Commun. Comput. Phys.
doi: 10.4208/cicp.OA-2022-0285

Vol. **34**, No. 2, pp. 290-317
August 2023

Neural Networks with Local Converging Inputs (NNLCI) for Solving Conservation Laws, Part I: 1D Problems

Haoxiang Huang[1], Vigor Yang[2] and Yingjie Liu[3,*]

[1] *Woodruff School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA.*
[2] *Daniel Guggenheim School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA.*
[3] *School of Mathematics, Georgia Institute of Technology, Atlanta, GA 30332, USA.*

Received 7 November 2022; Accepted (in revised version) 19 July 2023

Abstract. A novel neural network method is developed for solving systems of conservation laws whose solutions may contain abrupt changes of state, including shock waves and contact discontinuities. In conventional approaches, a low-cost solution patch is usually used as the input to a neural network for predicting the high-fidelity solution patch. With that technique, however, there is no way to distinguish a smeared discontinuity from a smooth solution with large gradient in the input, and the two almost identical inputs correspond to two fundamentally different high-fidelity solution patches in training and predicting. To circumvent this difficulty, we use local patches of two low-cost numerical solutions of the conservation laws in a converging sequence as the input to a neural network. The neural network then makes a correct prediction by identifying whether the solution contains discontinuities or just smooth variations with large gradients, because the former becomes increasingly steep in a converging sequence in the input, and the latter does not. The inputs can be computed from low-cost numerical schemes with coarse resolution, in a local domain of dependence of a space-time location where the prediction is to be made. Despite smeared input solutions, the output provides sharp approximations of solutions containing shock waves and contact discontinuities. The method works effectively not only for regions with discontinuities, but also for smooth regions of the solution. It is efficient to implement, once trained, and has broader applications for different types of differential equations.

AMS subject classifications: 65L99, 65M99, 65N99, 68T99, 76L05

Key words: Neural network, neural networks with local converging inputs, physics informed machine learning, conservation laws, differential equation, multi-fidelity optimization.

*Corresponding author. *Email addresses:* `yingjie@math.gatech.edu` (Y. Liu), `hcwong@gatech.edu` (H. Huang), `vigor.yang@aerospace.gatech.edu` (V. Yang)

1 Introduction

There has been extensive research on numerical methods for solving conservation laws whose solutions may contain shock waves and contact discontinuities. Proposed methods of approaching the problem include the Godunov [14], MUSCL [45,46], ENO [16,40], and WENO [18,23] schemes, adaptive mesh refinement [4], moving mesh [42], hierarchical reconstruction [24,25,52], preconditioning schemes [17,54], space-time methods [50], and many others. These numerical techniques have been broadly implemented to study a wide variety of fluid dynamics and combustion problems [43,44,48,53]. The development of machine learning techniques for solving hyperbolic conservation laws, however, is still in an early stage.

In the past decade, data-driven modeling in machine learning has been developed for many scientific and engineering disciplines, including image processing, biomedical applications, and engineering design optimization [6,7,10,20,21,29,31]. Advances in computational resources, including improvements in graphics processing units (GPUs) and tensor processing units (TPUs), have accelerated training in deep learning frameworks such as TensorFlow and PyTorch for computer vision [51], natural language processing [47], and other applications [8].

Recently, Sirignano and Spiliopoulos [41] approximate the unknown solution as a mapping from a space-time location to the solution value there with a deep neural network (Deep Galerkin Method), incorporating the finite difference residue error and initial and boundary constraints in the loss function. E and Yu [12] incorporate the Ritz energy of a finite element method into the loss function of a neural network (Deep Ritz Method.) Raissi *et al.* [33] develop physics-informed neural networks (PINN) by employing an automatic differentiation [3] to define the residue error in the loss function. This method has achieved much success in data-driven methods for identifying nonlinear PDEs. It is capable of predicting fluid flow dynamics with given governing PDEs, such as the Navier-Stokes equations [34,35]. PINN has also been applied to solve several other physical problems. These include lattice Boltzmann equations with the Bhatnagar-Gross-Krook collision [26] and parabolic PDE, for heat transfer problems [5,22], and so on. To explore the possibility that temporal evolution of learning could take place a priori in the absence of data, Psaros et al. [32] extended PINN to a meta-learning framework. The PINN algorithm usually is much more complex than conventional schemes for evolution equations, because in the loss function the equation is evaluated at space-time-filling points. It still remains a challenge for this algorithm to capture discontinuities and their ensuing interactions in the domain of interest.

In [9,27], finite expansions of neural networks were introduced to form a mapping that can map the entire initial value and a spatial location to a high-fidelity solution at the same location at a later time. Earlier works [15,38] had identified shock waves or contact surfaces from smeared solutions. In [28], the Rankine-Hugoniot jump conditions were added as a constraint to the loss function of the neural network for solving Riemann problems. In [11,37], neural networks were used to detect discontinuities and determine the magnitude of artificial viscosity needed in the presence of discontinuities for the scheme, using a local solution as the input. In [2],neural networks were used to detect discontinuities and tune the slope limiter of the scheme, using a local solution as the input. In [13], difficulties arising from the approximation of the singularity of the function by the output were circumvented by a post-processing approach. There remain, however, basic questions as to whether a neural network is capable of directly predicting the solution of the Euler equations with accuracy comparable to those of high resolution schemes.

In the present study, we introduce a novel neural network method with local converging inputs (NNLCI) as post-processor to solve conservation laws whose solutions may contain rapid variations of state, such as shock and contact discontinuities. For a given initial condition, two approximate solutions of the conservation laws are implemented in sequence (converging to the solution) and serve as the input to a neural network. The neural network then makes a correct prediction by identifying whether the solution contains discontinuities or just smooth variations with large gradients, because the former becomes increasingly steep in a converging sequence in the input, and the latter does not. The two approximate solutions are computed from low-cost numerical schemes with coarse resolution, in a local domain of dependence of the space-time location where the prediction is to be made. The inputs can be generated in a variety of ways: a single scheme on two different coarse grids (with one grid coarser than the other), a single scheme with two different numerical diffusion coefficients on the same coarse grid, or two schemes of different order of accuracy on the same or slightly different coarse grid. The NNLCI method works effectively, not only for discontinuities, but also for smooth areas of the solution. It is applicable to a wide variety of differential equations, and it is a local post-processing-type solver, so a low cost (e.g., first-order) scheme can be used on coarse grids to compute inputs.

The paper is structured as follows. Section 2 describes the problem setup. The proposed neural network with local converging inputs (NNLCI) is introduced in Sections 3 and 4. Section 5 introduces variants of the neural network method. Section 6 presents conclusions.

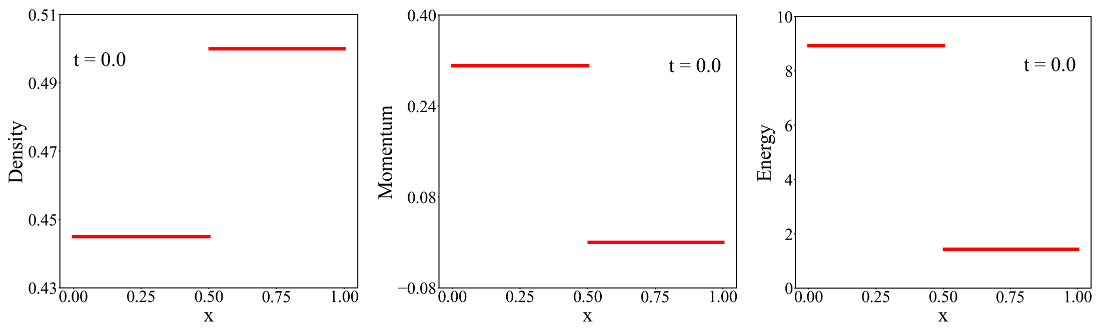
2 Problem setup

In this study, we consider 1-D conservation laws in the following form

$$\frac{\partial U}{\partial t}+\frac{\partial f(U)}{\partial x}=0, \quad x\in\Omega\subset\mathcal{R}, \quad t\in[0,T], \tag{2.1}$$

where Ω is an interval. The 1-D Euler equations for an ideal gas are

$$\frac{\partial}{\partial t}\begin{pmatrix}\rho\\ \rho v\\ E\end{pmatrix}+\frac{\partial}{\partial x}\begin{pmatrix}\rho v\\ \rho v^2\\ v(E+p)\end{pmatrix}=0, \quad x\in\Omega\subset\mathcal{R}, \quad t\in[0,T], \tag{2.2}$$



(a) (b) (c)

Figure 1: Initial values of the **Lax problem**: (a) density; (b) momentum; (c) energy.

where ρ, u and p are density, velocity and pressure respectively. The total energy E is

$$E=\frac{p}{\gamma-1}+\frac{1}{2}\rho v^2. \tag{2.3}$$

The ratio of specific heats γ is set at 1.4 throughout all cases.

The present work considers three cases: the Lax, Sod, and Woodward-Collela problems. The Lax and Sod problems are classical Riemann problems for the 1D Euler equation with initial conditions consisting of two constant states separated by a jump discontinuity. The Woodward-Colella problem is a challenging problem for testing numerical methods on 1D hyperbolic conservation laws. Two strong shock waves are initiated at the opposite ends of a uniform medium in a shock tube with closed ends, and evolve into multiple interactions of strong shocks, rarefaction waves and contact discontinuities. NNLCI is able to capture, accurately and robustly, solutions of these problems.

The Lax problem has the following initial condition, as shown in Fig. 1:

$$\begin{aligned}U(0,x)&=(0.445,0.311,8.928)^T, \quad 0<x\leq 0.5,\\ U(0,x)&=(0.5,0.0,1.4275)^T, \quad 0.5<x\leq 1.0,\end{aligned}$$

where $U(t,x)=(\rho,\rho v,E)^T$.

The Sod problem has the following initial condition, as shown in Fig. 2:

$$\begin{aligned}U(0,x)&=(1.0,0.0,2.5)^T, \quad 0<x\leq 0.5,\\ U(0,x)&=(0.125,0.0,0.25)^T, \quad 0.5<x\leq 1.0,\end{aligned}$$

where $U(t,x)=(\rho,\rho v,E)^T$.

The Woodward-Collela problem is an interacting blast wave problem for the Euler

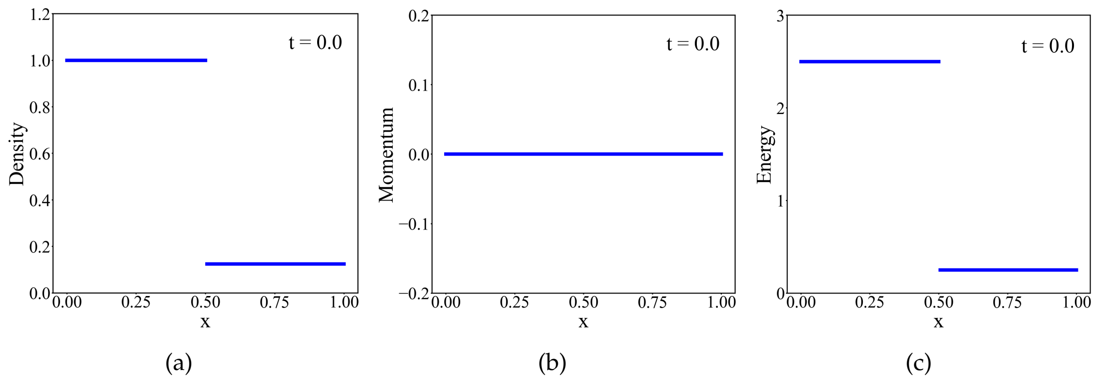


Figure 2: Initial values of the **Sod problem**: (a) density; (b) momentum; (c) energy.

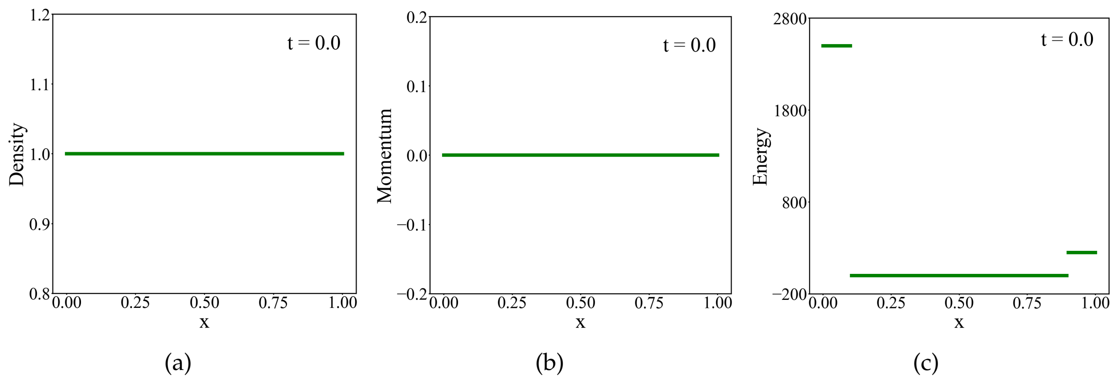


Figure 3: Initial values of the **Woodward Colella problem**: (a) density; (b) momentum; (c) energy.

equations with the following initial condition, as shown in Fig. 3:

$$
\begin{aligned}
&U(0,x) = (1.0,0.0,2500)^T, \quad 0 < x \le 0.1,\\
&U(0,x) = (1.0,0.0,0.025)^T, \quad 0.1 < x \le 0.9,\\
&U(0,x) = (1.0,0.0,250)^T, \quad 0.9 < x \le 1.0,
\end{aligned}
$$

where $U(t,x) = (\rho,\rho v,E)^T$.

The initial values of the training and validation cases for those problems are set by perturbing the magnitudes of the original initial values.

3 Neural network with local converging inputs (NNLCI)

It is an efficient strategy to use low-cost numerical solutions as the input to a neural network for predicting high-fidelity solutions. In [22], a numerical solution on a coarse grid was used in a neural network to predict the solution on a fine grid for a 2D heat-conduction equation. In [30], the gradient of the numerical solution on a coarse grid was employed to solve a second-order wave equation on a fine grid using neural networks. In the present study, we introduce a novel neural network method using local converging inputs, known as the NNLCI method, to accurately compute solutions of systems of conservation laws that may contain shock and contact discontinuities. The method proceeds as follows. First, two approximate solutions in a converging sequence are obtained by means of a low cost numerical scheme with coarse resolution. The resultant solutions are then used as inputs to a neural network, which, once trained, is able to provide an accurate solution for the entire domain, including regions with abrupt changes of state, such as shock waves and contact discontinuities.

3.1 Input, output and loss function

Consider the conservation laws in Eq. (2.1) for a spatial domain $[a,b]$, which is partitioned with a uniform coarse grid $a=x_0<x_1<\cdots<x_M=b$ having spatial grid size $\Delta x=x_1-x_0$ and time step size Δt, with Δt satisfying the CFL criterion. Refine the grid to obtain a finer uniform grid with spatial grid size $\frac{1}{2}\Delta x$ and time step size $\frac{1}{2}\Delta t$. Let L be a low-cost scheme for solving Eq. (2.1) on both the coarse and finer grids. For a given space-time location $(x_{i'},t_{n'})$ on the coarse grid where the solution is to be predicted by DNN, we choose the solution at $x_{i'-1}$, $x_{i'}$ and $x_{i'+1}$ at time level $t_{n'-1}$ and the solution at $(x_{i'},t_{n'})$ as the first part of the input. The refined-grid solution (also computed by L) at the same space-time locations is then used as the second part of the input. The four space-time locations enclose a space-time domain of dependence of the exact solution at $(x_{i'},t_{n'})$. The two inputs have different levels of approximation to the exact or reference solution. Through extrapolation DNN can utilize this information to achieve an accurate prediction.

Denote the first part of the input as

$$u_{i'-1}^{n'-1},u_{i'}^{n'-1},u_{i'+1}^{n'-1},u_{i'}^{n'}, \tag{3.1}$$

and the second part of input as

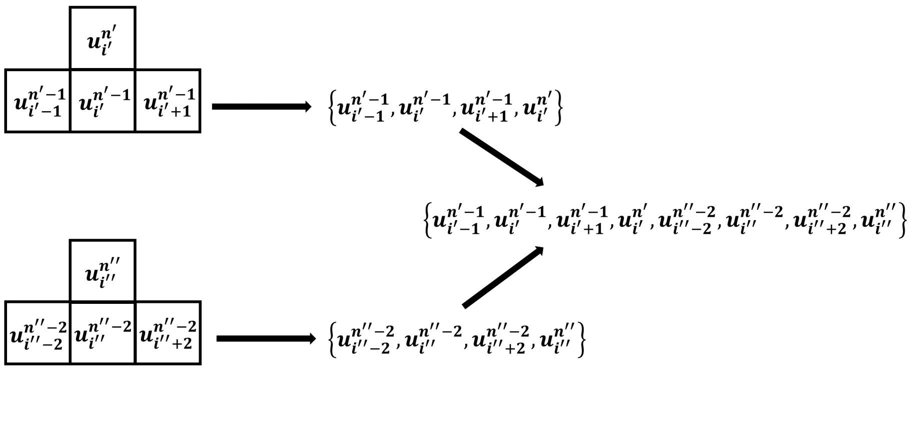
$$u_{i''-2}^{n''-2},u_{i''}^{n''-2},u_{i''+2}^{n''-2},u_{i''}^{n''}. \tag{3.2}$$

The space-time index (i',n') in the coarse grid refers to the same location as (i'',n'') does in the finer grid, $(i'-1,n'-1)$ refers to the same location as $(i''-2,n''-2)$ does, and so on.

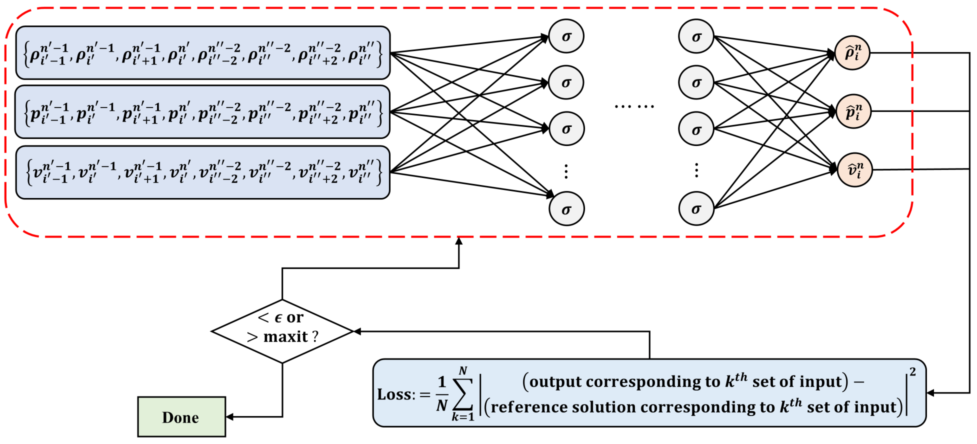
Suppose we are interested in the predicted solution at (x,t) which is referred to as (i',n') in the coarse grid, the input for the NNLCI

$$\{u_{i'-1}^{n'-1},u_{i'}^{n'-1},u_{i'+1}^{n'-1},u_{i'}^{n'},u_{i''-2}^{n''-2},u_{i''}^{n''-2},u_{i''+2}^{n''-2},u_{i''}^{n''}\}, \tag{3.3}$$

is called "input of u", and the corresponding output of NNLCI is the predicted solution at (x,t). Fig. 4 shows schematically the structure of the NNLCI method. For the Euler



(a)



(b)

Figure 4: Schematic of neural network with local converging inputs (NNLCI): (a) Procedure for transferring two-grid solutions into input format; (b) Training procedure for NNLCI.

system, the input and output of NNLCI are made up of the corresponding inputs and outputs for each prime variable. The input can be written in the vector form

$$\{\text{input of } \rho,\ \text{input of } v,\ \text{input of } p\} \tag{3.4}$$

with $8\times 3=24$ elements. The corresponding output has three elements as follows

$$\{\rho,v,p\}. \tag{3.5}$$

The loss function measures the difference between the output and the reference solution corresponding to the input, defined as follows.

$$\begin{aligned}\text{Loss}=\frac{1}{N}\sum_{k=1}^{N}|&(\text{output corresponding to } k^{th} \text{ set of input})\\&-(\text{reference solution corresponding to } k^{th} \text{ set of input})|^2,\end{aligned}\tag{3.6}$$

where $|\cdot|$ is the 2-norm measuring the distance between the output vector and the reference solution vector at the same space-time location. The summation includes every set of input (corresponding to different space-time location or initial condition) in the training data. Note that in the summation every output and its corresponding reference solution must be at the same space-time location. However, there may be terms (output and corresponding reference solution) in the summation at different time levels because the neural network only makes a local prediction which we can take advantage of. Suppose we want to predict the solution at the final time. The summation can include input (3.4) at the final or intermediate times for multiple initial conditions used in training. The latter is more costly but yields similar predictions in our numerical tests.

3.2 Generation of input and training data

In the present study, the associated DNN for NNLCI consists of 8 hidden layers and each layer has 300 neurons for the Lax or Sod problem, with the activation function *tanh*. The optimal structure of the neural network applied to all cases in the training procedure is achieved through multiple data experiments. During the training process, DNN minimizes the difference between the output from the neural network and the reference solution by means of an Adam optimizer and an L-BFGS optimizer in TensorFLow [1] (each with the number of iterations for optimization under 50000). After training, the neural network is used to predict high-fidelity solutions with the input computed by low cost scheme(s) on coarse grids. In the fully connected neural network, we explored different numbers of layers and neurons per layer to achieve the optimal setup. The prediction results are not sensitive to network structure, thereby allowing us to adjust the structure with at least 10% increment each time. In the case of excessive neurons, the minimization process could easily end up at a wrong local minimum. On the other hand, if the number of neurons is too small, the prediction error would be large because the neural network is not able to approximate the solution well. We will explore other types of neural networks in the future, such as the convolutional neural network, to reduce the number of parameters in a neural network. The prediction results by NNLCI, however, is not expected to render much difference when using different types of neural networks.

We use a first-order scheme on both a coarse (50 cells) and a finer (100 cells) uniform grid to generate the input data for several different initial values of the Euler system, including $\pm 2\%$, $\pm 4\%$, $\pm 6\%$, $\pm 8\%$ and $\pm 10\%$ perturbations of the initial value of the Lax or Sod problem. The high-resolution training data are computed on a uniform grid with 200 cells by means of a third-order finite-volume scheme using non-oscillatory hierarchical reconstruction (HR) limiting [25] and partial neighboring cells [52]. This scheme, which will be referred to as a third-order finite-volume scheme, is also used to compute reference solutions for all the validation cases in the present study. Three different first-order schemes for generating the inputs to the neural networks are considered, including the leapfrog with artificial diffusion scheme (3.7), the leapfrog and diffusion splitting scheme (3.8), and the Rusanov scheme (3.9). The time step size for the first order schemes is fixed to a constant, e.g., Δt for the 50-cell grid and $\frac{1}{2}\Delta t$ for the 100-cell grid, bounded by the CFL restriction with the largest characteristic speed estimated for the computational domain. The time duration of computation is $T=0.16$. The use of different low cost schemes in NNLCI helps demonstrate the robustness of the method. Prediction results based on schemes (3.7) and (3.8) are shown in Appendix A and B respectively.

Leapfrog scheme with artificial diffusion

$$\frac{U_i^{n+1}-U_i^{n-1}}{2\Delta t}+\frac{f(U)|_{i+1}^n-f(U)|_{i-1}^n}{2\Delta x}-\alpha\frac{U_{i+1}^{n-1}-2\cdot U_i^{n-1}+U_{i-1}^{n-1}}{\Delta x^2}=0, \tag{3.7}$$

where $\alpha=\Delta x$.

Leapfrog scheme with artificial diffusion splitting

$$\begin{cases} \frac{\tilde{U}_i-U_i^{n-1}}{2\Delta t}+\frac{f(U)|_{i+1}^n-f(U)|_{i-1}^n}{2\Delta x}=0, \\ \frac{U_i^{n+1}-\tilde{U}_i}{\Delta t}-\alpha\frac{\tilde{U}_{i+1}-2\cdot\tilde{U}_i+\tilde{U}_{i-1}}{\Delta x^2}=0, \end{cases} \tag{3.8}$$

where $\alpha=\Delta x$.

Rusanov scheme

$$\frac{U_i^{n+1}-U_i^n}{\Delta t}+\frac{\widehat{f(U)}|_{i+\frac{1}{2}}^n-\widehat{f(U)}|_{i-\frac{1}{2}}^n}{\Delta x}=0, \tag{3.9}$$

where

$$\widehat{f(U)}|_{i+\frac{1}{2}}^n=\frac{1}{2}\{f(U_{i+1}^n)+f(U_i^n)\}+\frac{\alpha}{2}\{U_i^n-U_{i+1}^n\}, \tag{3.10}$$

α is the largest characteristic speed defined by

$$\alpha=\max_i\{|v_i|+c_i\}, \tag{3.11}$$

and $c_i=\sqrt{\gamma\frac{p_i}{\rho_i}}$ is the speed of sound.

3.3 Results for the Lax problem

The high-fidelity simulation results of the original Lax problem and all the perturbation cases are used for training the neural network. The low-cost Rusanov scheme is employed to compute the inputs for NNLCI. We utilize input patches at all time levels and the high-fidelity solution at corresponding space-time locations for training the case with 2% perturbation from the initial value. For the remaining cases (i.e., $-2\%, \pm 4\%, \pm 6\%, \pm 8\%$, and $\pm 10\%$ perturbations of the initial value), only input patches at the final time are used for training, as discussed in Section 3.2. Once the neural network is trained, it can efficiently predict solutions with any prescribed perturbation of the initial condition in less than one second. Fig. 5 shows the prediction of the result at $t=0.16$ for the Lax problem. The NNLCI method accurately captures the shock and contact discontinuities, and full agreement with the high-resolution reference solution is achieved. Figs. 6 and 7 show, respectively, the predictions with the initial value varied by $\pm 3\%$ from that of the original Lax problem. The predictions with $\pm 5\%$ perturbations of the initial value are shown in Figs. 8 and 9, respectively.

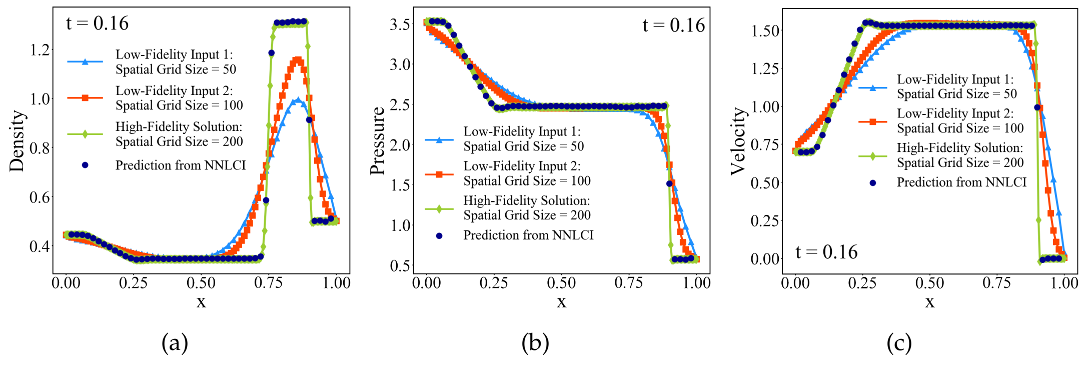


Figure 5: NNLCI prediction of the solution of the **Lax problem** at $t=0.16$.

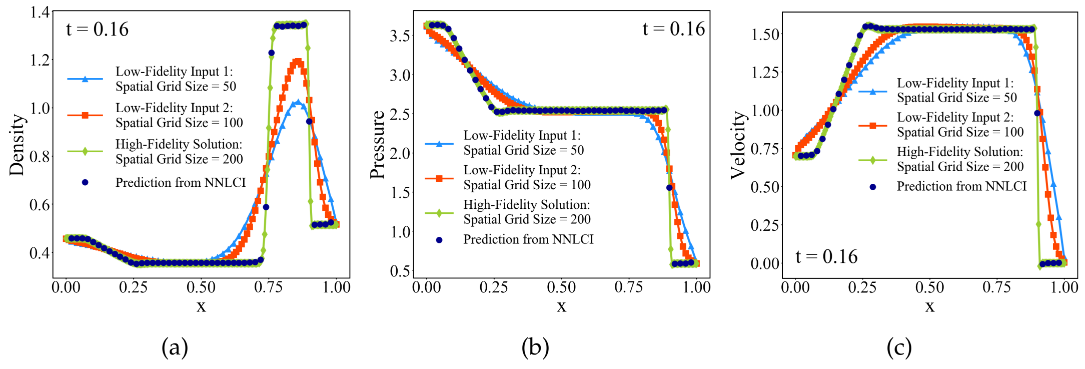


Figure 6: NNLCI prediction of the solution of the **Lax problem** at $t=0.16$; initial value perturbed by 3% from that of the original **Lax problem**.

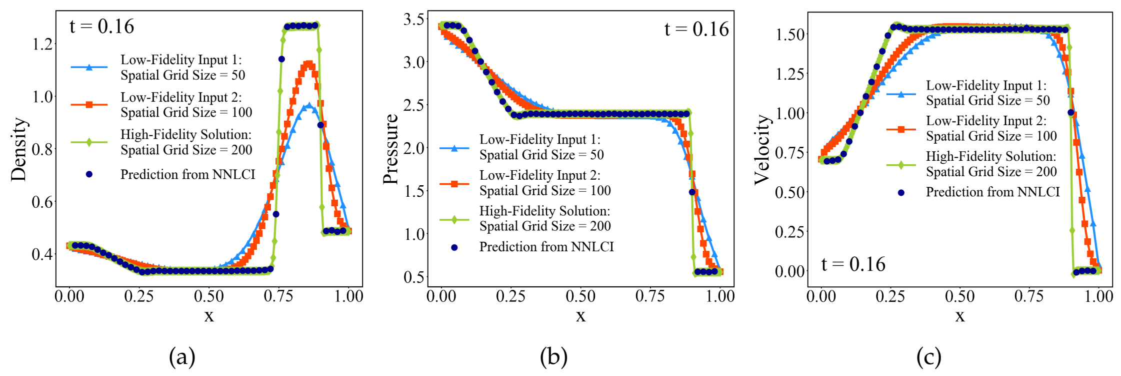


Figure 7: NNLCI prediction of the solution of the **Lax problem** at $t=0.16$; initial value perturbed by -3% from that of the original **Lax problem**.

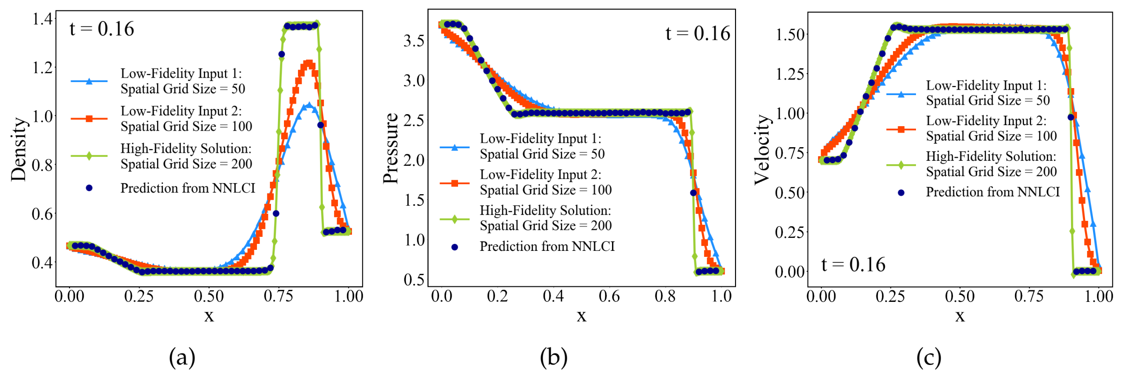


Figure 8: NNLCI prediction of the solution of the **Lax problem** at $t=0.16$; initial value perturbed by 5% from that of the original **Lax problem**.

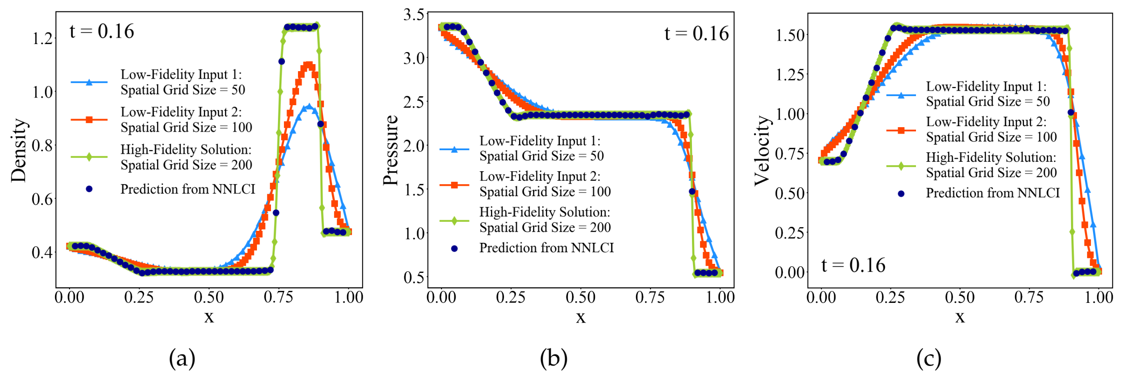


Figure 9: NNLCI prediction of the solution of the **Lax problem** at $t=0.16$; initial value perturbed by -5% from that of the original **Lax problem**.

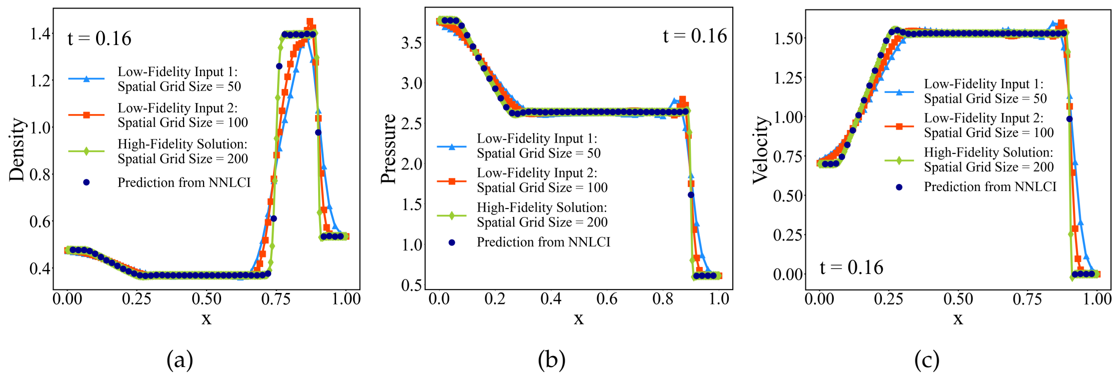


Figure 10: NNLCI prediction of the solution of the **Lax problem** at $t=0.16$; initial value perturbed by 7% from that of the original **Lax problem**.

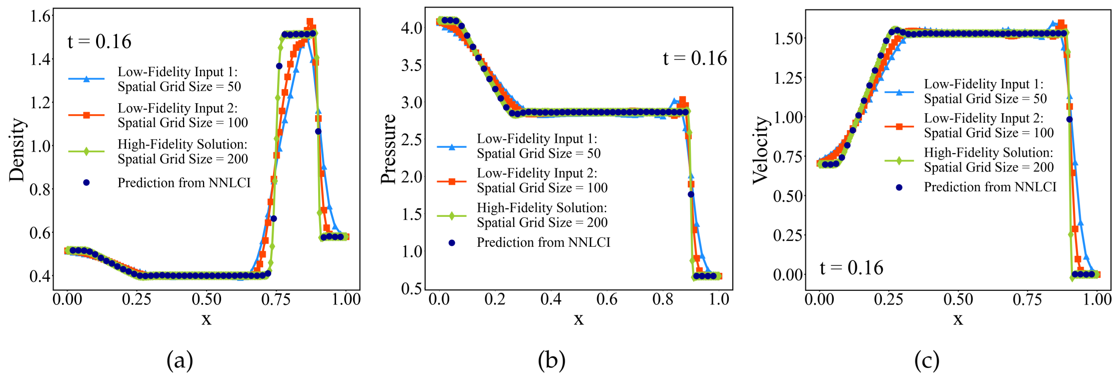
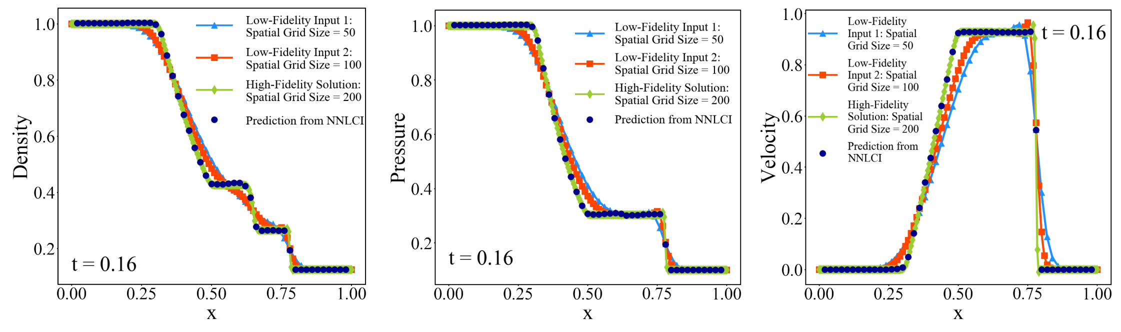


Figure 11: NNLCI prediction of the solution of the **Lax problem** at $t=0.16$; initial value perturbed by 16% from that of the original **Lax problem**.

To study the prediction accuracy when the training data are more sparse, we increase the spacing (defined as "relative distance" between two neighboring data points) in the training data from 2-4% to about 10% for the Lax problem. The training data now consist of solutions of the original Lax problem and cases with $\pm 10\%$ and $\pm 20\%$ perturbations from the initial value of the Lax problem. After training, the neural network is used to predict high-fidelity solutions for problems with other initial values. Figs. 10 and 11 show the prediction results for cases with 7% and 16% perturbed initial values. The inputs are computed by the leapfrog and diffusion-splitting scheme (3.8).

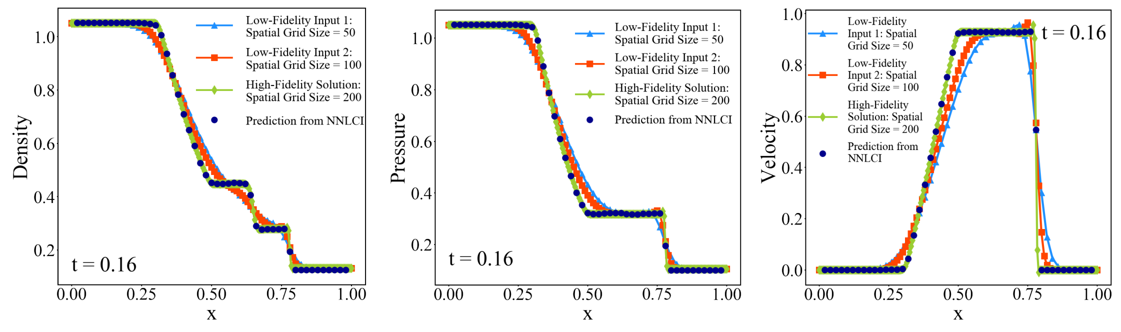
3.4 Results for the Sod problem

The training procedure for the Sod problem is identical to Lax problem. Fig. 12 shows the NNLCI prediction of the Sod problem at $t=0.16$. Full agreement between the NNLCI prediction and the high-fidelity reference solution is obtained throughout the entire field,



(a) (b) (c)

Figure 12: NNLCI prediction of the solution of the **Sod problem** at $t=0.16$.



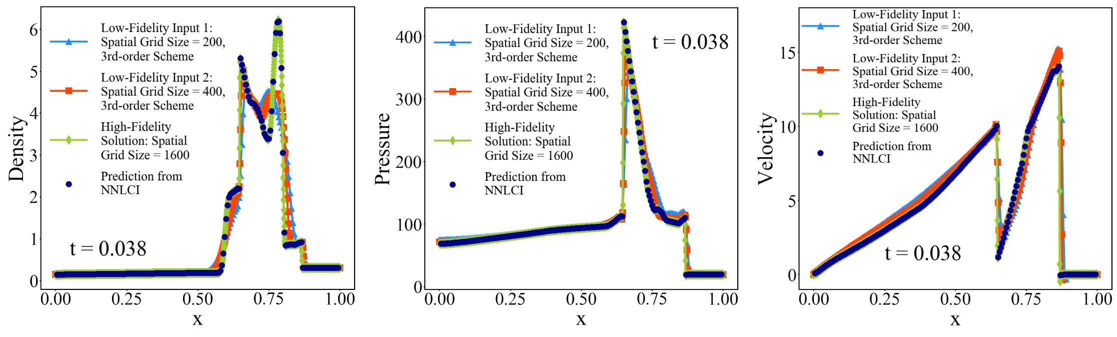
(a) (b) (c)

Figure 13: NNLCI prediction of the solution of the **Sod problem** at $t=0.16$; initial value perturbed by 5% from that of the original **Sod problem**.

including the shock and contact discontinuities. The first-order Rusanov Scheme was employed to compute the inputs to the NNLCI. The validity and effectiveness of the NNLCI technique was examined by considering $\pm3\%$ and $\pm5\%$ perturbations of the initial values of the original Sod problem. Fig. 13 shows a typical result of the flow evolution at $t=0.16$ for the case of 5% perturbation. Accurate prediction is achieved.

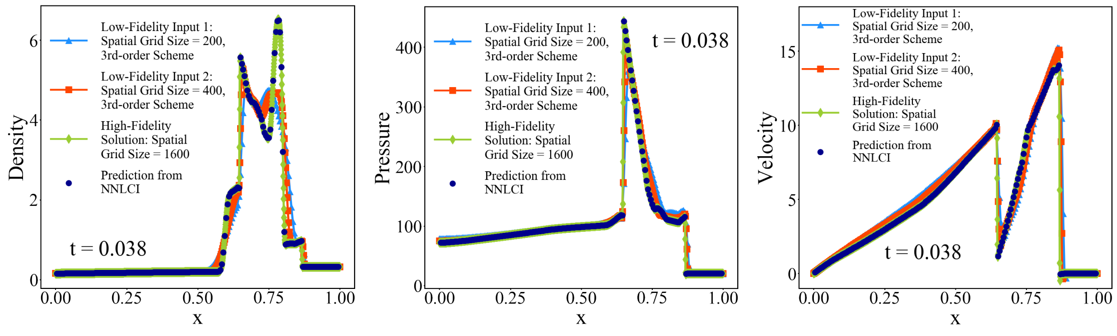
3.5 Results for the Woodward-Colella problem

For the Lax and Sod problems, the low-cost schemes for calculating the inputs are all first-order accurate. These schemes, however, are inadequate for the Woodward-Colella problem [49] because the inputs obtained from these schemes may considerably differ from the exact solution with reasonable grid sizes. A third-order finite-volume scheme is thus employed to compute the inputs with larger numbers of grid cells (200 and 400 grid cells in the present study). For the Woodward-Colella problem, a larger number of cells and time steps are needed than for the Lax or Sod problem. To reduce the computational



(a) (b) (c)

Figure 14: NNLCI prediction of the solution of the **Woodward-Colella problem** at $t=0.038$.



(a) (b) (c)

Figure 15: NNLCI prediction of the solution of the **Woodward-Colella problem** at $t=0.038$; initial value perturbed by 5% from that of the original **Woodward-Collela problem**.

cost, only low-cost numerical solutions at the final time are used for inputs in training. Figs. 14 and 15 show the excellent results of the interactive blast waves of the Woodward-Collela problem predicted by NNLCI. The associated DNN applied to the Woodward-Collela problem consists of 6 hidden layers and 180 neurons for each layer. The training of the neural network is similar to that for the Lax and Sod problems.

3.6 Other DNN inputs obtained from two different grids

There are other formats of the input that produce acceptable results. For example, one could use more data points adjacent to $x_{i'}$ for the local patch, obtained from the finer grid in the spatial domain. The input for a scalar conservation law can be changed from formula (3.3) to

$$\{u_{i'-1}^{n'-1}, u_{i'}^{n'-1}, u_{i'+1}^{n'-1}, u_{i'}^{n'}, u_{i''-2}^{n''-2}, u_{i''-1}^{n''-2}, u_{i''}^{n''-2}, u_{i''+1}^{n''-2}, u_{i''+2}^{n''-2}, u_{i''}^{n''}\}. \tag{3.12}$$

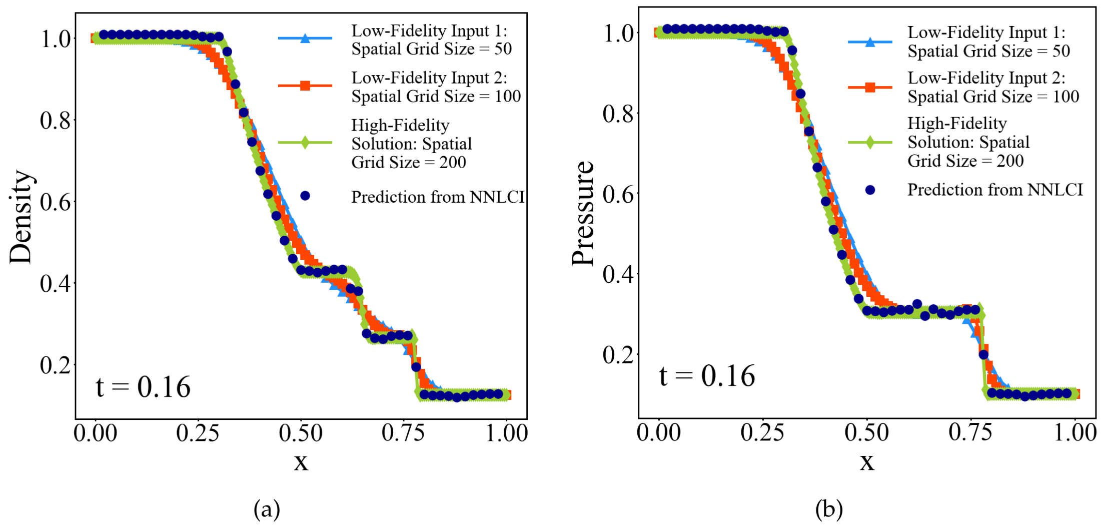


Figure 16: NNLCI prediction of the solution of the **Sod problem** at $t=0.16$ with input (3.13) obtained from the leapfrog and diffusion splitting scheme (3.8).

The same procedure applies for the Euler system. Predicted results from this type of input are comparable to (or slightly better than) those from input (3.3). Another variation of (3.3) is

$$\{u_{i'-1}^{n'-1}, u_{i'}^{n'-1}, u_{i'+1}^{n'-1}, u_{i'}^{n'}, u_{i''-1}^{n''-2}, u_{i''}^{n''-2}, u_{i''+1}^{n''-2}, u_{i''}^{n''}\}. \tag{3.13}$$

Predicted results from this type of input closely agree with (or are slightly worse than) those from input (3.3). Input (3.13) is computed by the leapfrog diffusion and splitting scheme (3.8). It generates some artifacts in the solution of the Sod problem, as shown in Fig. 16. Table 1 summarizes the results of all the cases tested for the baseline NNLCI method described in Section 3.

For a non-standard input (3.12) to the neural network, 9 layers and 300 neurons per layer are used to predict the solution of the Sod problem, while the numbers of neurons and layers remain unchanged for other cases. Table 2 presents the predicted results based on input (3.12) computed by the leapfrog and diffusion splitting scheme.

4 DNN input obtained from perturbed governing equation on single grid

Instead of computing the input to DNN from two different grids, the input can be generated by adding dissipation to the governing equation on a single grid, as described below. The perturbed equation takes the form,

$$\frac{\partial U}{\partial t}+\frac{\partial f(U)}{\partial x}=\alpha\frac{\partial^2 U}{\partial x^2}, \quad x\in\Omega\subset\mathcal{R}, \quad t\in[0,T]. \tag{4.1}$$

Table 1: Relative l_2 errors of NNLCI predictions of solutions of the Euler system (with initial values perturbed from those of the Lax, Sod or Woodward-Colella problems). The l_2 errors for the Lax and Sod problems are over the entire space-time computational domain, while for the Woodward-Colella problem, they measure the accuracy of the prediction at the final time. The low-fidelity input solutions are computed by 1st order schemes (the Rusanov scheme, stabilized leapfrog and diffusion scheme, and leapfrog and diffusion splitting scheme) on 2 different grids (50 and 100 cells) or by 3rd-order finite volume scheme on 2 different grids (200 and 400 cells).

	Rusanov scheme		**Stabilized leapfrog and diffusion scheme**	
	Lax Problem	**Sod Problem**	**Lax Problem**	**Sod Problem**
Initial Value	Relative l_2 Error	Relative l_2 Error	Relative l_2 Error	Relative l_2 Error
Original	5.97E−03	6.20E−03	8.73E−03	6.22E−03
+3%	7.29E−03	5.66E−03	5.47E−03	7.68E−03
−3%	9.10E−03	9.05E−03	1.14E−02	1.13E−02
+5%	8.61E−03	7.84E−03	1.01E−02	6.47E−03
−5%	1.32E−02	1.29E−02	1.29E−02	1.11E−02
+7%	9.96E−03	9.72E−03	1.42E−02	8.37E−03
−7%	1.75E−02	1.45E−02	1.53E−02	1.77E−02

	Leapfrog and diffusion splitting scheme		**3rd-order finite volume scheme**
	Lax Problem	**Sod Problem**	**Woodward-Colella Problem**
Initial Value	Relative l_2 Error	Relative l_2 Error	Relative l_2 Error
Original	8.56E−03	5.72E−03	1.29E−04
+3%	4.80E−03	8.03E−03	8.22E−05
−3%	6.86E−03	8.26E−03	8.38E−05
+5%	4.96E−03	6.60E−03	9.43E−05
−5%	7.05E−03	1.84E−02	8.63E−05
+7%	6.22E−03	8.55E−03	1.23E−04
−7%	8.28E−03	1.05E−02	9.10E−05

Table 2: Relative l_2 errors of NNLCI (with non-standard input (3.12)) predictions of solutions of the Euler system (with initial values perturbed from those of the Lax or Sod problems). The l_2 errors for the Lax and Sod problems are calculated over the entire space-time computational domain. The low-fidelity input solutions are computed by the leapfrog and diffusion splitting scheme on two different grids (50 and 100 cells).

	Lax Problem	**Sod Problem**
Initial Value	Relative l_2 Error	Relative l_2 Error
Original	8.17E−03	8.35E−03
+3%	4.06E−03	2.81E−02
−3%	5.96E−03	6.63E−03
+5%	5.27E−03	9.37E−03
−5%	1.06E−02	9.47E−03
+7%	8.51E−03	8.27E−03
−7%	1.49E−02	2.46E−02

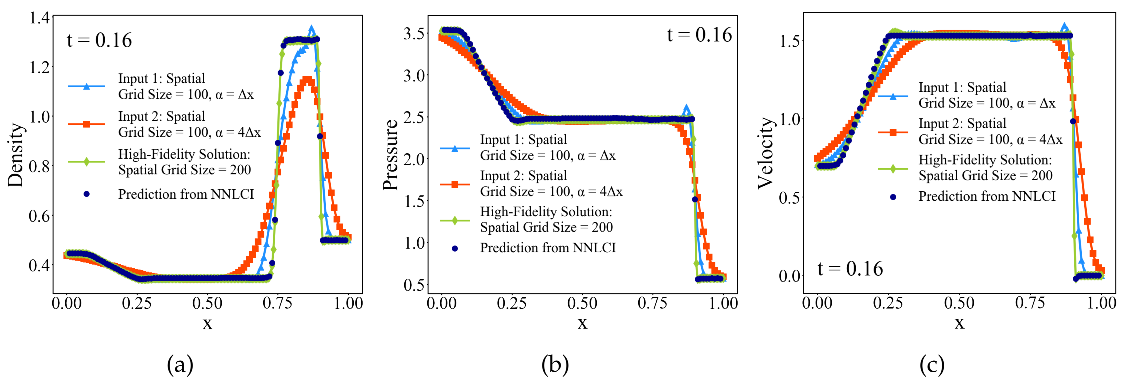


Figure 17: NNLCI prediction of the solution of the **Lax problem** at $t=0.16$; input obtained from a perturbed governing equation on the same grid.

Two different values of the diffusion coefficient α are employed to compute the input in a converging sequence by means of a low cost scheme on a single grid. For example, Eq. (4.1) can be solved using the leapfrog and diffusion splitting scheme with $\alpha=\Delta x$ and $c\Delta x$ as follows

$$\left\{\begin{aligned} &\frac{\tilde{U}_i-U_i^{n-1}}{2\Delta t}+\frac{f(U)|_{i+1}^n-f(U)|_{i-1}^n}{2\Delta x}=0,\\ &\frac{U_i^{n+1}-\tilde{U}_i}{\Delta t}-\Delta x\frac{\tilde{U}_{i+1}-2\cdot\tilde{U}_i+\tilde{U}_{i-1}}{\Delta x^2}=0,\end{aligned}\right. \tag{4.2}$$

and

$$\left\{\begin{aligned} &\frac{\tilde{V}_i-V_i^{n-1}}{2\Delta t}+\frac{f(V)|_{i+1}^n-f(V)|_{i-1}^n}{2\Delta x}=0,\\ &\frac{V_i^{n+1}-\tilde{V}_i}{\Delta t}-c\Delta x\frac{\tilde{V}_{i+1}-2\cdot\tilde{V}_i+\tilde{V}_{i-1}}{\Delta x^2}=0.\end{aligned}\right. \tag{4.3}$$

The input to DNN can be written as

$$\{U_{i-1}^{n-1},U_i^{n-1},U_{i+1}^{n-1},U_i^n,V_{i-1}^{n-1},V_i^{n-1},V_{i+1}^{n-1},V_i^n\}. \tag{4.4}$$

Figs. 17 and 18 show the predicted results by NNLCI. The input was computed on a uniform grid with 100 cells and $c=4$. Note that in the second step of Eq. (4.3), the larger diffusion coefficient may require a time step size smaller than that of Eq. (4.2). It may be necessary to compute the second step of Eq. (4.3) in two sub time steps, with $\frac{\Delta t}{2}$ for each sub time step. For example, for the Lax problem with a cell size of $\Delta x=0.01$ (100 cells), the numerical simulation was carried out for 358 time steps with a uniform time step size of Δt=4.47×10^{-4}. And the sub time step size for the second step of (19) is $\frac{\Delta t}{2}$=2.24×10^{-4}.

The low cost, first order leapfrog and diffusion splitting scheme functions well in providing inputs to NNLCI for the Lax and Sod problems. The first order scheme, however, does not work properly for the Woodward-Colella problem, as mentioned in Section 3.5. Here, therefore, the first order Rusanov scheme and the third order finite-volume scheme previously used for reference solutions are employed to compute inputs on a coarse grid.

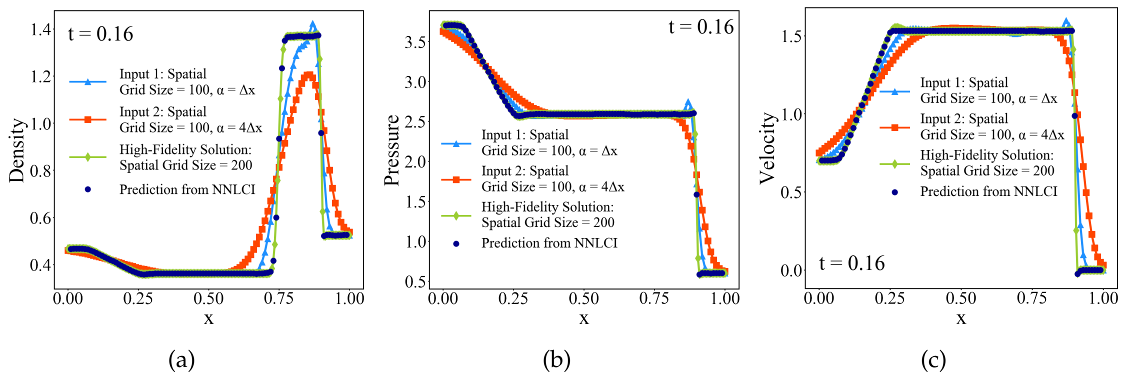


Figure 18: NNLCI prediction of the solution of the **Lax problem** at $t=0.16$, with the initial value perturbed by 5% from the original value; input obtained from a perturbed governing equation on the same grid.

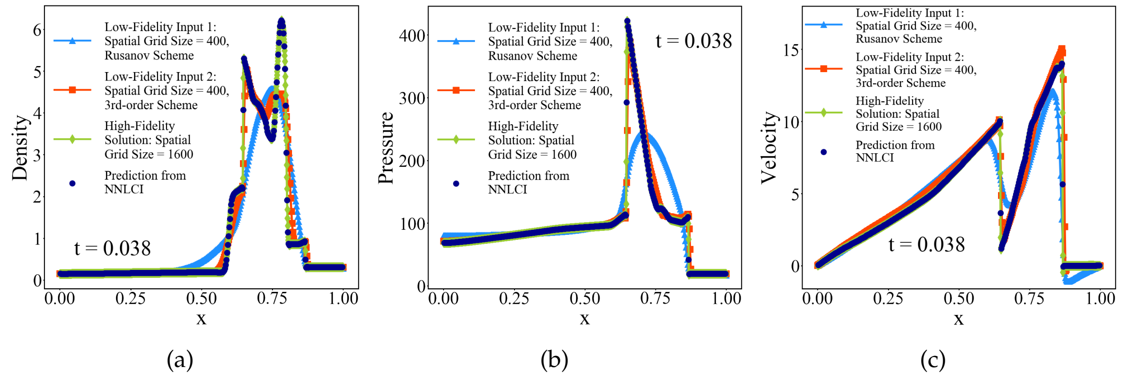


Figure 19: NNLCI prediction of the solution of the **Woodward-Colella problem** at $t=0.038$; input obtained from two different numerical schemes on the same grid.

This process can be viewed as a variant of the baseline NNLCI because only one grid is used in computing inputs. The input format is similar to (4.4), with the first part of the input computed by the Rusanov scheme [36] and the second part of the input by a higher order scheme at the same space-time locations. The structure of the DNN in this case for the training procedure consists of 6 hidden layers with 180 neurons per layer. Figs. 19 and 20 show the interactive blast waves of the Woodward-Colella problem predicted by NNLCI. The predicted results agree well with the reference solutions.

5 NNLCI with variable time step size Δt

5.1 Cross-training and inclusion of time step in input

In Sections 3 and 4 the time step of the low-cost scheme is fixed at Δt. To accommodate concurrent training of multiple cases with different time steps, the time step Δt for each

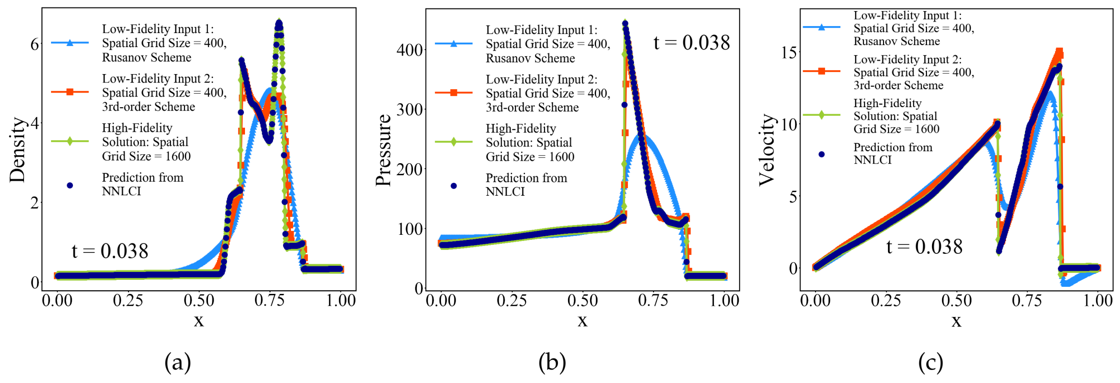


Figure 20: NNLCI prediction of the solution of the **Woodward-Colella problem** at $t=0.038$, with the initial value perturbed by 5% from the original value; input obtained from two different numerical schemes on the same grid.

Table 3: Relative l_2 errors of variants of NNLCI predictions of solutions of the Euler system (with initial values perturbed from those of the Lax, Sod, or Woodward-Colella problem). The l_2 errors for the Lax and Sod problems are over the entire space-time computational domain, while for the Woodward-Colella problem, they measure the accuracy of the prediction at the final time. Low-fidelity input solutions for the Lax or Sod problem are computed by the Leapfrog and diffusion splitting scheme (3.8) with 2 different diffusion coefficients ($\alpha=\Delta x$ and $4\Delta x$) on a uniform grid with 100 cells; low-fidelity input solutions for the Woodward-Colella problem are computed by the Rusanov scheme and a 3rd order finite volume scheme on a uniform grid with 400 cells.

	Leapfrog and diffusion splitting scheme		**3rd-order finite volume scheme**
	Lax Problem	**Sod Problem**	**Woodward Colella Problem**
Initial Value	Relative l_2 Error	Relative l_2 Error	Relative l_2 Error
Original	$5.24\text{E}-03$	$6.47\text{E}-03$	$2.56\text{E}-04$
$+3\%$	$3.43\text{E}-03$	$6.80\text{E}-03$	$1.41\text{E}-04$
-3%	$5.34\text{E}-03$	$8.57\text{E}-03$	$1.45\text{E}-04$
$+5\%$	$7.13\text{E}-03$	$7.52\text{E}-03$	$1.33\text{E}-04$
-5%	$6.59\text{E}-03$	$1.07\text{E}-02$	$1.44\text{E}-04$
$+7\%$	$8.19\text{E}-03$	$8.87\text{E}-03$	$1.32\text{E}-04$
-7%	$7.51\text{E}-03$	$1.30\text{E}-02$	$1.44\text{E}-04$

case is used as part of the input, as follows

$$\{u_{i'-1}^{n'-1}, u_{i'}^{n'-1}, u_{i'+1}^{n'-1}, u_{i'}^{n'}, u_{i''-2}^{n''-2}, u_{i''}^{n''-2}, u_{i''+2}^{n''-2}, u_{i''}^{n''}, \Delta t\}. \tag{5.1}$$

The output remains unchanged. Similarly, the input for the Euler system (2.2) can include Δt with the output unchanged. The associated neural network typically consists of 6 hidden layers, and each layer has 180 neurons. When performing "cross-training" (that is, the training data covers several different problems, such as both the Lax and Sod problems [39]), the neural network may be up to 6 hidden layers with 255 neurons per layer, or even 300 neurons per layer. When Δt is chosen to be the same in the Lax and Sod

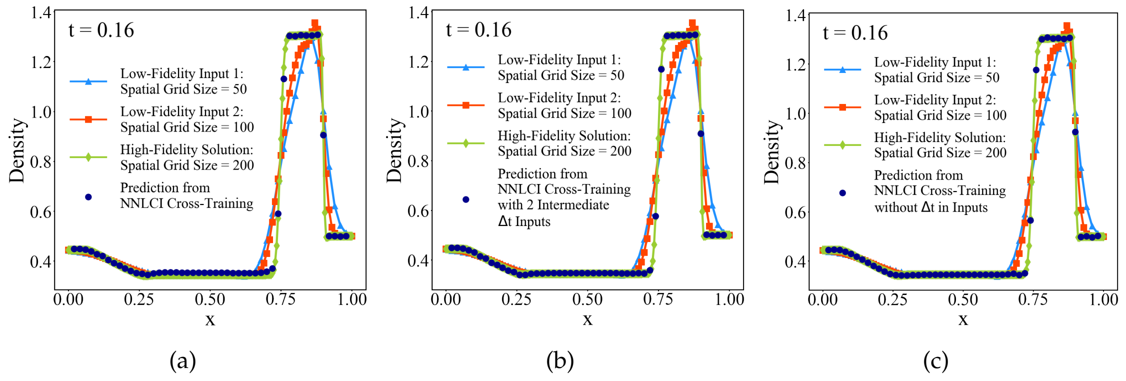


Figure 21: Cross-training predictions of density solution of the **Lax problem** at $t=0.16$.

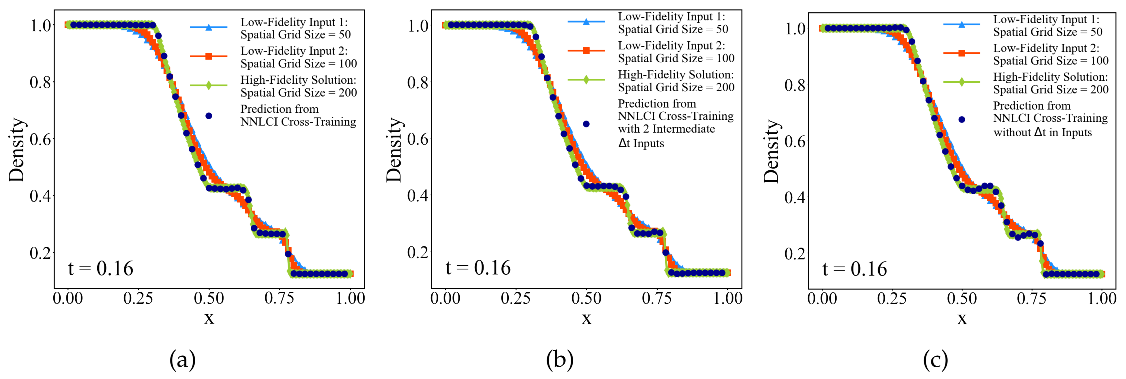


Figure 22: Cross-training predictions of density solution of the **Sod problem** at $t=0.16$.

problems, there is no need to treat Δt as part of the input. In the present work, we only use training data in the final time of the calculation, thereby reducing the computational cost. The neural network for predicting the final-time solutions of the Lax and Sod problems consists of 5 hidden layers with 66 neurons per layer.

In order to generate the inputs to NNLCI, we use a first-order scheme on a coarse (50 cells) and a finer (100 cells) uniform grid to compute input data for the Euler system with several different initial values, including $\pm 2\%$, $\pm 4\%$, $\pm 6\%$, $\pm 8\%$ and $\pm 10\%$ perturbations of the baseline condition. The high-resolution reference solutions of the training data are computed on a uniform grid with 200 cells by a third order finite volume scheme with non-oscillatory hierarchical reconstruction (HR) limiting [25] and partial neighboring cells [52] in HR. The first order scheme used for generating the inputs is the same as that described in Sections 3 and 4. The time step is fixed at Δt for the 50-cell grid and $\frac{\Delta t}{2}$ for the 100-cell grid, satisfying the CFL condition.

Figs. 21 and 22 present the predicted density profiles of the Lax and Sod problems using three different types of training approach. The first strategy employs training data

Table 4: Comparison of cross-training in Section 5.1 with 3 different types of training or inputs: Relative l_2 norm errors between the predicted and "exact" (reference) solutions for the Lax problem. The l_2 errors are at the final time for the case not including Δt in the input, while for the other two cases, they measure the accuracy over the entire space-time computational domain.

	Lax Problem		
	Δt in Input	Δt in Input and 2 Intermediate Δt values in training	Δt not in Input
Initial Value	Relative l_2 Errors	Relative l_2 Errors	Relative l_2 Errors
Original	9.36E−03	6.95E−03	1.22E−03
+3%	4.29E−03	3.65E−03	1.23E−03
−3%	8.89E−03	6.01E−03	1.49E−03
+5%	8.46E−03	4.69E−03	1.29E−03
−5%	9.35E−03	8.13E−03	1.40E−03
+7%	9.04E−03	7.01E−03	1.27E−03
−7%	1.11E−02	8.81E−03	1.21E−03

generated from the perturbed Lax and Sod problems described above. Note that the same spatial and temporal grid sizes must be used when computing the inputs for the two problems. The predicted results are shown in the right plots of Fig. 21 and 22 respectively. The second strategy uses the input type of (5.1) with the training data generated from the perturbed Lax and Sod problems. Since the time step is included as part of the input, we can use the same spatial grid size with different time steps when computing the inputs for the two problems. The different time steps help the neural network switch between the two problems, and the prediction results are improved, as shown in the left plots of Figs. 21 and 22 respectively. To increase the number of Δt in input during training, two evenly distributed (between those originally used for numerical simulations of the Lax and Sod problems) are used in computing the inputs. The extra training data corresponding to the two intermediate Δt are incorporated in the loss function. Results of the validation cases show that the training data associated with intermediate Δt in corresponding computed inputs improve the prediction accuracy for the Lax problem, but to a less extent for the Sod problem. The prediction results are shown in the middle plots of Figs. 21 and 22.

6 Conclusion

We have developed the NNLCI method for solving the one-dimensional Euler system whose solutions may contain abrupt changes of state, including shock waves and contact discontinuities.The method has been successfully applied to treat the Lax, Sod, and Woodward-Colella problems. The predicted results using NNLCI, even with smeared inputs, agree well with high-fidelity solutions. The NNLCI method is robust and efficient,

Table 5: Comparison of cross-training in Section 5.1 with 3 different types of training or inputs: Relative l_2 norm errors between the predicted and "exact" (reference) solutions for the Sod problem. The l_2 errors are at the final time for the case not including Δt in the input, while for the other two cases, they measure the accuracy over the entire space-time computational domain.

	Sod Problem		
	Δt in Input	Δt in Input and 2 Intermediate Δt values in training	Δt not in Input
Initial Value	Relative l_2 Errors	Relative l_2 Errors	Relative l_2 Errors
Original	3.31E−03	5.36E−03	6.45E−03
+3%	3.03E−03	4.93E−03	6.39E−03
−3%	4.63E−03	7.35E−03	6.54E−03
+5%	3.45E−03	6.43E−03	6.38E−03
−5%	5.59E−03	8.21E−03	6.63E−03
+7%	4.19E−03	6.59E−03	6.42E−03
−7%	6.62E−03	9.29E−03	6.75E−03

and is not sensitive to the low-cost schemes used for computing the inputs to neural network. (See Appendices A and B for additional prediction results, with alternative schemes for computing inputs.) In addition, an effective approach is to accommodate different time steps as part of the input, enabling the neural network to handle different time steps for various cases. A combination of a high-order and a low-order scheme can also be employed on a coarse grid to determine the input to NNLCI. Future work will include extension of NNLCI to multidimensional systems of conservation equations, and exploration of other ways to generate a converging sequence as input to a neural network, such as the use of a relaxation scheme [19] with two different relaxation parameters.

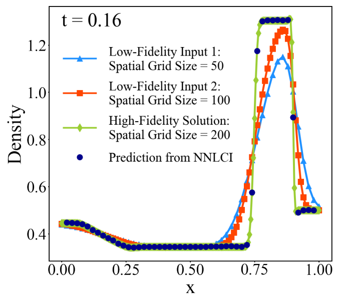
Acknowledgments

The research of Y. Liu is supported in part by NSF grant DMS-1522585. This work was partly sponsored by the Ralph N. Read Endowment of the Georgia Institute of Technology. The authors gratefully acknowledge fruitful conversation with Prof. Wen Shen of the Pennsylvania State University.

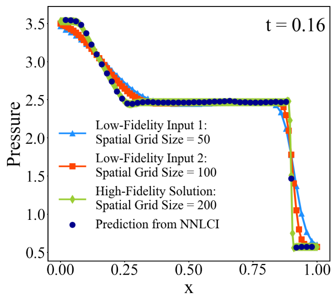
Appendices

A Stabilized leapfrog and diffusion scheme (3.7) for computing inputs for NNLCI

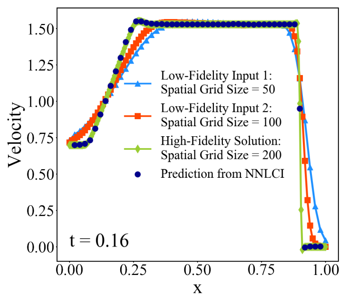
The NNLCI predictions of the final-time solutions of the perturbed Lax and Sod problems are presented in Appendices A and B, respectively, based on inputs computed by the stabilized leapfrog and diffusion scheme, and leapfrog and diffusion splitting scheme. Inputs computed by the two schemes are less diffusive than those by the Rusanov scheme. The prediction results are similar for the three first-order schemes, further demonstrating the robustness of NNLCI.



(a)

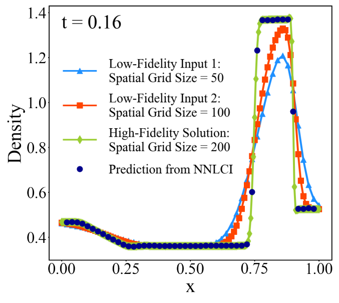


(b)

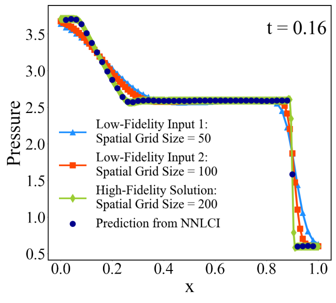


(c)

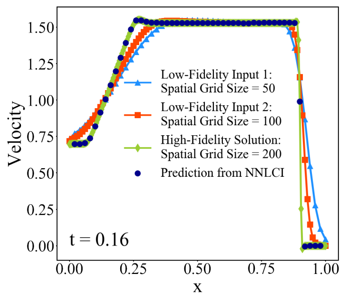
Figure 23: NNLCI (Section 3.1) prediction of solution of the **Lax problem** at final-time ($t=0.16$).



(a)

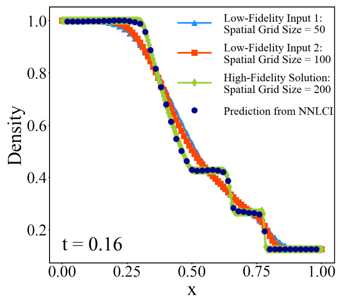


(b)

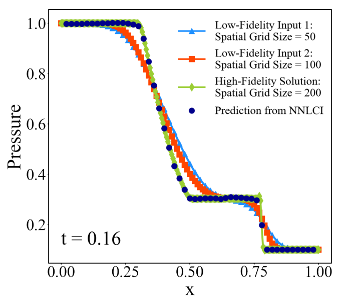


(c)

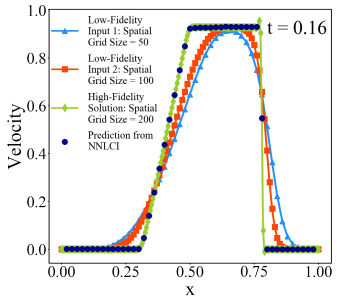
Figure 24: NNLCI (Section 3.1) prediction of solution of the **Lax problem** at final-time ($t=0.16$); initial value perturbed by 5% from that of the original **Lax problem**.



(a)



(b)



(c)

Figure 25: NNLCI (Section 3.1) prediction of solution of the **Sod problem** at final-time ($t=0.16$).

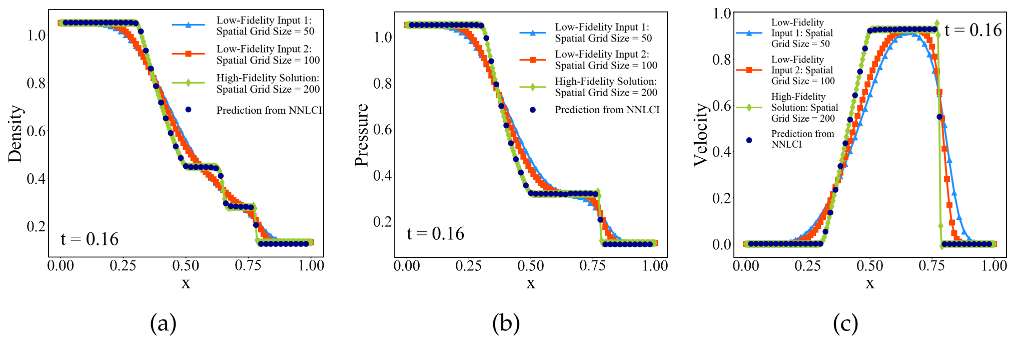


Figure 26: NNLCI (Section 3.1) prediction of solution of the **Sod problem** at final-time (t = 0.16); initial value perturbed by 5% from that of the original **Sod problem**.

B Leapfrog and diffusion splitting scheme (3.8) for computing inputs for NNLCI

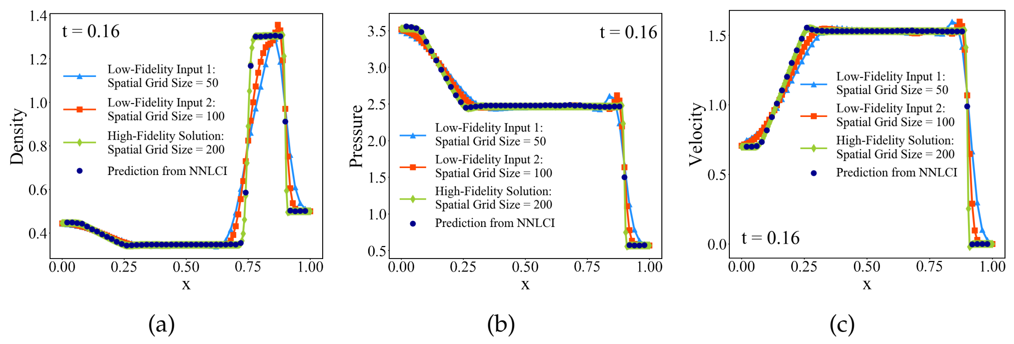


Figure 27: NNLCI (Section 3.1) prediction of solution of the **Lax problem** at final-time ($t=0.16$).

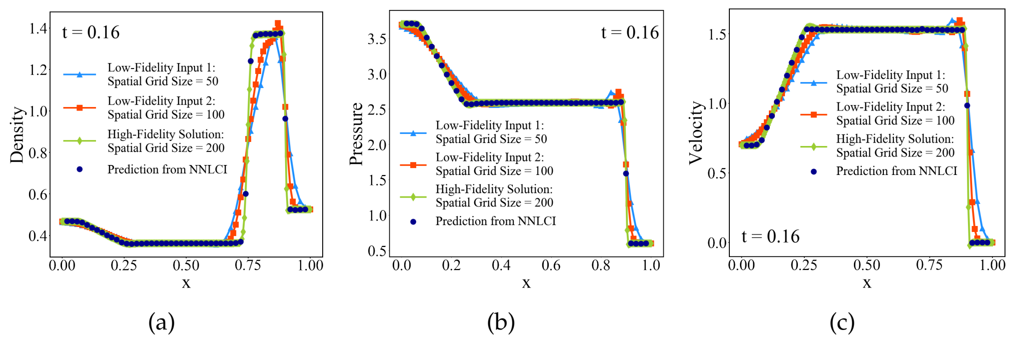
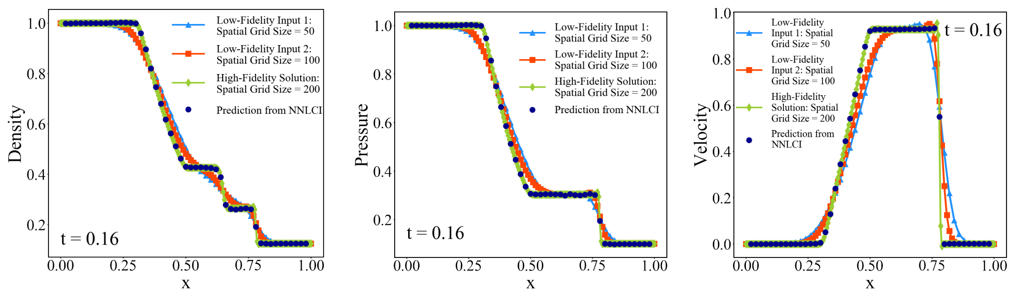
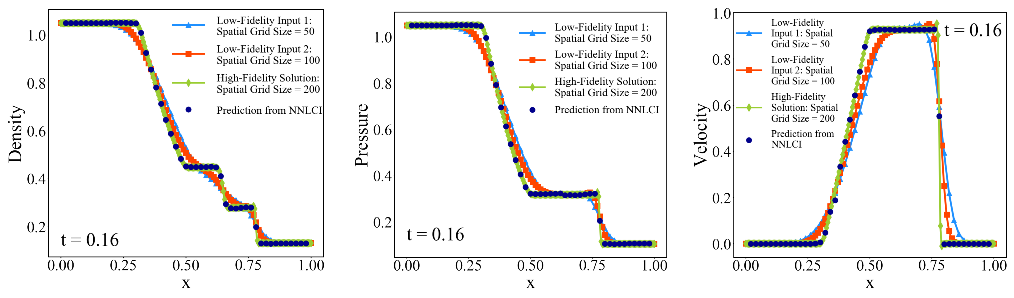


Figure 28: NNLCI (Section 3.1) prediction of solution of the **Lax problem** at final-time ($t=0.16$); initial value perturbed by 5% from that of the original **Lax problem**.



(a) (b) (c)

Figure 29: NNLCI (Section 3.1) prediction of solution of the **Sod problem** at final-time ($t=0.16$).



(a) (b) (c)

Figure 30: NNLCI (Section 3.1) prediction of solution of the **Sod problem** at final-time ($t=0.16$); initial value perturbed by 5% from that of the original **Sod problem**.

References

[1] Abadi, M., Agarwal, A., Barham, P., and et al. TensorFlow: Large-scale machine learning on heterogeneous distributed systems. *arXiv: 1603.04467* (2016).

[2] Abgrall, R., and Veiga, M. Neural network-based limiter with transfer learning. *Communications on Applied Mathematics and Computation* (2020), 1–41.

[3] Baydin, A. G., Pearlmutter, B. A., Radul, A. A., and Siskind, J. M. Automatic differentiation in machine learning: A survey. *Journal of Machine Learning Research 18*, 1 (2017), 5595–5637.

[4] Berger, M., and Colella, P. Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics 82*, 1 (1989), 64–84.

[5] Cai, S., Wang, Z., Wang, S., Perdikaris, P., and Karniadakis, G. Physics-informed neural networks for heat transfer problems. *Journal of Heat Transfer 143*, 6 (2021), 060801.

[6] Chang, Y., Wang, X., Zhang, L., and et.al. Common kernel-smoothed proper orthogonal decomposition (CKSPOD): An efficient reduced-order model for emulation of spatiotemporally evolving flow dynamics. *arXiv: 2101.08893* (2021).

[7] Chang, Y., Zhang, L., Wang, X., and et.al. Kernel-smoothed proper orthogonal decomposition–based emulation for spatiotemporally evolving flow dynamics prediction.

AIAA Journal 57, 12 (2019), 5269–5280.

[8] CHEN, L., PALEJA, R., AND GOMBOLAY, M. Learning from suboptimal demonstration via self-supervised reward regression. *arXiv: 2010.11723* (2020).

[9] CHEN, T., AND CHEN, H. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Transactions on Neural Networks 6*, 4 (1995), 911–917.

[10] DARCY, A., LOUIE, A., AND ROBERTS, L. Machine learning and the profession of medicine. *JAMA 315*, 6 (2016), 551–552.

[11] DISCACCIATI, N., HESTHAVEN, J., AND RAY, D. Controlling oscillations in high-order discontinuous Galerkin schemes using artificial viscosity tuned by neural networks. *Journal of Computational Physics 409* (2020), 109304.

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

[13] GENG, Z., JOHNSON, D., AND FEDKIW, R. Coercing machine learning to output physically accurate results. *Journal of Computational Physics 406* (2020), 109099.

[14] GODUNOV, S. K. A difference scheme for numerical solution of discontinuous solution of hydrodynamic equations. *Matematicheskii Sbornik 47* (1959), 271–306.

[15] HARTEN, A. ENO schemes with subcell resolution. *Journal of Computational Physics 83*, 1 (1989), 48–184.

[16] HARTEN, A., ENGQUIST, B., OSHER, S., AND CHAKRAVARTHY, S. R. Uniformly high order accuracy essentially non-oscillatory schemes III. *Journal of Computational Physics 71*, 2 (1987), 231–303.

[17] HSIEH, S. Y., AND YANG, V. A preconditioned flux-differencing scheme for chemically reacting flows at all Mach numbers. *International Journal of Computational Fluid Dynamics 8*, 1 (1997), 31–49.

[18] JIANG, G.-S., AND SHU, C.-W. Efficient implementation of weighted ENO schemes. *Journal of Computational Physics 126* (1996), 202–228.

[19] JIN, S., AND XIN, Z. The relaxation schemes for systems of conservation laws in arbitrary space dimensions. *Communications on Pure and Applied Mathematics 48*, 3 (2020), 235–276.

[20] KARNIADAKIS, G., KEVREKIDIS, I., LU, L., PERDIKARIS, P., WANG, S., AND YANG, L. Physics-informed machine learning. *Nature Reviews Physics 3* (2021), 422–440.

[21] KRIZHEVSKY, A., SUTSKEVER, I., AND HINTON, G. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems 25* (2012).

[22] LIU, D., AND WANG, Y. Multi-fidelity physics-constrained neural network and its application in materials modeling. *Journal of Mechanical Design 141*, 12 (2019), 121403.

[23] LIU, X. D., OSHER, S., AND CHAN, T. Weighted essentially non-oscillatory schemes. *Journal of Computational Physics 115*, 1 (1994), 200–212.

[24] LIU, Y., SHU, C.-W., TADMOR, E., AND ZHANG, M.-P. Central discontinuous Galerkin methods on overlapping cells with a non-oscillatory hierarchical reconstruction. *SIAM Journal on Numerical Analysis 45* (2007), 2442–2467.

[25] LIU, Y., SHU, C.-W., TADMOR, E., AND ZHANG, M.-P. Non-oscillatory hierarchical reconstruction for central and finite volume schemes. *Communications in Computational Physics 2* (2007), 933–963.

[26] LOU, Q., MENG, X., AND KARNIADAKIS, G. Physics-informed neural networks for solving forward and inverse flow problems via the Boltzmann-BGK formulation. *arXiv: 2010.09147* (2020).

[27] LU, L., JIN, P., AND KARNIADAKIS, G. DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv: 1910.03193* (2019).

[28] MAGIERA, J., RAY, D., HESTHAVEN, J., AND ROHDE, C. Constraint-aware neural networks for Riemann problems. *Journal of Computational Physics 409* (2020), 109345.

[29] MAK, S., SUNG, C., WANG, X., YEH, S., CHANG, Y., JOSEPH, V., YANG, V., AND WU, C. F. J. An efficient surrogate model for emulation and physics extraction of large eddy simulations. *Journal of the American Statistical Association 113*, 524 (2018), 1443–1456.

[30] NGUYEN, H., AND TSAI, R. Numerical wave propagation aided by deep learning. *arXiv: 2107.13184* (2021).

[31] PEHERSTORFER, B., WILLCOX, K., AND GUNZBURGER, M. Survey of multifidelity methods in uncertainty propagation, inference, and optimization. *SIAM Review 60*, 3 (2018), 550–591.

[32] PSAROS, A. F., KAWAGUCHI, K., AND KARNIADAKIS, G. Meta-learning pinn loss functions. *arXiv:2107.05544* (2021).

[33] RAISSI, M., PERDIKARIS, P., AND KARNIADAKIS, G. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics 378* (2019), 686–707.

[34] RAISSI, M., WANG, Z., TRIANTAFYLLOU, M., AND KARNIADAKIS, G. Deep learning of vortex-induced vibrations. *Journal of Fluid Mechanics 861* (2019), 119–137.

[35] RAISSI, M., YAZDANI, A., AND KARNIADAKIS, G. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science 367*, 6481 (2020), 1026–1030.

[36] RUSANOV, V. V. Calculation of intersection of non-steady shock waves with obstacles. *USSR Computational Mathematics and Mathematical Physics 1*, 2 (1962), 304–320.

[37] SCHWANDER, L., RAY, D., AND HESTHAVEN, J. Controlling oscillations in spectral methods by local artificial viscosity governed by neural networks. *Journal of Computational Physics 431* (2021), 110144.

[38] SHEN, W., AND PARK, M. R. Optimal tracing of viscous shocks in solutions ofviscous conservation laws. *SIAM Journal on Mathematical Analysis 38*, 5 (2007), 1474–1488.

[39] SHU, C.-W. *Essentially Non-Oscillatory and Weighted Essentially Non-Oscillatory Schemes for Hyperbolic Conservation Laws*. Springer, Berlin, Heidelberg, 1998.

[40] SHU, C.-W., AND OSHER, S. Efficient implementation of essentially non-oscillatory shock-capturing schemes. *Journal of Computational Physics 77* (1988), 439–471.

[41] SIRIGNANO, J., AND SPILIOPOULOS, K. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics 375* (2018), 1339–1364.

[42] TANG, H., AND TANG, T. Adaptive mesh methods for one-and two-dimensional hyperbolic conservation laws. *SIAM Journal on Numerical Analysis 41*, 2 (2003), 487–515.

[43] TEYSSIER, R., AND COMMERÇON, B. Numerical methods for simulating star formation. *Frontiers in Astronomy and Space Sciences 6* (2019), 51.

[44] UNNIKRISHNAN, U., HUO, H., WANG, X., AND YANG, V. Subgrid scale modeling considerations for large eddy simulation of supercritical turbulent mixing and combustion. *Physics of Fluids 33*, 7 (2021), 075112.

[45] VAN LEER, B. Towards the ultimate conservative difference scheme I. *Lecture Notes in Physics 18* (1973), 163–168.

[46] VAN LEER, B. Towards the ultimate conservative difference scheme V, a second-order sequel to Godunov's method. *Journal of Computational Physics 32* (1979), 101–136.

[47] VASWANI, A., SHAZEER, N., PARMAR, N., AND ET.AL. Attention is all you need. *Advances in Neural Information Processing Systems 30* (2017).

[48] WANG, X., AND YANG, V. Supercritical mixing and combustion of liquid-oxygen/kerosene bi-swirl injectors. *Journal of Propulsion and Power 33*, 2 (2017), 316–322.

[49] WOODWARD, P., AND COLLELA, P. The numerical simulation of two-dimensional fluid flow with strong shocks. *Journal of Computational Physics 54*, 1 (1984), 115–173.

[50] WU, Y., MA, F., AND YANG, V. Space–time method for detonation problems with finite-rate chemical kinetics. *International Journal of Computational Fluid Dynamics 18*, 3 (2004), 277–287.

[51] XIN, S., NOUSIAS, S., KUTULAKOS, K. N., SANKARANARAYANAN, A. C., NARASIMHAN, S. G., AND GKIOULEKAS, I. A theory of fermat paths for non-line-of-sight shape reconstruction. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), 6800–6809.

[52] XU, Z., LIU, Y., AND SHU, C.-W. Hierarchical reconstruction for discontinuous Galerkin methods on unstructured grids with a WENO type linear reconstruction and partial neighboring cells. *Journal of Computational Physics 228* (2009), 2194–2212.

[53] YANG, V. Modeling of supercritical vaporization, mixing, and combustion processes in liquid-fueled propulsion systems. *Proceedings of the Combustion Institute 28*, 1 (2000), 925–942.

[54] ZONG, N., AND YANG, V. An efficient preconditioning scheme for real-fluid mixtures using primitive pressure–temperature variables. *International Journal of Computational Fluid Dynamics 21*, 5-6 (2007), 217–230.