LEAST-SQUARES RELU NEURAL NETWORK (LSNN) METHOD FOR SCALAR NONLINEAR HYPERBOLIC CONSERVATION LAW[*]

ZHIQIANG CAI[†], JINGSHUANG CHEN[†], AND MIN LIU[‡]

Abstract. In [6], we introduced the least-squares ReLU neural network (LSNN) method for solving the linear advection-reaction problem with discontinuous solution and showed that the method outperforms mesh-based numerical methods in terms of the number of degrees of freedom. This paper studies the LSNN method for scalar nonlinear hyperbolic conservation law. The method is a discretization of an equivalent least-squares (LS) formulation in the set of neural network functions with the ReLU activation function. Evaluation of the LS functional is done by using numerical integration and conservative finite volume scheme. Numerical results of some test problems show that the method is capable of approximating the discontinuous interface of the underlying problem automatically through the *free* breaking lines of the ReLU neural network. Moreover, the method does not exhibit the common Gibbs phenomena along the discontinuous interface.

Key words. Least-Squares Method, ReLU Neural Network, Scalar Nonlinear Hyperbolic Conservation Law

AMS subject classifications.

1. Introduction.

Let Ω be a bounded domain in $\mathbb{R}^d$ ($d = 1$, 2, or 3) with Lipschitz boundary, consider the scalar nonlinear hyperbolic conservation law

$$(1.1)\qquad \left\{\begin{array}{rcll} u_t(\mathbf{x},t) + \nabla_{\mathbf{x}} \cdot \mathbf{f}(u) & = & 0, & \text{in } \ \Omega \times I, \\ u & = & g, & \text{on } \ \Gamma_-, \\ u(\mathbf{x},0) & = & u_0(\mathbf{x}), & \text{in } \ \Omega, \end{array}\right.$$

where u_t is the partial derivative of u with respect to temporal variable t; $\nabla_{\mathbf{x}}\cdot$ is a divergence operator with respect to spatial variable $\mathbf{x}$; $\mathbf{f}(u) = (f_1(u), ..., f_d(u))$ is the spatial flux vector field; $I = (0,T)$ is temporal interval; Γ_- is the part of the boundary $\partial\Omega \times I$ where the characteristic curves enter the domain $\Omega \times I$; and the boundary data g and the initial data u_0 are given scalar-valued functions.

Numerical methods for (1.1) have been intensively studied during the past several decades by many researchers and many numerical schemes have been developed. A major difficulty in simulation is that the solution of a scalar hyperbolic conservation law is often discontinuous due to the discontinuous initial/boundary condition or shock formation; moreover, there is no *a priori* knowledge of the location of the discontinuities. It is well-known that traditional mesh-based numerical methods often exhibit oscillations near a discontinuity (called the Gibbs phenomena). Such spurious oscillations are unacceptable for many applications (see, e.g, [18]). To eliminate or reduce the Gibbs phenomena, finite difference and finite volume methods often use numerical techniques such as limiters and filters and conservative schemes such as Roe, ENO/WENO, etc. have been developed [14, 17, 18, 23]; and finite element methods usually employ discontinuous finite elements [3, 8, 11] and/or adaptive mesh refinement (AMR) to generate locally refined elements along discontinuous interfaces (see, e.g., [4, 19, 20]).

Recently, there has been increasing interests in using neural networks (NNs) to solve partial differential equations (see, e.g., [1, 7, 12, 26, 30]). NNs produce a large class of functions through compositions of linear transformations and activation functions. One of the striking features of NNs is that this class of functions is not subject to a hand-crafted geometric mesh or point cloud

[*]This work was supported in part by the National Science Foundation under grant DMS-2110571.

[†]Department of Mathematics, Purdue University, 150 N. University Street, West Lafayette, IN 47907-2067 (caiz@purdue.edu, chen2042@purdue.edu).

[‡]School of Mechanical Engineering, Purdue University, 585 Purdue Mall, West Lafayette, IN 47907-2088(liu66@purdue.edu).

as are the traditional, well-studied finite difference, finite volume, and finite element methods. The physical partition of the domain Ω, formed by free hyper-planes, can automatically adapt to the target function. To make use this powerful approximation property of NNs, in [6], we studied least-squares neural network (LSNN) method for solving linear advection-reaction problem with discontinuous solution. The LSNN method is based on a direct application of the lease-squares principle to the underlying problem studied in ([2, 9]) and on the ReLU neural network as the approximation class of functions. Compared to various AMR methods that locate the discontinuous interface through local mesh refinement, the LSNN method is much more effective in terms of the number of the degrees of freedom.

The purpose of this paper is to study the space-time LSNN method for solving the scalar hyperbolic conservation law. For the nonlinear hyperbolic conservation law, the differential equation is not generally sufficient to determine the solution. An additional constraint the so-called Rankine-Hugoniot (RH) jump condition [13, 23, 31], is needed at where the solution is not continuous. To enforce this condition weakly, [10] introduced an independent variable, the spatial-temporal flux, for the inviscid Burgers equation and applied the least-squares principle to the resulting equivalent system. A variant of this method was also studied in [10, 21] by using the Helmholtz decomposition of the flux.

Due to the training difficulty of the least-squares method of [10], in this paper we employ the naive least-squares method used for the linear advection-reaction problem [6], i.e., a direct application of least-squares principle to the PDE, initial and inflow boundary conditions. To ensure that the numerical solution enforces the RH jump condition, we introduce implicit discrete finite difference operators in section 3 by following ideas of the explicit conservative schemes such as Roe, ENO, etc.

Based on our numerical experience, it is difficult to train the network for problems with sharp changes when the LSNN method is applied to the entire computational domain Ω. This is due to the nature of nonlinear hyperbolic conservation laws since information is transported from initial and inflow boundary to the rest of the domain along the flow direction. To overcome this training difficulty, we propose the block space-time LSNN method in section 4. Basically, we partition the computational domain into a number of blocks based on the "inflow" boundary and initial conditions, then the method solves problems on these blocks sequentially. The trained parameters of the NN model for the previous block is used as an initial for the current block.

NN-based numerical methods for solving scalar nonlinear hyperbolic conservation laws have been studied recently by many researchers (see, e.g., [1, 25]). The most popular one is the so-called the physics informed neural network (PINN) method proposed in [25]. Basically, the PINN is a archaic version of the LSNN method; it employs a primitive form of least-squares formulation and uses the automatic differentiation method to differentiate the neural network. This is why the PINN and other NN-based methods are only applicable to problems with smooth solution such as the viscid Burgers equation but not the inviscid one studied in this paper.

The paper is organized as follows. The ReLU neural network and the space-time LSNN method are introduced in section 2. Conservative finite difference operators are discussed in section 3. The block LSNN method is explained in section 4. Finally, implementation and numerical results for various one dimensional benchmark test problems are presented in section 5.

2. Space-Time Least-Squares Neural Network Method.

In this section, we describe least-squares neural network method for the scalar hyperbolic conservation law.

A deep neural network (DNN) defines a scalar-valued function

$$\mathcal{N}:\, \mathbf{z} = (\mathbf{x}, t) \in \mathbb{R}^{d+1} \longrightarrow \mathcal{N}(\mathbf{z}) \in \mathbb{R}.$$

A DNN function $\mathcal{N}(\mathbf{z})$ is typically represented as compositions of many layers of functions:

$$\mathcal{N}(\mathbf{z}) = \boldsymbol{\omega}^{(L)} \left(\mathcal{N}^{(L-1)} \circ \cdots \mathcal{N}^{(2)} \circ \mathcal{N}^{(1)}(\mathbf{z}) \right) - b^{(L)}, \tag{2.1}$$

where $\boldsymbol{\omega}^{(L)} \in \mathbb{R}^{n_{L-1}}$, $b^{(L)} \in \mathbb{R}$, the symbol $\circ$ denotes the composition of functions, and L is the depth of the network. For $l = 1, \cdots, L-1$, the $\mathcal{N}^{(l)} : \mathbb{R}^{n_{l-1}} \to \mathbb{R}^{n_l}$ is called the l^{th} hidden layer of the network defined as follows:

$$\mathcal{N}^{(l)}(\mathbf{z}^{(l-1)}) = \sigma(\boldsymbol{\omega}^{(l)}\mathbf{z}^{(l-1)} - \mathbf{b}^{(l)}) \quad \text{for } \mathbf{z}^{(l-1)} \in \mathbb{R}^{n_{l-1}}, \tag{2.2}$$

where $\boldsymbol{\omega}^{(l)} \in \mathbb{R}^{n_l \times n_{l-1}}$, $\mathbf{b}^{(l)} \in \mathbb{R}^{n_l}$, $\mathbf{z}^{(0)} = \mathbf{z}$, and σ is the activation function and its application to a vector is defined component-wisely. This paper will use the popular rectified linear unit (ReLU) activation function defined by

$$\sigma(s) = \max\{0,\, s\} = \begin{cases} 0, & \text{if } s \le 0, \\ s, & \text{if } s > 0. \end{cases} \tag{2.3}$$

Denote the set of NN functions by

$$\mathcal{M}(\boldsymbol{\theta}, L) = \left\{\mathcal{N}(\mathbf{z}) = \boldsymbol{\omega}^{(L)}\left(\mathcal{N}^{(L-1)} \circ \cdots \mathcal{N}^{(2)} \circ \mathcal{N}^{(1)}(\mathbf{z})\right) - b^{(L)} : \boldsymbol{\omega}^{(l)} \in \mathbb{R}^{n_l \times n_{l-1}},\ \mathbf{b}^{(l)} \in \mathbb{R}^{n_l}\right\},$$

where $\mathcal{N}^{(l)}(\mathbf{z}^{(l-1)})$ is defined in (2.2) and $\boldsymbol{\theta}$ denotes all parameters: $\boldsymbol{\omega}^{(l)}$ and $\mathbf{b}^{(l)}$ for $l = 1, ..., L$. It is easy to see that $\mathcal{M}(\boldsymbol{\theta}, L)$ is a set, but not a linear space.

Applying the least-squares principle directly to the problem in (1.1), we have the following least-squares (LS) functional

$$\mathcal{L}(v; g) = \|v_t + \nabla_{\mathbf{x}} \cdot \mathbf{f}(v)\|^2_{0,\Omega\times I} + \|v - g\|^2_{0,\Gamma_-} + \|v(\mathbf{x}, 0) - u_0(\mathbf{x})\|^2_{0,\Omega}. \tag{2.4}$$

Then the least-squares approximation is to find $u_{_N}(\mathbf{x}, t; \boldsymbol{\theta}^*) \in \mathcal{M}(\boldsymbol{\theta}, L)$ such that

$$\mathcal{L}\big(u_{_N}(\cdot; \boldsymbol{\theta}^*);\, \mathbf{f}\big) = \min_{v \in \mathcal{M}(\boldsymbol{\theta}, L)} \mathcal{L}\big(v(\cdot; \boldsymbol{\theta});\, g\big) = \min_{\boldsymbol{\theta} \in \mathbb{R}^N} \mathcal{L}\big(v(\cdot; \boldsymbol{\theta});\, g\big), \tag{2.5}$$

where N is the total number of parameters in $\mathcal{M}(\boldsymbol{\theta}, L)$ given by

$$N = M_d(L) = \sum_{l=1}^{L} n_l \times (n_{l-1} + 1).$$

Similar to [6, 7], the integral in the LS functional is evaluated by numerical integration. To do so, let

$$\mathcal{T} = \{K : K \text{ is an open subdomain of } \Omega \times I\}$$

be a partition of the domain Ω. Then

$$\mathcal{E}_- = \{E = \partial K \cap \Gamma_- :\ K \in \mathcal{T}\} \quad \text{and} \quad \mathcal{E}_0 = \{E = \partial K \cap (\Omega \times \{0\}) :\ K \in \mathcal{T}\}$$

are partitions of the boundary Γ_- and $\Omega \times \{0\}$, respectively. Let $\mathbf{z}_{_K} = (\mathbf{x}_{_K}, t_{_K})$ and $\mathbf{z}_{_E} = (\mathbf{x}_{_E}, t_{_E})$ be the centroids of $K \in \mathcal{T}$ and E in $\mathcal{E}_-$ or $\mathcal{E}_0$, respectively. Define the discrete LS functional as follows:

$$\mathcal{L}_{\mathcal{T}}\big(v(\cdot; \boldsymbol{\theta}); g\big) = \sum_{K \in \mathcal{T}} \big(\delta_\tau v + \nabla_{\mathbf{h}} \cdot \mathbf{f}(v)\big)^2(\mathbf{z}_{_K}; \boldsymbol{\theta})\,|K| + \sum_{E \in \mathcal{E}_-} \big(v - g\big)^2(\mathbf{z}_{_E}; \boldsymbol{\theta})|E| + \sum_{E \in \mathcal{E}_0} \big(v - u_0\big)^2(\mathbf{z}_{_E}; \boldsymbol{\theta})|E|, \tag{2.6}$$

where $|K|$ and $|E|$ are the d and $d-1$ dimensional measures of K and E, respectively; δ_τ and $\nabla_{\mathbf{h}} \cdot$ are finite difference operators to be defined in the subsequent section. Then the discrete least-squares approximation is to find $u_{_\mathcal{T}}(\mathbf{z}, \boldsymbol{\theta}^*) \in \mathcal{M}(\boldsymbol{\theta}, L)$ such that

$$\mathcal{L}_{\mathcal{T}}\big(u_{_\mathcal{T}}(\cdot, \boldsymbol{\theta}^*); g\big) = \min_{v \in \mathcal{M}(\boldsymbol{\theta}, L)} \mathcal{L}_{\mathcal{T}}\big(v(\cdot; \boldsymbol{\theta});\, g\big) = \min_{\boldsymbol{\theta} \in \mathbb{R}^N} \mathcal{L}_{\mathcal{T}}\big(v(\cdot; \boldsymbol{\theta}); g\big). \tag{2.7}$$

3. Conservative Finite Volume Operator. How to discretize the differential operator is critical for the success of the LSNN method. Using finite difference quotient along coordinate directions to approximate the differential operator usually results in very poor numerical approximation. To overcome this difficulty, we employ conservative finite volume schemes to evaluate the derivatives in the least-squares functional in (2.6). There are many conservative schemes such as Roe's scheme, ENO, and WENO, etc. (see, e.g., [27, 28, 29]). For simplicity, we briefly describe the finite volume operator using the idea of either Roe's scheme or second-order accurate ENO in this section. Note that the finite volume operators described in this section is implicit in time.

For any $K \in \mathcal{T}$, let $(\mathbf{x}_K, t_K)$ be the centroid of K. Let $(\mathbf{h}, \tau) = (h_1, ..., h_d, \tau)$ be a step size such that $(\mathbf{x}_K \pm \mathbf{h}/2, t_K \pm \tau/2) \in K$. For $i = 1, ..., d$, let $\mathbf{h}_i = h_i \mathbf{e}_i$, where $\mathbf{e}_i$ is the unit vector in the x_i-coordinate direction. Then the finite volume operator of the least-squares functional in (2.6) at the point $(\mathbf{x}_K, t_K)$ is given by

$$\begin{aligned}&\big(\delta_\tau v + \nabla_{\mathbf{h}} \cdot \mathbf{f}(v)\big)(\mathbf{x}_K, t_K)\\ &= \frac{v(\mathbf{x}_K, t_K) - v(\mathbf{x}_K, t_K - \frac{1}{2}\tau)}{\tau} + \sum_{i=1}^{d} \frac{\hat{f}_i(v(\mathbf{x}_K + \frac{1}{2}\mathbf{h}_i, t_K)) - \hat{f}_i(v(\mathbf{x}_K - \frac{1}{2}\mathbf{h}_i, t_K))}{h_i},\end{aligned} \tag{3.1}$$

where $\hat{f}_i(v(\mathbf{x}_K \pm \frac{1}{2}\mathbf{h}_i, t_K))$ are the i^{th} component of the numerical flux at $(\mathbf{x}_K \pm \frac{1}{2}\mathbf{h}_i, t_K)$. Various conservative schemes are more or less on how to reconstruct proper numerical flux.

Below, we describe the numerical fluxes by Roe and ENO. To this end, we introduce the Roe speed at point $(\mathbf{x}_K \pm \frac{1}{2}\mathbf{h}_i, t_K)$ in the $\mathbf{e}_i$ direction

$$a_i\left(\mathbf{x}_K \pm \frac{1}{2}\mathbf{h}_i, t_K\right) = \begin{cases} \dfrac{f_i(v(\mathbf{x}_K \pm \mathbf{h}_i, t_K)) - f_i(v(\mathbf{x}_K, t_K))}{v(\mathbf{x}_K \pm \mathbf{h}_i, t_K) - v(\mathbf{x}_K, t_K)}, & \text{if } v(\mathbf{x}_K \pm \mathbf{h}_i, t_K) \neq v(\mathbf{x}_K, t_K), \\ f_i'(v(\mathbf{x}_K, t_K)), & \text{if } v(\mathbf{x}_K \pm \mathbf{h}_i, t_K) = v(\mathbf{x}_K, t_K). \end{cases} \tag{3.2}$$

Then the i^{th} components of the Roe numerical flux at $(\mathbf{x}_K \pm \frac{1}{2}\mathbf{h}_i, t_K)$ are given by

$$\begin{aligned}&\hat{f}_i\left(v(\mathbf{x}_K \pm \frac{1}{2}\mathbf{h}_i, t_K)\right)\\ &= \frac{f_i(v(\mathbf{x}_K, t_K)) + f_i(v(\mathbf{x}_K \pm \mathbf{h}_i, t_K))}{2} \mp \left|a_i\left(\mathbf{x}_K \pm \frac{1}{2}\mathbf{h}_i, t_K\right)\right| \frac{v(\mathbf{x}_K \pm \mathbf{h}_i, t_K) - v(\mathbf{x}_K, t_K)}{2}.\end{aligned} \tag{3.3}$$

The key idea of the Roe's scheme is to use only grid points, if possible, on one side of the interface for constructing a finite volume scheme so that the RH condition is not violate. This is done through the signs of the Roe speed a_i at midpoints. This key idea was further explored for developing higher order schemes, e.g., the ENO schemes introduced in [16] (see also [28, 29]), by employing extra grid points. To make sure all used grid points locate on one side of the interface, it requires additional decisions and, hence, the ENO schemes are generally sophisticated.

For simplicity, we describe the second order ENO numerical flux here. The ENO uses the sign of the Roe speed to build up upwind scheme. Specifically,

$$\hat{f}_i\left(v(\mathbf{x}_K + \frac{1}{2}\mathbf{h}_i, t_K)\right) = \begin{cases} \hat{f}_i^-\left(v(\mathbf{x}_K + \frac{1}{2}\mathbf{h}_i, t_K)\right), & \text{if } a_i\left(\mathbf{x}_K + \frac{1}{2}\mathbf{h}_i, t_K\right) \geq 0, \\ \hat{f}_i^+\left(v(\mathbf{x}_K + \frac{1}{2}\mathbf{h}_i, t_K)\right), & \text{if } a_i\left(\mathbf{x}_K + \frac{1}{2}\mathbf{h}_i, t_K\right) < 0. \end{cases} \tag{3.4}$$

Additionally, the ENO uses the magnitudes of the finite difference quotient of the i^{th} component of the flux with respect to x_i over the neighbouring intervals to determine which side of grid points are used. In this way, the ENO again tries to use grid points on one side of the discontinuity if possible. More precisely, let

$$f_i(\mathbf{x}_K, t_K; \mathbf{h}_i) := \frac{f_i\big(v(\mathbf{x}_K + \mathbf{h}_i, t_K)\big) - f_i\big(v(\mathbf{x}_K, t_K)\big)}{h_i}$$

$$\text{and} \quad f_i(\mathbf{x}_K, t_K; -\mathbf{h}_i) := \frac{f_i\big(v(\mathbf{x}_K, t_K)\big) - f_i\big(v(\mathbf{x}_K - \mathbf{h}_i, t_K)\big)}{h_i}.$$

In the case that $a_i\left(\mathbf{x}_K + \frac{1}{2}\mathbf{h}_i, t_K\right) \geq 0$, combining with (3.4), the ENO numerical flux is given by

$$\hat{f}_i^-\left(v(\mathbf{x}_K + \frac{1}{2}\mathbf{h}_i, t_K)\right)$$

$$(3.5) \quad = \begin{cases} -\frac{1}{2} f_i\big(v(\mathbf{x}_K - \mathbf{h}_i, t_K)\big) + \frac{3}{2} f_i\big(v(\mathbf{x}_K, t_K)\big), & \text{if } \left|f_i(\mathbf{x}_K, t_K; -\mathbf{h}_i)\right| < \left|f_i(\mathbf{x}_K, t_K; \mathbf{h}_i)\right|, \\ \frac{1}{2} f_i\big(v(\mathbf{x}_K, t_K)\big) + \frac{1}{2} f_i\big(v(\mathbf{x}_K + \mathbf{h}_i, t_K)\big), & \text{if } \left|f_i(\mathbf{x}_K, t_K; -\mathbf{h}_i)\right| \geq \left|f_i(\mathbf{x}_K, t_K; \mathbf{h}_i)\right|. \end{cases}$$

If $a_i\left(\mathbf{x}_K + \frac{1}{2}\mathbf{h}_i, t_K\right) < 0$, then the numerical flux is reconstructed by

$$\hat{f}_i^+\left(v(\mathbf{x}_K + \frac{1}{2}\mathbf{h}_i, t_K)\right)$$

$$= \begin{cases} \frac{1}{2} f_i\big(v(\mathbf{x}_K, t_K)\big) + \frac{1}{2} f_i\big(v(\mathbf{x}_K + \mathbf{h}_i, t_K)\big), & \text{if } \left|f_i(\mathbf{x}_K + \mathbf{h}_i, t_K; -\mathbf{h}_i)\right| < \left|f_i(\mathbf{x}_K + \mathbf{h}_i, t_K; \mathbf{h}_i)\right|, \\ \frac{3}{2} f_i\big(v(\mathbf{x}_K + \mathbf{h}_i, t_K)\big) - \frac{1}{2} f_i\big(v(\mathbf{x}_K + 2\mathbf{h}_i, t_K)\big), & \text{if } \left|f_i(\mathbf{x}_K + \mathbf{h}_i, t_K; -\mathbf{h}_i)\right| \geq \left|f_i(\mathbf{x}_K + \mathbf{h}_i, t_K; \mathbf{h}_i)\right|. \end{cases}$$

In a similar fashion, $\hat{f}_i\left(v(\mathbf{x}_K - \frac{1}{2}\mathbf{h}_i, t_K)\right)$ may be defined accordingly.

4. Block Space-Time Least-Squares Neural Network Method.

Our numerical results show that it is difficult to train the space-time LSNN method for problems with shock formation when the computational domain Ω is relatively large, even though the NN model we used is relatively small for approximating the solution of the underlying problem well. This is due to the nature of nonlinear hyperbolic conservation laws since information is transported from initial and inflow boundary to the rest of the domain along the flow direction. To overcome this training difficulty, we propose the block space-time LSNN method.

For clarity of exposition, let us consider one-dimensional problem defined on $\Omega = (a, b) \times (0, T)$. Without loss of generality, assume that $\tilde{\Gamma}_- = \{(a, t) | \, t \in (0, T)\}$ is the part of the boundary where the characteristic curves enter the domain Ω. Hence,

$$\Gamma_- = \tilde{\Gamma}_- \cup \{(x, 0) | \, x \in (0, T)\}$$

is the "inflow" boundary of Ω. Let m_0 be a positive integer and let

$$\Omega_1 = \left(a, a + \frac{b-a}{m_0}\right) \times \left(0, \frac{T}{m_0}\right) \quad \text{and} \quad \Omega_i = \left(a, a + i\frac{(b-a)}{m_0}\right) \times \left(0, i\frac{T}{m_0}\right) \setminus \Omega_{i-1}$$

for $i = 2, ..., m_0$. The sketch of domains $\{\Omega_i\}_{i=1}^{m_0}$ is presented in Fig 1 and it is clear that $\{\Omega_i\}_{i=1}^{m_0}$ forms a partition of the domain Ω. Denote by $u_i = u|_{\Omega_i}$ the restriction of the solution u of (1.1) on Ω_i, then u_i is the solution of the following problem:

$$(4.1) \qquad \begin{cases} (u_i)_t + \nabla_{\mathbf{x}} \cdot \mathbf{f}(u_i) &= 0, & \text{in } \Omega_i \in \mathbb{R}^2, \\ u_i &= g, & \text{on } \Gamma_-^i = \Gamma_- \cap \partial\Omega_i, \\ u_i &= u_{i-1}, & \text{on } \Gamma_{i-1,i} = \partial\Omega_{i-1} \cap \partial\Omega_i \end{cases}$$

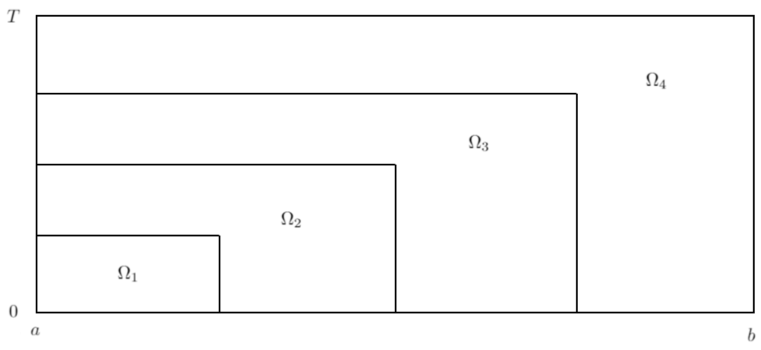


FIG. 1. *Sketch of domains $\{\Omega_i\}_{i=1}^{m_0}$ for $m_0 = 4$*

for $i = 1, ..., m_0$, where $\partial\Omega_0 = \emptyset$.

Define the least-squares functional for problem (4.1) by

$$\mathcal{L}^i\big(v; u_{i-1}, g\big) = \|v_t(\mathbf{x}, t) + \nabla_{\mathbf{x}} \cdot \mathbf{f}(v)\|^2_{0,\Omega_i} + \|v - g\|^2_{0,\Gamma^i_-} + \|v - u_{i-1}\|^2_{0,\Gamma_{i-1,i}}.$$

Then the corresponding discrete least-squares functional $\mathcal{L}^i_{\mathcal{T}}\big(v(\cdot;\boldsymbol{\theta}); u_{i-1}, g\big)$ over the subdomain Ω_i may be defined in a similar fashion as in (2.6). Now, the block space-time LSNN method is to find $u^i_{\mathcal{T}}(\mathbf{z}, \boldsymbol{\theta}^*_i) \in \mathcal{M}(\boldsymbol{\theta}, L)$ such that

$$\mathcal{L}^i_{\mathcal{T}}\big(u^i_{\mathcal{T}}(\cdot, \boldsymbol{\theta}^*_i); u_{i-1}, g\big) = \min_{v \in \mathcal{M}(\boldsymbol{\theta}, L)} \mathcal{L}^i_{\mathcal{T}}\big(v(\cdot;\boldsymbol{\theta});\, u_{i-1}, g\big) = \min_{\boldsymbol{\theta} \in \mathbb{R}^N} \mathcal{L}^i_{\mathcal{T}}\big(v(\cdot;\boldsymbol{\theta}); u_{i-1}, g\big) \tag{4.2}$$

for $i = 1, ..., m_0$.

REMARK 4.1. *The NN model $\mathcal{M}(\boldsymbol{\theta}, L)$ is determined by the first subdomain and will be used for all subdomains. The trained parameter $\boldsymbol{\theta}^*_i$ from the i^{th}-subdomain is a good approximation to the parameters of the $(i+1)^{th}$-subdomain and, hence, may be used as an initial. This is because the solution in the current block is the evolution of the solution in the previous block.*

The block space-time LSNN method is based on a proper partition of the domain Ω depending on the "inflow" boundary of the domain. For example, in one dimension again, if

$$\Gamma_- = \{(x, t) \in [a, b] \times [0, T] | \, x = a, \;\; x = b, \;\; \text{or } , t = 0\},$$

then the domain Ω may be partitioned by time blocks as

$$\Omega_i = (a, b) \times \big((i-1)T/m_0, iT/m_0\big) \tag{4.3}$$

for $i = 1, ..., m_0$. Then the block space-time LSNN method may be defined accordingly.

5. Implementation and Numerical Experiments.

In this section, we present numerical results for one dimensional benchmark test problems. Test problems include scalar nonlinear hyperbolic conservation law: (1) inviscid Burgers equation, i.e., $\mathbf{f}(u) = \frac{1}{2}u^2$ (section 5.1-5.2) and (2) $\mathbf{f}(u) = \frac{1}{4}u^4$ (section 5.3). Additionally, we analyze the effects of integration mesh and network structure in section 5.4; and compare the Roe and ENO schemes in section 5.5.

The domain $\Omega = (a, b) \times (0, T)$ is partitioned into time blocks as (4.3) and m_0 is the number of blocks. Unless otherwise stated, the integration mesh $\mathcal{T}$ is obtained by uniformly partitioning all subdomains Ω_i into identical squares with the mesh size $h = 0.01$ for $i = 1, \cdots, m_0$. To preserve the conservation, the spatial mesh size h_i and the temporal step size τ in both Roe (3.1) and ENO (3.5) schemes are chosen to be the same as the integration mesh size, i.e., $h_i = \tau = h = 0.01$.

The block space-time LSNN method is implemented, and the minimization problem in (4.2) is numerically solved using the Adams version of gradient descent [22] with a fixed or an adaptive learning rate.

As shown in [6], a three-layer NN is needed in order to approximate discontinuous solutions along non-straight line interfaces. Since there is no prior knowledge of geometric shape of the discontinuous interfaces for nonlinear problems, we use three-layer NN models for all test problems. Moreover, the same architecture of three-layer NN models are used for all blocks. As suggested in Remark 4.1, the parameters of the NN for the current block are initialized by the values of the parameters of the NN in the previous block. For the first block, the parameters of the second hidden layer are initialized randomly; and those of the first hidden layer are initialized using the strategy introduced in [?]. For convenience of readers, we briefly describe here. Let $\boldsymbol{\omega}_i \in \mathcal{S}^1$ and $b_i \in \mathbb{R}$ be the weights and bias of the i^{th} neuron of the first hidden layer of the first block NN model, respectively, where $\mathcal{S}^1$ is the unit circle in $\mathbb{R}^2$. Initial of $\{(\boldsymbol{\omega}_i, b_i)\}_{i=1}^{n_1}$ is chosen so that the hyper-planes $\{\boldsymbol{\omega}_i \cdot (x,t) = b_i)\}_{i=1}^{n_1}$ form a uniform partition of the first block Ω_1. In addition, without an effective training strategy, we observe from the experiment that adding a weight α to the L^2 loss of the initial condition is helpful for the training. Specifically, the following least-squares functional is used in the implementation

$$\mathcal{L}^i\big(v; u_{i-1}, g\big) = \|v_t(\mathbf{x},t) + \nabla_{\mathbf{x}} \cdot \mathbf{f}(v)\|^2_{0,\Omega_i} + \|v-g\|^2_{0,\Gamma^i_-} + \alpha\|v-u_{i-1}\|^2_{0,\Gamma_{i-1,i}} \quad \text{for } i = 1, \cdots, m_0 \tag{5.1}$$

and the weight α is empirically determined.

Let u_i be the solution of the problem in (4.1) and $u_{i,\mathcal{T}}$ be the NN approximation. Using all quadrature points including those near the discontinuity, the relative error in the L^2 norm for each block is reported in Table 1-8. Note that the points near discontinuity are usually excluded when reporting the error of existing traditional methods. The network structure is expressed as 2-n_1-n_2-1 for a three-layer network with n_1 and n_2 neurons in the respective first and second layers. The traces of the exact solution and the numerical approximation are depicted in Figures 2-7 on a plane perpendicular to the space-time plane. Those traces exhibit the capability of the numerical approximation in resolving the shock/rarefaction. Since those planes are generally not perpendicular to the discontinuous interface, the errors shown in those traces are larger than the actual error.

5.1. Riemann problem for the inviscid Burgers equation. For the one dimensional inviscid Burgers equation, $\mathbf{f}(u) = f(u) = \frac{1}{2}u^2$, we report numerical results for the corresponding Riemann problem where the initial condition with a single discontinuity is given by:

$$u_0(x) = \begin{cases} u_{_L}, & \text{if } x \le 0, \\ u_{_R}, & \text{if } x > 0. \end{cases} \tag{5.2}$$

When $u_{_L} > u_{_R}$, the characteristic lines intersect and a shock forms immediately for $t > 0$. The weak solution is given by

$$u(x,t) = \begin{cases} u_{_L}, & \text{if } x \le st, \\ u_{_R}, & \text{if } x > st, \end{cases} \tag{5.3}$$

with the shock speed determined by the RH condition

$$s = \frac{f(u_{_L}) - f(u_{_R})}{u_{_L} - u_{_R}}.$$

When $u_{_L} < u_{_R}$, the range of influence of all points in $\mathbb{R}$ is a proper subset of $\mathbb{R} \times [0, \infty)$. This fact implies that the weak solution of the scalar hyperbolic conservation law is not unique. To ensure the underlying Cauchy problem having a unique solution over the whole domain $\mathbb{R}\times[0,\infty)$, the so-called vanishing viscosity weak solution is introduced (see, e.g., [13, 23, 31]) and given by

$$u(x,t) = \begin{cases} u_{\scriptscriptstyle L}, & \text{if} \quad x < u_{\scriptscriptstyle L} t, \\ x/t, & \text{if} \quad u_{\scriptscriptstyle L} t \le x \le u_{\scriptscriptstyle R} t, \\ u_{\scriptscriptstyle R}, & \text{if} \quad x > u_{\scriptscriptstyle R} t. \end{cases}$$

5.1.1. Shock formation. The first test problem is corresponding to the case

$$u_{\scriptscriptstyle L} = 1 > 0 = u_{\scriptscriptstyle R}$$

with a computational domain $\Omega = (-1,2)\times(0,1)$. The inflow boundary is

$$\Gamma_- = \Gamma_-^L \cup \Gamma_-^R \equiv \{(-1,t) : t\in[0,1]\} \cup \{(2,t) : t \in [0,1]\}$$

with the boundary conditions: $g = u_{\scriptscriptstyle L}$ on Γ_-^L and $g = u_{\scriptscriptstyle R}$ on Γ_-^R. The block space-time LSNN method is employed with $m_0 = 5$ blocks, a fixed learning rate 0.003, and 30000 iterations for each block.

The set of experiments is done by using the numerical fluxes of Roe (3.1) and the second order ENO (3.5). By choosing $\alpha = 20$ in (5.1), numerical results of both schemes are reported in Table 1. Since the results of the Roe flux are similar, Figure 2 (b)-(f) only depict the traces of the exact and numerical solution generated by the ENO flux on the planes $t = iT/m_0$ for $i = 1, \cdots, m_0$. Clearly, the block space-time LSNN method with a conservative scheme is able to resolve the shock and accurately approximate the solution without the Gibbs phenomena. For this simple Riemann problem, Roe and ENO schemes produce similar results. Given the additional evaluations involving in the ENO scheme, we do not observe many advantages of using higher-order scheme for flux reconstruction despite the fact that ENO performs slightly better in the L^2 relative errors.

TABLE 1
Relative errors of Riemann problem (shock) for Burgers' equation using Roe and ENO fluxes

Network structure	Block	Roe flux $\frac{\|u_i - u_{i,\mathcal{T}}\|_0}{\|u_i\|_0}$	ENO flux $\frac{\|u_i - u_{i,\mathcal{T}}\|_0}{\|u_i\|_0}$
2-10-10-1	Ω_1	0.049553	0.030901
2-10-10-1	Ω_2	0.046321	0.030178
2-10-10-1	Ω_3	0.044123	0.028984
2-10-10-1	Ω_4	0.042621	0.028791
2-10-10-1	Ω_5	0.041182	0.032253

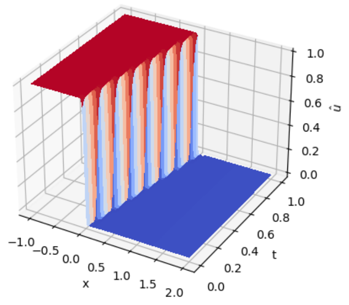
5.1.2. Rarefaction waves. The second test problem is corresponding to the case

$$u_{\scriptscriptstyle L} = 0 < 1 = u_{\scriptscriptstyle R}$$

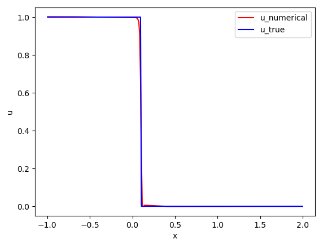
with a computational domain $\Omega = (-1,2)\times(0,0.2)$. The inflow boundary is

$$\Gamma_- = \Gamma_-^L \cup \Gamma_-^R \equiv \{(-1,t) : t\in[0,0.2]\} \cup \{(2,t) : t \in [0,0.2]\}$$

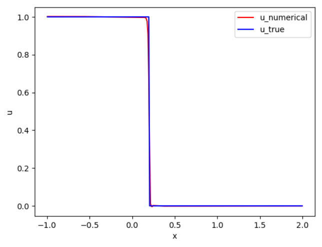
with the boundary conditions: $g = u_{\scriptscriptstyle L}$ on Γ_-^L and $g = u_{\scriptscriptstyle R}$ on Γ_-^R. The block space-time LSNN method is employed with $m_0 = 2$ blocks, a fixed learning rate 0.003, and 20000 iterations for each block.



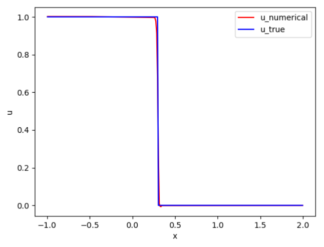
(a) Network approximation $u_{\mathcal{T}}$ on Ω



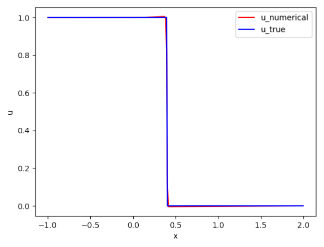
(b) Traces of exact and numerical solutions $u_{1,\mathcal{T}}$ on the plane $t = 0.2$



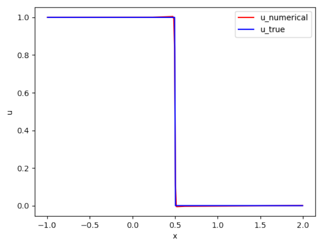
(c) Traces of exact and numerical solutions $u_{2,\mathcal{T}}$ on the plane $t = 0.4$



(d) Traces of exact and numerical solutions $u_{3,\mathcal{T}}$ on the plane $t = 0.6$



(e) Traces of exact and numerical solutions $u_{4,\mathcal{T}}$ on the plane $t = 0.8$



(f) Traces of exact and numerical solutions $u_{5,\mathcal{T}}$ on the plane $t = 1.0$

FIG. 2. *Approximation results of Riemann problem (shock) for Burgers' equation using Roe flux*

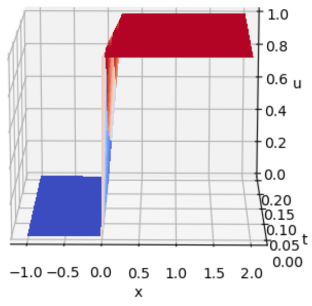
Numerical results of a 2-10-10-1 network using the Roe flux (3.1) are reported in Table 2. The traces of the exact and numerical solutions in Fig. 3 indicate that Roe's scheme fails to resolve the rarefaction. For the traditional mesh-based method, it is well-known that Roe's scheme is not able to compute the physical solution of the rarefaction wave. This is because the scheme approximates the numerical flux depending the sign of the speed a_i (3.2) at midpoints. If the sign differs on two sides and $u(x, t)$ travels slower on the left, Roe's scheme may not be able to capture such behavior. From this perspective, we observe certain limitations of using such conservative scheme and that the discretization scheme is very important for the block space-time LSNN method.

REMARK 5.1. *In* [15, 23], the authors proposed the "entropy fix" traditional mesh-based approach to address such issue. In a forthcoming paper, we will propose a discretization scheme for the LSNN method which is capable of resolving the rarefaction.

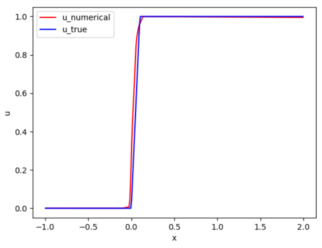
TABLE 2
Relative errors of Riemann problem (rarefaction) for Burgers' equation using Roe flux

Network structure	Time block	$\frac{\|u_i - u_{i,\mathcal{T}}\|_0}{\|u_i\|_0}$
2-10-10-1	Ω_1	0.047435
2-10-10-1	Ω_2	0.074521

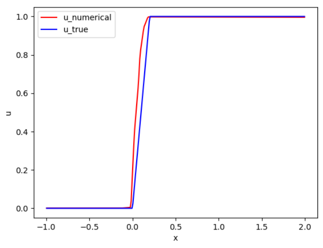
5.2. Inviscid Burgers equation with smooth initial condition. The third problem is again the Burgers equation defined on the computational domain $\Omega = (0, 2) \times (0, 0.4)$ with the



(a) Exact solution u on Ω



(b) Traces of exact and numerical solutions $u_{1,\mathcal{T}}$ on the plane $t = 0.1$



(c) Traces of exact and numerical solutions $u_{2,\mathcal{T}}$ on the plane $t = 0.2$

FIG. 3. *Approximation results of Riemann problem (rarefaction) for Burgers' equation using Roe flux*

inflow boundary

$$\Gamma_- = \Gamma_-^L \cup \Gamma_-^R \equiv \{(0,t) : t \in [0, 0.4]\} \cup \{(2,t) : t \in [0, 0.4]\}$$

and a sinusoidal initial condition

$$u_0(x) = 0.5 + \sin(\pi x).$$

The shock forms at $t = 1/\pi \approx 0.318$. Since the exact solution of the test problem is defined implicitly, in order to measure the quality of the NN approximation, we generate a benchmark reference solution $\hat{u}$ using the traditional mesh-based approach. Specifically, the third order accurate WENO scheme [28] is employed for the spatial discretization with a fine grid ($\Delta x = 0.001$ and $\Delta t = 0.0002$) on the computational domain Ω; and the fourth order Runge-Kutta method is used for the temporal discretization [32]. The block space-time LSNN method is implemented with $m_0 = 8$ blocks and an adaptive learning rate which starts at 0.005 and decays by half for every 25000 iterations. The learning rate decay strategy is employed with 50000 iterations on each time block.

Since the initial condition of the test problem is a smooth function, it is expected that a network with additional neurons is needed for approximation. Choosing $\alpha = 5$ in (5.1), the numerical results of a 2-30-30-1 network using the ENO flux in (3.5) are reported in Table 3. Figure 4 depicts the traces of the reference solution and numerical approximation on the plane $t = iT/m_0$ for $i = 1, \cdots, m_0$. We observe some error accumulation when block evolves, and the block space-time LSNN method can resolve the shock. It is noticeable in Fig. 4 that approximation near the local maximum is poor. Possibly this is due to inaccuracy of the second order ENO scheme for complicated initial data. A new and accurate discretization scheme for the LSNN method will be reported in a forthcoming paper [5].

5.3. Riemann problem with $f(u) = \frac{1}{4}u^4$. The fourth numerical experiment is the Riemann shock problem with a convex flux $\mathbf{f}(u) = f(u) = \frac{1}{4}u^4$. We choose the initial condition

$$u_L = 1 > 0 = u_R$$

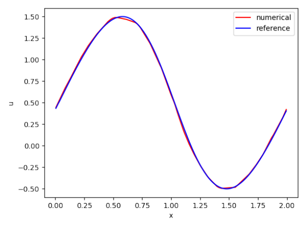
in (5.2), then the weak solution is given by (5.3) with the speed $s = 1/4$. The computational domain of problem is given by $\Sigma = (-1, 1) \times (0, 1)$ and the inflow boundary is

$$\Gamma_- = \Gamma_-^L \cup \Gamma_-^R \equiv \{(-1,t) : t \in [0, 1]\} \cup \{(1,t) : t \in [0, 1]\}$$

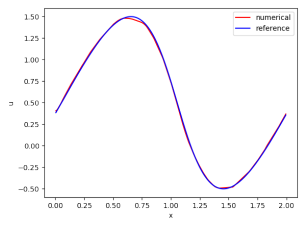
with the boundary conditions: $g = 1$ on Γ_-^L and $g = 0$ on Γ_-^R.

Table 3
Relative errors of Burgers' equation with a sinusoidal initial condition using ENO flux

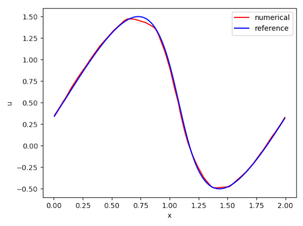
Network structure	Time block	$\frac{\|\hat{u}_i - u_{i,\mathcal{T}}\|_0}{\|\hat{u}_i\|_0}$
2-30-30-1	Ω_1	0.010461
2-30-30-1	Ω_2	0.012517
2-30-30-1	Ω_3	0.019772
2-30-30-1	Ω_4	0.022574
2-30-30-1	Ω_5	0.029011
2-30-30-1	Ω_6	0.038852
2-30-30-1	Ω_7	0.075888
2-30-30-1	Ω_8	0.078581



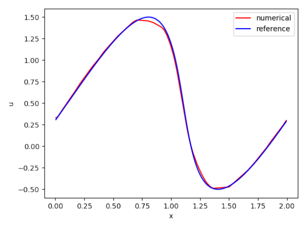
(a) Traces of reference and numerical solutions $u_{1,\mathcal{T}}$ on the plane $t = 0.05$



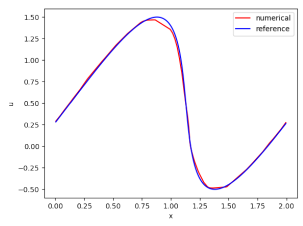
(b) Traces of reference and numerical solutions $u_{2,\mathcal{T}}$ on the plane $t = 0.1$



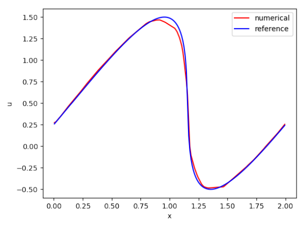
(c) Traces of reference and numerical solutions $u_{3,\mathcal{T}}$ on the plane $t = 0.15$



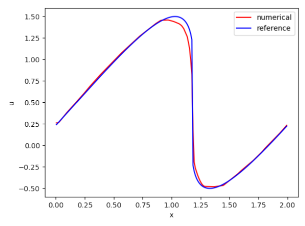
(d) Traces of reference and numerical solutions $u_{4,\mathcal{T}}$ on the plane $t = 0.2$



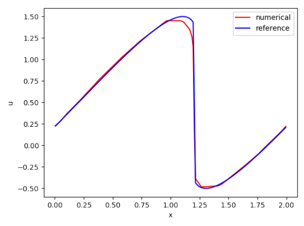
(e) Traces of reference and numerical solutions $u_{5,\mathcal{T}}$ on the plane $t = 0.25$



(f) Traces of reference and numerical solutions $u_{6,\mathcal{T}}$ on the plane $t = 0.3$



(g) Traces of reference and numerical solutions $u_{7,\mathcal{T}}$ on the plane $t = 0.35$



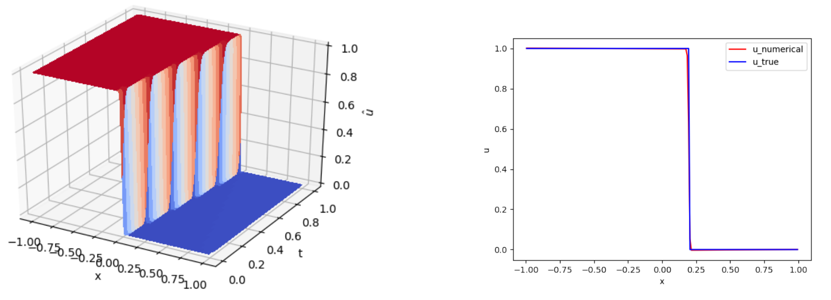
(h) Traces of reference and numerical solutions $u_{8,\mathcal{T}}$ on the plane $t = 0.4$

Fig. 4. *Approximation results of Burgers' equation with a sinusoidal initial using ENO flux*

Empirically, we choose $\alpha = 20$ in (5.1) in the implementation. Employing the block space-time LSNN method with $m_0 = 5$ blocks, a fixed learning rate 0.003 and 30000 iterations for each block, we report the numerical results of a 2-10-10-1 network in Table 4 and Fig. 5. Obviously, the discontinuous interface can be accurately captured as the NN approximation is almost overlapped with the exact solution, which suggests that the LSNN method is not only capable of solving the Burgers equation but also the problem with a general convex flux.

TABLE 4
Relative errors of Riemann problem (shock) with $f(u) = \frac{1}{4}u^4$ using Roe flux

Network structure	Block	$\frac{\|u_i - u_{i,\mathcal{T}}\|_0}{\|u_i\|_0}$
2-10-10-1	Ω_1	0.035034
2-10-10-1	Ω_2	0.036645
2-10-10-1	Ω_3	0.036798
2-10-10-1	Ω_4	0.037217
2-10-10-1	Ω_5	0.037451



(a) Network approximation $u_{\mathcal{T}}$ on Ω (b) Traces of exact and numerical solutions $u_{5,\mathcal{T}}$ on the plane $t = 1.0$

FIG. 5. *Approximation results of Riemann problem (shock) with $f(u) = \frac{1}{4}u^4$ using Roe flux*

5.4. Effects of integration mesh and network structure. The goal of this section is to analyze the effects of integration mesh and network structure for the LSNN method. Specifically, we use the inviscid Burgers equation defined on the computational domain $\Omega = (-1, 2) \times (0, 0.6)$ with a continuous piece-wise linear initial condition

$$u_0(x) = \begin{cases} 1, & \text{if} \quad x < 0, \\ 1 - 2x, & \text{if} \quad 0 \le x \le 1/2, \\ 0, & \text{if} \quad x > 1/2. \end{cases}$$

The inflow boundary is

$$\Gamma_- = \Gamma_-^L \cup \Gamma_-^R \equiv \{(-1, t) : t \in [0, 0.6]\} \cup \{(2, t) : t \in [0, 0.6]\},$$

with the boundary conditions: $g = 1$ on Γ_-^L and $g = 0$ on Γ_-^R. Even though the initial value u_0 is continuous, the shock will appear at some point since $u(x, t)$ travels faster on the left-hand side than on the right-hand side. Specifically, when $t < 1/2$, the solution is continuous and it is determined by the characteristic lines as well as the initial conditions:

$$u(x,t) = \begin{cases} 1, & \text{if} \quad x < t < 1/2, \\ \dfrac{1-2x}{1-2t}, & \text{if} \quad t \le x \le 1/2, \\ 0, & \text{if} \quad x > 1/2. \end{cases}$$

When $t > 1/2$, the shock forms and the desired weak solution satisfying RH condition is given by

$$u(x,t) = \begin{cases} 1, & \text{if} \quad x < (2t+1)/4, \\ 0, & \text{if} \quad x \ge (2t+1)/4. \end{cases}$$

The block space-time LSNN is implemented with $m_0 = 3$ blocks, a fixed learning rate 0.003 and $\alpha = 10$ in the training. In order to explore the network approximation power, we do not constrain the number of iterations on each block. The stopping criteria for the gradient descent solver is set as follows: the solver stops when the loss function (5.1) decreases within 0.1% in the last 2000 iterations.

The first set of experiments is to observe the impact of integration mesh size when a fixed network structure 2-10-10-1 is used. Starting with the same initialization for both layers, the relative L^2 errors of the ENO scheme on uniform meshes with different mesh size are reported in Table 5. The results display that the error decreases as the integration mesh size reduces, but the decreasing rate is small when the mesh size is sufficiently fine. This indicates that $h = 0.01$ is fine enough to accurately evaluate the discrete LS functional in (4.2) for the given 2-10-10-1 network.

Table 5
Relative L^2 errors of the problem with a piece-wise linear initial using different integration mesh sizes

Time block	Integration mesh size				
	$h = 0.1$	$h = 0.05$	$h = 0.025$	$h = 0.01$	$h = 0.005$
Ω_1	0.056712	0.029366	0.017376	0.013946	0.013121
Ω_2	0.079633	0.064069	0.048971	0.037891	0.036501
Ω_3	0.120129	0.115123	0.102844	0.084728	0.081209

The second set of numerical tests is to investigate the effect of network structure. To accurately evaluate the functional, we use a fine integration mesh with size $h = 0.005$. The approximation errors generated by four different network structures using the ENO scheme are presented in Table 6. As expected, the approximation error decreases as the number of neurons in the network increases.

Table 6
Relative L^2 errors of the problem with a piece-wise linear initial using different network structures

Time block	Network structure			
	2-4-4-1	2-7-7-1	2-10-10-1	2-13-13-1
Ω_1	0.021699	0.017004	0.013121	0.009895
Ω_2	0.047953	0.040521	0.036501	0.028159
Ω_3	0.095649	0.088201	0.081209	0.072244

Tables 5 and 6 indicate that the approximation error grows significantly as the block evolves. To address this issue, a simple way is to decrease the size of each block, or equivalently, increase the number of blocks m_0. Specifically, increasing m_0 from 3 to 6, Table 7 reports numerical results of the ENO scheme using $h = 0.01$, which is better than the results in the corresponding column in Table 5. Due to the increasing computational cost when using a smaller block, in practice we need to balance the cost and the accuracy when choosing the size of the block in the block space-time LSNN method.

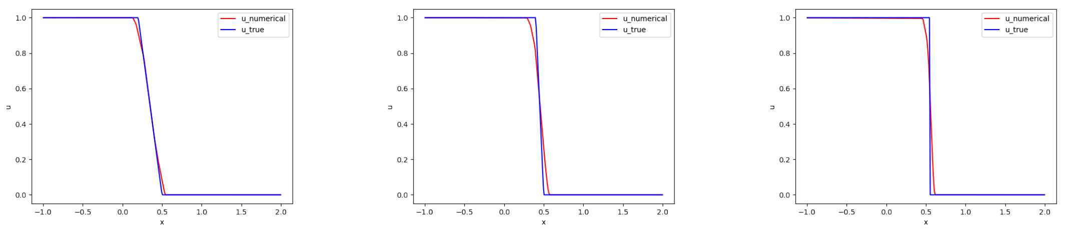
Table 7
Relative L^2 errors of the problem with a piece-wise linear initial using ENO flux with $m_0 = 6$

Network structure	Block	$\frac{\|u_i-u_{i,\mathcal{T}}\|_0}{\|u_i\|_0}$
2-10-10-1	Ω_1	0.008958
2-10-10-1	Ω_2	0.014842
2-10-10-1	Ω_3	0.019282
2-10-10-1	Ω_4	0.025562
2-10-10-1	Ω_5	0.047431
2-10-10-1	Ω_6	0.052606

5.5. Comparison of the Roe and ENO schemes. This choosing set of numerical experiments aims to compare numerical performances of the LSNN method using the Roe (3.1) and ENO (3.5) schemes. The test problem in section 5.4 is used, and the block space-time LSNN method is employed with $m_0 = 6$ blocks, the 2-10-10-1 network structure, and with a uniform integration mesh of the size $h = 0.01$. We note that the ENO performs better than the Roe in the relative L^2 norm (see Table 8) and near the discontinuous interface (see Fig. 6 and 7). Note that the former is more expansive than the latter in the computational cost due to additional testing.

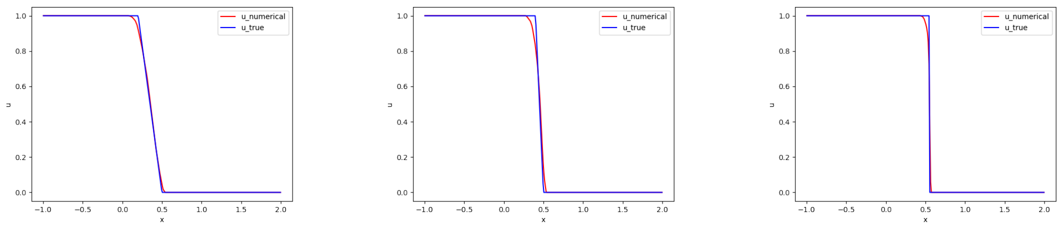
Table 8
Relative errors of Burgers' equation with a piece-wise linear initial condition

Network structure	Block	Roe flux $\frac{\|u_i-u_{i,\mathcal{T}}\|_0}{\|u_i\|_0}$	ENO flux $\frac{\|u_i-u_{i,\mathcal{T}}\|_0}{\|u_i\|_0}$
2-10-10-1	Ω_1	0.008333	0.008958
2-10-10-1	Ω_2	0.017503	0.014842
2-10-10-1	Ω_3	0.025342	0.019282
2-10-10-1	Ω_4	0.036211	0.025562
2-10-10-1	Ω_5	0.061251	0.047431
2-10-10-1	Ω_6	0.068512	0.052606



(a) Traces of exact and numerical solutions $u_{2,\mathcal{T}}$ on the plane $t = 0.2$
(b) Traces of exact and numerical solutions $u_{4,\mathcal{T}}$ on the plane $t = 0.4$
(c) Traces of exact and numerical solutions $u_{6,\mathcal{T}}$ on the plane $t = 0.6$

Fig. 6. *Approximation results of Burgers' equation with a piece-wise linear initial using Roe flux*



(a) Traces of exact and numerical solutions $u_{2,\mathcal{T}}$ on the plane $t = 0.2$

(b) Traces of exact and numerical solutions $u_{4,\mathcal{T}}$ on the plane $t = 0.4$

(c) Traces of exact and numerical solutions $u_{6,\mathcal{T}}$ on the plane $t = 0.6$

FIG. 7. *Approximation results of Burgers' equation with a piece-wise linear initial using ENO flux*

6. Discussions and Conclusions.

The block space-time LSNN method is proposed for solving scalar nonlinear hyperbolic conservation laws. The least-squares formulation is a direct application of the least-squares principle to the underlying problem: the equation, the inflow boundary condition, and the initial condition. The block space-time version of the LSNN method is introduced to compensate with some uncertainty of the not well-understood nonlinear optimization procedure.

How to approximate the differential operator in the least-squares functional is critical for the success of the space-time LSNN method. As mentioned in the introduction, existing NN-based methods are not applicable to the inviscid Burgers equation whose solution is discontinuous. Employing the Roe and the second order ENO schemes, we show numerically that the resulting LSNN method is capable of resolving the shock without smearing and oscillations. Moreover, the LSNN method has much less degrees of freedom (DoF) than traditional mesh-based methods.

Despite the great potential demonstrated in this paper, the current version of the LSNN method needs to be improved in both accuracy and efficiency in order to grow into a viable numerical method. First, the method is inaccurate for complicated initial condition; moreover, it has limitations for problems with rarefaction waves and with non-convex spatial fluxes. To overcome these deficiency, we recently study a new version of the LSNN method that uses a novel and accurate finite volume method developed in [5]. Second, the computational cost of the current version of the LSNN method is expensive because of the resulting non-convex optimization. To reduce the cost, we will explore physical meanings of non-linear parameters of NNs (see, e.g., [24]) to develop good initial strategies and then employ fast iterative solvers such as methods of BFGS type (the Broyden–Fletcher–Goldfarb–Shanno algorithm).

REFERENCES

[1] Y. Bar-Sinai, S. Hoyer, J. Hickey, and M. P. Brenner. Learning data-driven discretizations for partial differential equations. *Proceedings of the National Academy of Science of USA*, 116 (31):15344–15349, 2019.
[2] P. Bochev and J. Choi. Improved least-squares error estimates for scalar hyperbolic problems. *Computational Methods in Applied Mathematics*, 1(2):115–124, 2001.
[3] F. Brezzi, L. D. Marini, and E. Süli. Discontinuous galerkin methods for first-order hyperbolic problems. *Mathematical models and methods in applied sciences*, 14(12):1893–1903, 2004.
[4] E. Burman. A posteriori error estimation for interior penalty finite element approximations of the advection-reaction equation. *SIAM journal on numerical analysis*, 47(5):3584–3607, 2009.
[5] Z. Cai, J. Chen, and M. Liu. Finite volume least-squares neural network (FV-LSNN) method for scalar nonlinear hyperbolic conservation laws. *arXiv preprint arXiv:2110.10895*, 2021.
[6] Z. Cai, J. Chen, and M. Liu. Least-squares ReLU neural network (LSNN) method for linear advection-reaction equation. *Journal of Computational Physics*, 110514, 2021.
[7] Z. Cai, J. Chen, M. Liu, and X. Liu. Deep least-squares methods: An unsupervised learning-based numerical method for solving elliptic PDEs. *Journal of Computational Physics*, 109707, 2020.

[8] W. Dahmen, C. Huang, C. Schwab, and G. Welper. Adaptive petrov–galerkin methods for first order transport equations. *SIAM journal on numerical analysis*, 50(5):2420–2445, 2012.

[9] H. De Sterck, T. A. Manteuffel, S. F. McCormick, and L. Olson. Least-squares finite element methods and algebraic multigrid solvers for linear hyperbolic PDEs. *SIAM Journal on Scientific Computing*, 26(1):31–54, 2004.

[10] H. De Sterck, T. A. Manteuffel, S. F. McCormick, and L. Olson. Numerical conservation properties of H(div)-conforming least-squares finite element methods for the burgers equation. *SIAM Journal on Scientific Computing*, 26(5):1573–1597, 2005.

[11] L. Demkowicz and J. Gopalakrishnan. A class of discontinuous petrov–galerkin methods. part I: The transport equation. *Computer Methods in Applied Mechanics and Engineering*, 199(23-24):1558–1572, 2010.

[12] W. E and B. Yu. The deep ritz method: A deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1):1–12, 3 2018.

[13] E. Godlewski and P.-A. Raviart. *Numerical approximation of hyperbolic systems of conservation laws*, volume 118. Springer Science & Business Media, 2013.

[14] D. Gottlieb and C.-W. Shu. On the gibbs phenomenon and its resolution. *SIAM review*, 39(4):644–668, 1997.

[15] A. Harten. High resolution schemes for hyperbolic conservation laws. *Journal of computational physics*, 135(2):260–278, 1997.

[16] A. Harten, B. Engquist, S. Osher, and S. R. Chakravarthy. Uniformly high order accurate essentially non-oscillatory schemes, iii. In *Upwind and high-resolution schemes*, pages 218–290. Springer, 1987.

[17] J. S. Hesthaven. *Numerical methods for conservation laws: From analysis to algorithms*. SIAM, 2017.

[18] J. S. Hesthaven and T. Warburton. *Nodal discontinuous Galerkin methods: algorithms, analysis, and applications*. Springer Science & Business Media, 2007.

[19] P. Houston, J. A. Mackenzie, E. Süli, and G. Warnecke. A posteriori error analysis for numerical approximations of friedrichs systems. *Numerische Mathematik*, 82(3):433–470, 1999.

[20] P. Houston, R. Rannacher, and E. Süli. A posteriori error analysis for stabilised finite element approximations of transport problems. *Computer methods in applied mechanics and engineering*, 190(11-12):1483–1508, 2000.

[21] D. Z. Kalchev and T. A. Manteuffel. A least-squares finite element method based on the helmholtz decomposition for hyperbolic balance laws. *arXiv preprint arXiv:1911.05831v2*, 2020.

[22] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Representation Learning, San Diego*, 2015.

[23] R. J. LeVeque. *Numerical methods for conservation laws*, volume 132. Springer, 1992.

[24] M. Liu, Z. Cai, and J. Chen. Adaptive two-layer ReLU neural network: I. best least-squares approximation. *arXiv preprint arXiv:2107.08935v1*, 2021.

[25] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics informed deep learning (part I): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017.

[26] M. Raissi, P. Perdikaris, and G. E. 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.

[27] P. L. Roe. Approximate riemann solvers, parameter vectors, and difference schemes. *Journal of computational physics*, 43(2):357–372, 1981.

[28] C.-W. Shu. Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws. In *Advanced numerical approximation of nonlinear hyperbolic equations*, pages 325–432. Springer, 1998.

[29] C.-W. Shu and S. Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes. *Journal of computational physics*, 77(2):439–471, 1988.

[30] J. Sirignano and K. Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1139–1364, 2018.

[31] J. W. Thomas. *Numerical partial differential equations: finite difference methods*, volume 22. Springer Science & Business Media, 2013.

[32] Y. Wang, Z. Shen, Z. Long, and B. Dong. Learning to discretize: solving 1D scalar conservation laws via deep reinforcement learning. *arXiv preprint arXiv:1905.11079*, 2019.