end{cases}$$

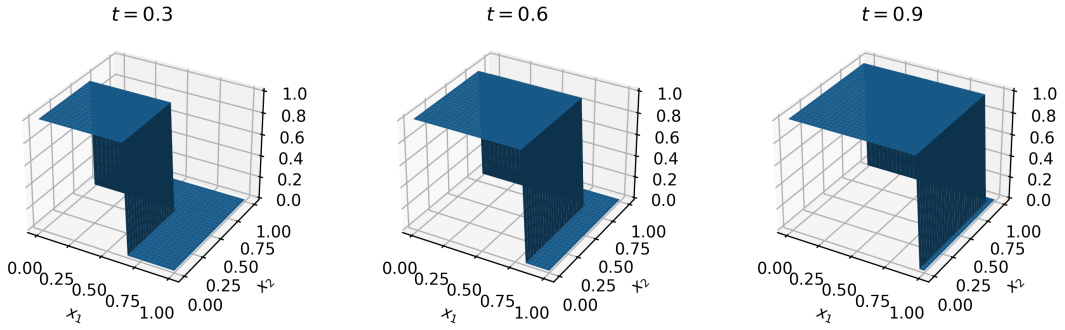
The reference solution and the predicted solution at different times by IFNN are shown and compared in Fig. 10, from which we see that IFNN still predict the solution and capture the 2D shock waves very well.

3.3. LWR model

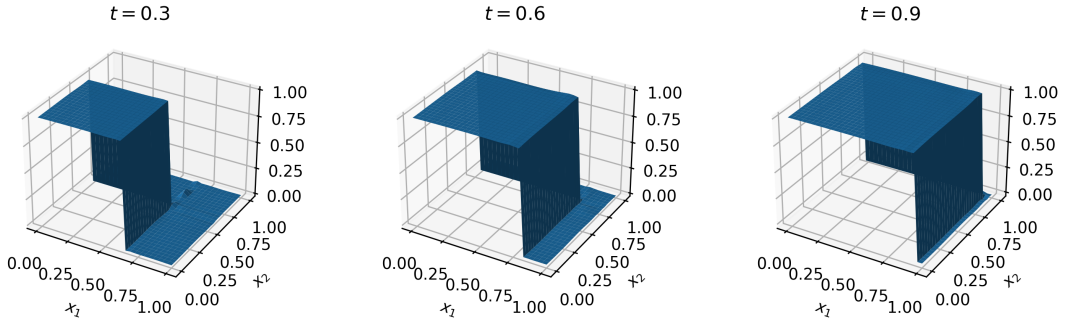
In addition to inviscid Burgers' equation, we also investigate the scalar conservation laws for modeling the traffic flow problem, i.e., the Lighthill-Whitham-Richards (LWR) (Lighthill and Whitham, 1955; Richards, 1956) model. The LWR model can be obtained from (1) by choosing $d = 1$ and $f(u) = u(1-u)$, and equivalently we obtain

$$\begin{cases} u_t + [u(1-u)]_x = 0, & x \in \mathbb{R}, t > 0, \\ u(x,0) = \phi(x), & x \in \mathbb{R}, \end{cases} \tag{28}$$

where u represents the density of cars on a road and $u \in [0,1]$. When $u = 0$, there are no car on the road, and the road is completely full when $u = 1$.



(a) Reference solution



(b) Predicted solution by IFNN

Figure 10: Comparison results between the reference solution and the predicted solution by IFNN for the 2D inviscid Burgers' equation (24).

Traffic jams In this case, we choose $\Omega = [-1/2, 1/2]$, the initial condition

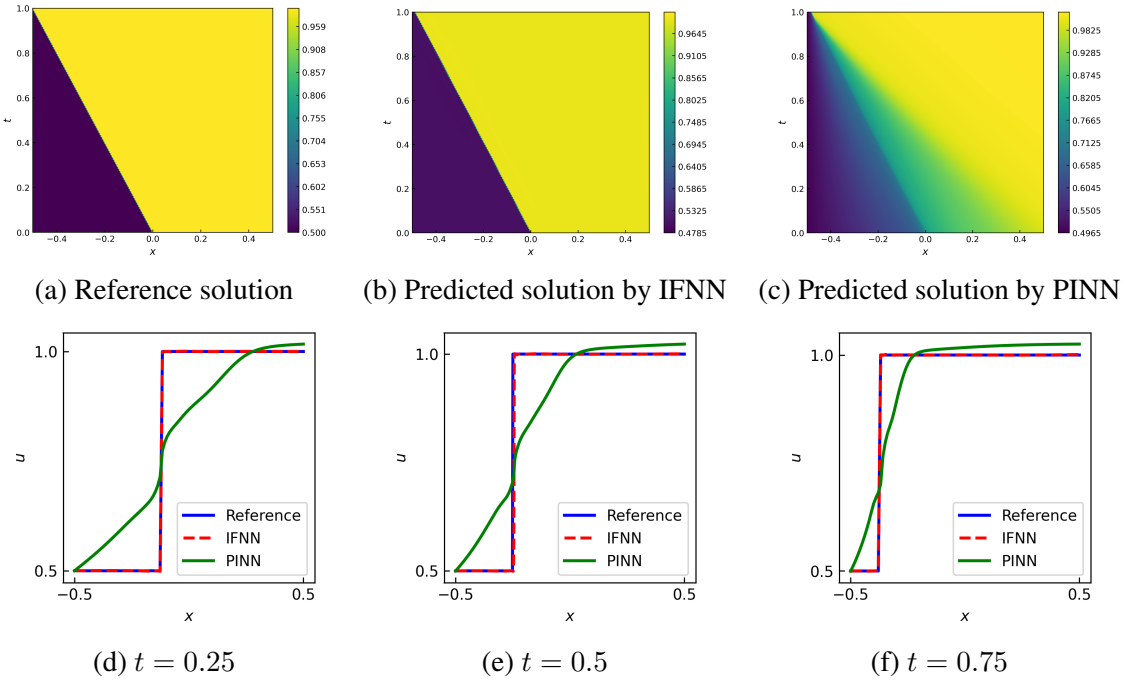
$$\phi(x) = \begin{cases} 1/2, & x < 0, \\ 1, & x > 0, \end{cases} \tag{29}$$

and the Dirichlet boundary condition $u(-1/2, 0, t) = 1/2$ for $t \in (0, 1]$. For this case, the equation (28) describes the traffic state during traffic jams (Ketcheson et al., 2020) and a traveling shock appears. The comparison results shown in Fig. 11 and Table 6 exhibit the superiority of IFNN over PINN in agreement with the reference solution in the whole spatial-temporal domain and in capturing the traveling shock. In addition, $\tanh$ significantly outperforms $\sin$ and ReLU as the activation function for both IFNN and PINN.

Traffic light turning green Next we consider the LWR model (28) describing the traffic state when the traffic light turns green, and in this case, we choose $\Omega = [-1, 1]$, and the initial condition

$$\phi(x) = \begin{cases} 1, & x < 0, \\ 0, & x > 0, \end{cases} \tag{30}$$

and the Dirichlet boundary condition $u(-1, t) = 1$ for $t \in (0, 1]$. In the case of traffic light turning green the solution is continuous like the rarefaction wave. The comparison results of IFNN and



(a) Reference solution (b) Predicted solution by IFNN (c) Predicted solution by PINN

(d) $t = 0.25$ (e) $t = 0.5$ (f) $t = 0.75$

Figure 11: Comparison results between the reference solution and the predicted solutions by IFNN and PINN for the LWR model (28) – the case of traffic jams.

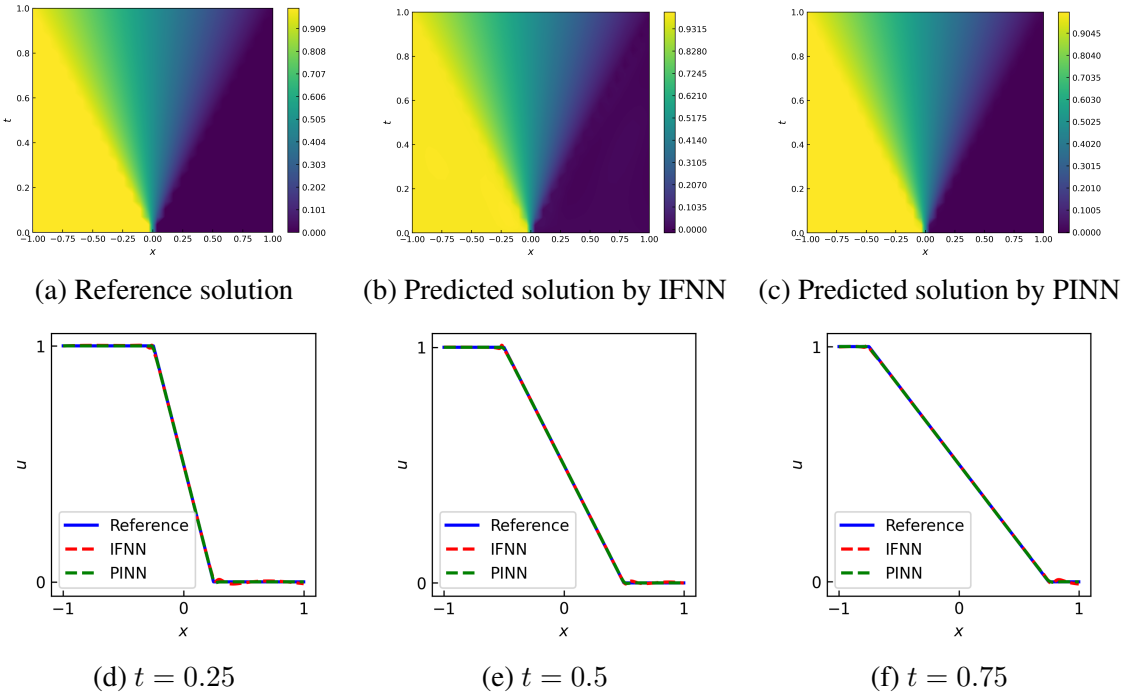
Table 6: Relative L_2 errors of the predicted solutions by IFNN and PINN with different activation functions for solving the LWR model (28) – the case of traffic jams.

model \ activation	tanh	sin	ReLU
PINN	1.25×10^{-3}	2.73×10^{-1}	4.18×10^{-1}
IFNN	6.12×10^{-3}	1.32×10^{-2}	5.17×10^{-2}

PINN are presented in Fig. 12 and Table 7. It is clear that both IFNN and PINN can approximate the continuous solution very well, just like the case of the rarefaction wave. Among the three activation functions, tanh still performs the best but not significantly for both IFNN and PINN.

Table 7: Relative L_2 errors of the predicted solutions by IFNN and PINN with different activation functions for solving the LWR model (28) – the case of traffic light turning green.

model \ activation	tanh	sin	ReLU
PINN	1.31×10^{-3}	3.26×10^{-3}	8.10×10^{-3}
IFNN	5.80×10^{-3}	7.47×10^{-3}	6.71×10^{-3}



(a) Reference solution (b) Predicted solution by IFNN (c) Predicted solution by PINN

(d) $t = 0.25$ (e) $t = 0.5$ (f) $t = 0.75$

Figure 12: Comparison results between the reference solution and the predicted solutions by IFNN and PINN for the LWR model (28) – the case of traffic light turning green.

4. Conclusion

In this paper, we have proposed a fully-connected neural network with skip connection, "IFNN", for solving the scalar conservation laws. The essential difference between our IFNN and PINN lies the choice of the loss function. IFNN takes the implicit form of the solution to formulate one of the essential terms for the loss function while PINN directly uses the residual of the original PDE. Extensive numerical experiments in 1D and 2D show that our IFNN is superior to PINN in capturing shock waves while their performance are comparable for the continuous solution cases. In addition, the training of IFNN is also much easier than that of PINN since it need not to use automatic differentiation for calculations of differential operators. On the other hand, IFNN requires the target PDEs to have the specific implicit form for their solutions, thus the scope of its feasibility may not be as wide as PINN. It would be interesting in our future research to explore the use of IFNN in more application fields such as PDE-based uncertainty quantification and inverse problems.

Acknowledgments

X. Zhang's research has been partially supported by National Natural Science Foundation of China under grant numbers 11771364 and 11671313. L. Ju's research is partially supported by US National Science Foundation grant DMS-1818438.

References

Jens Berg and Kaj Nystrom. A unified deep artificial neural network approach to partial differential equations in complex geometries. *Neurocomputing*, 317:28–41, 2018.

Jingrun Chen, Rui Du, and Keke Wu. A comparison study of deep galerkin method and deep ritz method for elliptic problems with different boundary conditions. *Communications in Mathematical Research*, 36(3):354–376, 2020.

Jacob Devlin, Mingwei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Conference of the North American Chapter of the ACL: Human Language Technologies*, pages 4171–4186, 2019.

Weinan E and Bing Yu. The deep ritz method: A deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1):1–12, 2018.

Jiequn Han, Arnulf Jentzen, and Weinan E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115:8505–8510, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of International Conference on Machine Learning*, pages 448–456, 2015.

Ameya Jagtap, Ehsan Kharazmi, and George Em Karniadakis. Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 365:113028, 06 2020.

David I. Ketcheson, Randall J. LeVeque, and Mauricio J. del Razo. *Riemann Problems and Jupyter Solutions*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2020.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 25, 2012.

Hyuk Lee and In Seok Kang. Neural algorithm for solving differential equations. *Journal of Computational Physics*, 91(1):110–131, 1990.

Yulei Liao and Pingbing Ming. Deep nitsche method: Deep ritz method with essential boundary conditions. *arXiv:1912.01309*, 2019.

Michael J Lighthill and Gerald Beresford Whitham. On kinematic waves. ii. a theory of traffic flow on long crowded roads. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 229(1178):317–345, 1955.

Yang Liu, Dongkun Zhang, and George Em Karniadakis. Physics-informed generative adversarial networks for stochastic differential equations. *SIAM Journal of Scientific Computing*, 42(1): A292–A317, 2020.

Yang Liu, Xuhui Meng, and George Em Karniadakis. B-pinns: Bayesian physics-informed neural networks for forward and inverse pde problems with noisy data. *Journal of Computational Physics*, 425:109913, 2021.

Xuhui Meng, Zhen Li, Dongkun Zhang, and George Em Karniadakis. Ppinn: Parareal physics-informed neural network for time-dependent pdes. *Computer Methods in Applied Mechanics and Engineering*, 370(1):113250, 2020.

Johannes Müller and Marius Zeinhofer. Deep ritz revisited. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020.

Guofei Pang, Lu Lu, and George Em Karniadakis. Fpinns: Fractional physics-informed neural networks. *SIAM Journal on Scientific Computing*, 41(4):A2603–A2626, 2019.

Maziar Raissi and George Em Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357:125–141, 2018.

Maziar Raissi, Alireza Yazdani, and George Em Karniadakis. Hidden fluid mechanics: A navier-stokes informed deep learning framework for assimilating flow visualization data, 2018.

Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

Paul I Richards. Shock waves on the highway. *Operations research*, 4(1):42–51, 1956.

Justin A Sirignano and Konstantinos Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, 2018.

Luning Sun, Han Gao, Shaowu Pan, and Jianxun Wang. Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Computer Methods in Applied Mechanics and Engineering*, 361:112732, 2020.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, 2014.

Yufei Wang, Ziju Shen, Zichao Long, and Bin Dong. Learning to discretize: Solving 1d scalar conservation laws via deep reinforcement learning. *Communication in Computational Physics*, 28(5):2158–2179, 2020.

Zhongjian Wang and Zhiwen Zhang. A mesh-free method for interface problems using the deep learning approach. *Journal of Computational Physics*, 400:108963, 2020.

Yibo Yang and Perdikaris Paris. Adversarial uncertainty quantification in physics-informed neural networks. *Journal of Computational Physics*, 394:136–152, 2019.

Dongkun Zhang, Ling Guo, and George Em Karniadakis. Learning in modal space: Solving time-dependent stochastic pdes using physics-informed neural networks. *SIAM Journal on Scientific Computing*, 42(2):A639–A665, 2020.