Commun. Comput. Phys.
doi: 10.4208/cicp.OA-2020-0199

Vol. **28**, No. 5, pp. 2075-2108
November 2020

Convolution Neural Network Shock Detector for Numerical Solution of Conservation Laws

Zheng Sun[1], Shuyi Wang[2], Lo-Bin Chang[2], Yulong Xing[1] and Dongbin Xiu[1,*]

[1] *Department of Mathematics, The Ohio State University, Columbus, OH 43210, USA.*
[2] *Department of Statistics, The Ohio State University, Columbus, OH 43210, USA.*

Received 6 October 2020; Accepted (in revised version) 22 October 2020

Abstract. We propose a universal discontinuity detector using convolution neural network (CNN) and apply it in conjunction of solving nonlinear conservation laws in both 1D and 2D. The CNN detector is trained offline with synthetic data. The training data are generated using randomly constructed piecewise functions, which are then processed using randomized linear advection solver to count for the cases of numerical errors in practice. The detector is then paired with high-order numerical solvers. In particular, we combined high-order WENO in troubled cells with high-order central difference in smooth region. Extensive numerical examples are presented. We observe that the proposed method produces notably sharper and cleaner signals near the discontinuities, when compared to other well known troubled cell detector methods.

AMS subject classifications: 35L65, 35L67, 65M06, 65M99

Key words: Deep neural network, convolution neural network, discontinuity detection, troubled cell, hybrid method, hyperbolic conservation laws.

1 Introduction

One of the challenges for solving nonlinear hyperbolic conservation laws is that even with smooth initial data, the solution may develop discontinuities in finite time. Without an appropriate treatment near the discontinuities, a high-order numerical scheme may generate spurious Gibbs oscillations and may even converge to an entropy-violating solution. The mesh cells containing low regularity of the solution are called "troubled cells." Capture of these cells is crucial in simulations, and its corresponding numerical techniques are referred to as troubled cell indicators or shock detectors in the literature.

*Corresponding author. *Email addresses:* sun.2516@osu.edu (Z. Sun), wang.7649@osu.edu (S. Wang), lobinchang@stat.osu.edu (L.-B. Chang), xing.205@osu.edu (Y. Xing), xiu.16@osu.edu (D. Xiu)

Various shock detection techniques have been developed and studied. Some are developed for finite volume or discontinuous Galerkin (DG) schemes and are used in the form of suitable limiters, see, for example, the minmod-based TVB limiter [4], the moment limiter of Biswas et al. [2] and its modification [3], the monotonicity-preserving limiter [32] and its modification [27], the KXRCF shock-detection technique by Krivodonova et al. [17], and the troubled cell indicator of Fu and Shu [8]. Some of the other shock detection techniques are aimed at improving the numerical performance when high-order essentially non-oscillatory (ENO) or weighted ENO (WENO) type approximations are used. See, for example, multi-resolution (MR) analysis of Harten [10], strong troubled cell indicator of Xu and Shu [37], etc. There have been extensive studies of these shock detection techniques in the literature, and we refer to [24] for a thorough numerical comparison in the context of DG methods and also [21] for hybrid finite difference methods.

Besides these traditional shock detection techniques, there are a few recent work on identifying troubled cells using artificial neural networks (ANNs). Compared with classical indicators, the neural network based indicators are usually free of problem-dependent parameters and are able to avoid certain mislabels, such as smooth extrema in minmod-based indicators. In [25], Ray and Hesthaven proposed to train a multi-layer perceptron, based on a supervised learning strategy, as a troubled cell indicator for DG methods. Its generalization to two-dimensional problems on unstructured grids was studied in [26]. In [33], the idea has also been pursued by Veiga and Abgrall with the study on transferred learning for adapting different methods or meshes. In [7], Feng et al. proposed a characteristic-featured shock wave indicator with one linear hidden layer. Their analysis shows that the indicator guarantees the only detection of discontinuities caused by characteristic curves compressing or intersecting. In [35], Wen et al. studied the combination of an ANN based troubled cell indicator and hybrid finite difference WENO methods.

Motivated by [34], our paper pursues the use of the convolution neural network (CNN) architecture for training the indicator. Widely used in image classification, CNN employs a series of convolution, pooling, and activation operations, followed by fully connected layers. The convolutional operations with kernels involved allow for parameter sharing and feature sharing, which to some degrees carry out invariance and generalizability properties when the CNN is well-trained. With those properties, a well-trained CNN is capable to reliably detect different types of discontinuities as demonstrated in [34]. Therefore, instead of specifying the local stencil as in some existing work, we use more global data in a subregion as the input, and rely on the CNN to extract the relevant information for determining troubled cells. Note that the key for a CNN detector to be successful is the training procedure, which often involves a large training set. Therefore, one of the main challenges in this project is to generate appropriate synthetic data to train our CNNs. We construct the training data in the finite dimensional numerical solution space which may exhibit some different structures near discontinuities, for instance, either smeared discontinuity or spurious oscillation may be observed. We start from piecewise smooth functions as the initial condition and apply numerical solvers on linear advection equations to evolve them for a few time steps to generate the training

data (paired with the exact discontinuity location). The details are given in Section 3.

As an application, we consider the hybrid finite difference WENO method [5, 6, 9, 20, 23, 28] for solving the hyperbolic conservation laws, and examine the numerical performance of the proposed CNN detector. The main idea of the hybrid method is to use the shock detector to identify the location of troubled cells first, and then apply the sophisticated and robust WENO method [1, 14, 29, 30] near the troubled cells and a simple linear scheme away from them. In this paper, we particularly consider the conjugation between the fifth-order WENO method (at troubled cells) and the sixth-order central difference (in smooth regions). Extensive one- and two-dimensional numerical results are provided to illustrate the performance of the proposed CNN shock detectors in identifying discontinuities. Detailed comparisons with the traditional KXRCF type indicator and MR indicator are presented, which demonstrate that the CNN based detector tends to produce cleaner and sharper signal near the discontinuities, and the appropriate threshold for the indicator tends to be less problem-dependent.

The rest of the paper is organized as follows. In Section 2, we outline the idea of the CNN indicator and briefly review some of other indicators in the literature. In Section 3, we give detailed explanations on the training of the network and its coupling with the hybrid finite difference solver in the one-dimensional setting. The two-dimensional network and hybrid method are presented in Section 4. Extensive numerical examples are provided in Section 5 and conclusions are given in Section 6.

2 Problem setup and preliminaries

In this paper, we focus on nonlinear conservation laws

$$u_t+\nabla\cdot f(u)=0, \quad \text{on } D\subset\mathbb{R}^d, \quad d=1,2. \tag{2.1}$$

where f is the flux function. Here D is a bounded domain in $\mathbb{R}^d$ with $D=[a,b]$ for $d=1$ and $D=[a^x,b^x]\times[a^y,b^y]$ for $d=2$. The solutions to (2.1) are usually piecewise continuous and admit jump discontinuities at finite number of points. In a finite difference scheme, we use grid functions associated with a mesh partition to approximate the solutions. For the one-dimensional case, the domain D is partitioned with a set of grid points, $S=\{x_i=(a+ih)\colon i\in\{0,1,\cdots,N\}\}$, where h is the mesh size. We also use the notation $I_{i+\frac{1}{2}}=[x_i,x_{i+1})$ to represent the i-th (mesh) cell. For the two-dimensional case, the set of grid points is taken as $S=\{(x_i,y_j)=(a^x+ih^x,a^y+jh^y)\colon i\in\{0,1,\cdots,N^x\},\ j\in\{0,1,\cdots,N^y\}\}$. For simplicity, we consider square meshes only with $h^x=h^y=h$. The (i,j)-th (mesh) cell is denoted by $I_{i+\frac{1}{2},j+\frac{1}{2}}=[x_i,x_{i+1}]\times[y_j,y_{j+1}]$. At a given time t, the numerical solution is denoted by

$$u_h=u_h(t,x), \quad x\in S. \tag{2.2}$$

The shock detector aims at finding "troubled cells" associated with u_h. Its input is a grid function u_h and its output is a list of detected troubled cells.

2.1 CNN detector

Our work on the development of CNNs for shock detection is motivated by the CNN jump detectors studied in [34], in which the detectors are developed to detect jump discontinuities of a function based on the function values on a fixed grid. Similar to [34], our shock detection procedure is based on a one-step CNN detector for one-dimensional detection ($d=1$) and a two-step CNN detector for two-dimensional detection ($d=2$). Each of the CNNs constructed consists of an input layer, an output layer, and multiple hidden layers such as convolutional layers, pooling layers, and fully connected layers (see Section 2.2 in [34]) where the activation function is the rectified linear unit (ReLU).

For the one-step CNN detector, let $\eta=(\eta_1,\cdots,\eta_N)$ be a binary vector indicating the ground truth cell labels, where $\eta_i=1$ if the i-th cell contains a shock, and $\eta_i=0$ otherwise. Here we construct a detector that first standardizes the numerical PDE solution values in u_h and feeds them into a CNN to obtain an output vector of N real values,

$$\hat{\eta}=(\hat{\eta}_1,\cdots,\hat{\eta}_N)=\mathcal{N}(\tilde{u}_h),$$

as an estimate of the ground truth η, where

$$\tilde{u}_h=\tilde{u}_h(t,x)=\frac{u_h(t,x)-\mu_t}{\sigma_t},\quad x\in S,$$

denotes the set of standardized function values, μ_t and σ_t denote the mean and standard deviation of u_h, respectively. Then, the detector labels each of i-th cell a trouble cell if $\hat{\eta}_i>th$, where the threshold th=0.2 is used in our experiments. See Fig. 1 for an illustration of the shock detection algorithm.

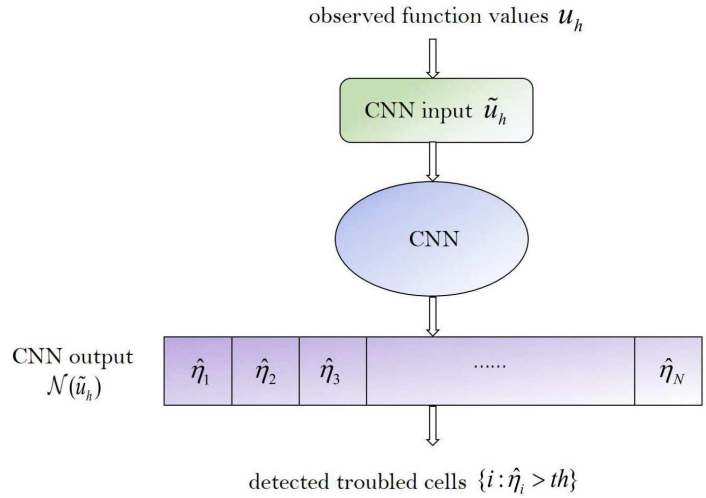


Figure 1: Diagram for the one-step method for discontinuity detection. $\tilde{u}_h$ denotes the standardized vector of u_h.

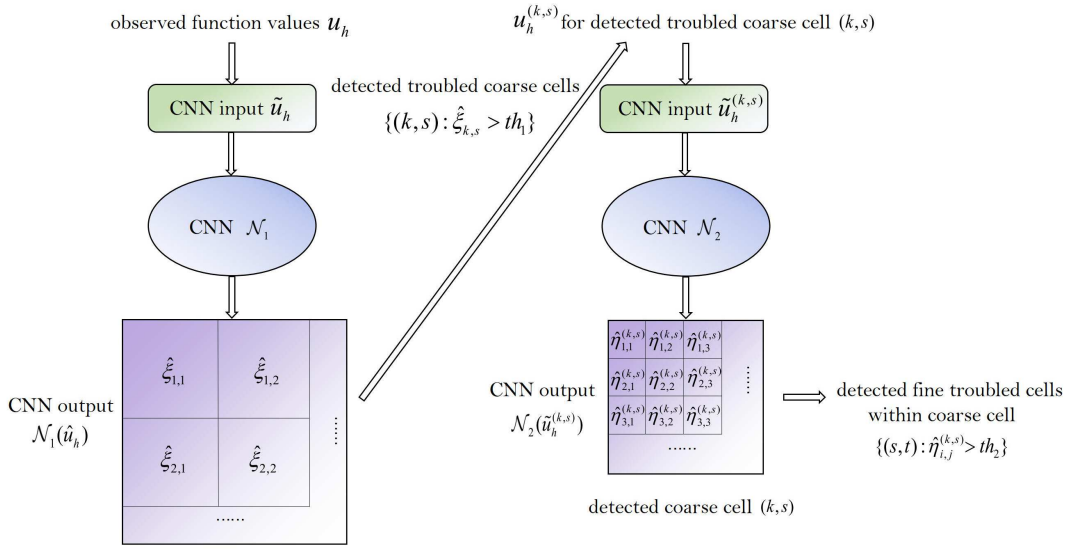


Figure 2: Diagram for the two-step method for discontinuity detection. $\tilde{u}_h$ and $\tilde{u}_h^{(k,s)}$ denote the standardized vectors of u_h and $u^{(k,s)}$, respectively.

For the two-step CNN detector, the algorithm also takes the input u_h and outputs a binary matrix to predict the ground truth labels η, but the detection procedure contains two steps with two CNNs involved to reduce the computational cost. The two-step procedure is illustrated as follows.

Step 1. Define a coarse uniform grid, where each cell is the union of several neighboring cells of the original grid. The original cells are now subcells of the coarse cells. As an example, if the size of the original grid is 101×101, we can choose an 11×11 coarse grid such that each coarse cell contains $10\times10=100$ subcells (original cells). We can then construct a one-step CNN detector to detect coarse cells that contain shocks (troubled coarse cells). Here, we use $\xi=\{\xi_{k,s}\}$ to indicate the ground truth labels, where binary variable $\xi_{k,s}=1$ denotes that the (k,s)-th coarse cell is a troubled coarse cell. The output of the CNN is $\hat{\xi}=\{\hat{\xi}_{k,s}\}$ and a coarse cell (k,s) is detected if $\hat{\xi}_{k,s}>th_1$ where the threshold $th_1=0.5$ is used in our experiments.

Step 2. Next, we refine our detection by detecting fine troubled cells (subcells that contain shocks) within each detected coarse cell. The detection of fine troubled cells is operated on a very small grid – encompassing only the coarse cells. This can be efficiently done with a one-step CNN detector as discussed above. See Fig. 2 for the diagram of the two-step method, where within each detected coarse cell indexed by (k,s), the fine cell $I_{i+\frac{1}{2},j+\frac{1}{2}}$ is detected if $\eta_{i,j}^{(k,s)}>th_2$. The threshold th_2 used in our experiments is 0.2.

2.2 Summary of other methods

In this section, we give a brief review on a few existing troubled cell indicators in the literature, including both the traditional and neural network based indicators. For traditional troubled cell indicators, we present the KXRCF indicator and MR indicator, which are among the "best" indicators as studied in [21] and will be compared with our CNN detector in the numerical tests. For ease of notations, we drop the subscript h and use u to represent the numerical data within this section and also later in Appendix A.

Traditional Troubled Cell Indicators.

- *KXRCF indicator*. The KXRCF shock detection technique was proposed by Krivodonova et al. in [17] for DG method, motivated by its strong superconvergence at the outflow cell boundaries in smooth regions. Later, the strategy was generalized for hybrid finite difference schemes and was shown to be well performed in extensive tests [20–22]. For each cell I_i centered at x_i, one first constructs a pth order interpolation polynomial, denoted by v, using nodal values from the neighboring cells. We take $p=2$ in this paper. Then we evaluate the following quantity:

$$\kappa_i^{\mathrm{KXRCF}}=\frac{\left|\int_{\partial I_i^-}\left(v|_{I_i}-v|_{I_{\mathrm{nb},i}}\right)ds\right|}{h^{\frac{3}{2}}|\partial I_i^-|\,\|v\|_{I_i}}, \tag{2.3}$$

  where ∂I_i^- is the inflow portion of the cell boundary, $I_{\mathrm{nb},i}$ is the neighbor of I_i on the side of ∂I_i^-, and h is the radius of the circumscribed circle in I_i.

  Typically, I_i is marked as a troubled cell if $\kappa_i^{\mathrm{KXRCF}}>1$. While it is reported, a larger threshold, possibly dependent on h, may provide sharper detecting results [8]. In this light, we define

$$\eta_i^{\mathrm{KXRCF}}=-\frac{\log\kappa_i^{\mathrm{KXRCF}}}{\log h}, \tag{2.4}$$

  as the indicator for the mesh cell I_i.

- *MR indicator*. The MR analysis was introduced by Harten in [10, 11] and then systematically studied in [12]. Its main idea is to generate approximations of a function on nested dyadic grids and then locate discontinuities by comparing the coarse and fine grid values. The MR indicator we particularly concern in this paper is from [21], which uses this methodology on single-level grids.

$$\kappa_i^{\mathrm{MR}}=|u_i-\tilde{u}_i|,\quad\text{with}\quad\tilde{u}_i=\frac{1}{2}(u_{i-1}+u_{i+1}). \tag{2.5}$$

  Suppose u has $q-1$ continuous derivatives and a jump discontinuity in its qth derivative, it is then expected that $\kappa_i^{\mathrm{MR}}=\mathcal{O}(h^{\min(2,q)})$. Typically, the cell I_i is marked

as a troubled cell if $\kappa_i^{\rm MR} > \varepsilon_{\rm MR} h$, with a properly chosen $\varepsilon_{\rm MR}$. In our paper, to include h in the cut-off, we instead consider the following indicator

$$\eta_i^{\rm MR} = \frac{\kappa_i^{\rm MR}}{h}. \tag{2.6}$$

Neural Network Based Indicators.

- *Indicator of Ray and Hesthaven*. It uses the same input as that of the minmod-based indicators, i.e.,

$$\{\bar{u}_{i-1}, \bar{u}_i, \bar{u}_{i+1}, u^+_{i-\frac{1}{2}}, u^-_{i+\frac{1}{2}}\}, \tag{2.7}$$

  where $\bar{u}_i$ is the average of u on I_i and $u^+_{i-\frac{1}{2}}$, $u^-_{i+\frac{1}{2}}$ are the right and left limits of u at $x_{i-\frac{1}{2}}$ and $x_{i+\frac{1}{2}}$, respectively. The indicator is trained using exact functions with known discontinuities. The generalization to two-dimensional problems on unstructured grids is pursued in [26].

- *Indicator of Veiga and Abgrall*. In the indicator proposed by Veiga and Abgrall [33], a wider class of local features is used as the input for the multilayer perceptron neural networks. In the one-dimensional case, they are

$$\left\{\bar{u}_{i-1}, \bar{u}_i, \bar{u}_{i+1}, u^+_{i-\frac{1}{2}}, u^-_{i+\frac{1}{2}}, u^-_{i-\frac{1}{2}}, u^+_{i+\frac{1}{2}}, \bar{u}_i - \bar{u}_{i-1}, \frac{\bar{u}_{i+1} - \bar{u}_{i-1}}{2}, \bar{u}_i - \bar{u}_{i-1}, h\right\}. \tag{2.8}$$

  The data set for training is generated with a DG solver for the advection equation with different initial conditions. The troubled cells are flagged with a high-order limiter in [16]. The two-dimensional generalization, along with the transfer learning strategy, has also been considered.

- *Indicator of Wen et al.*. In [35], Wen et al. proposed to combine the neural network shock detection technique with the hybrid finite difference methods. The 5-point local stencil

$$\{u_{i-2}, u_{i-1}, u_i, u_{i+1}, u_{i+2}\} \tag{2.9}$$

  is used as the input of the neural network for flagging the discontinuity at x_i. The indicator is trained with data generated from exact functions as those in [25] and is applied in a dimension-by-dimension manner in two-dimensional simulations.

3 Method description: One-dimensional case

In this section, we present our algorithm in detail. One of the distinct features of our method is the construction of synthetic training data. Rather than using grid functions, as in many other existing work, we employ a randomized procedure to cast the training data into a proper linear subspace where u_h resides.

3.1 Randomized construction of training data

Compared with the exact solution $u(t,x)$ of conservation laws, the numerical solution $u_h(t,x)$ may exhibit a few different structures near discontinuities. For example, these structures may include smearing caused by numerical dissipation and spurious oscillations if an improper spatial difference scheme is chosen. With these in mind, when preparing the training data, we propose to use numerical solutions as the input to include the effects, and pair them up with the discontinuities of the exact solutions. The procedure is briefly outlined below.

$$\text{Input data:} \quad u^0 \xrightarrow{\text{sampling at grid points}} u_h^0 \xrightarrow{\text{numerical solver}} u_h. \tag{3.1a}$$

$$\text{Output data:} \quad u^0 \xrightarrow{\text{exact solver}} u \xrightarrow{\text{detecting discontinuity}} \eta. \tag{3.1b}$$

Note the discontinuities of u_h may locate differently from those of u due to the phase error in the numerical discretization. To avoid this mismatch, we apply the numerical solver only for a few steps, so that the difference is negligible.

Following the diagram in (3.1), we construct the grid function u_h by solving $u_t+au_x=0$ on $D=[-1,1]$ with one of the following schemes for a random number of time steps. The simple linear advection equation is chosen so that we know the exact location of the discontinuity after a few time steps. To simplify the description, we denote by $a\sim U(A)$, if a is a random variable with a uniform distribution on A. The detailed procedure is given below.

1. (Problem setups) Randomly generate the advection coefficient $a\sim U(\{-1,1\})$, the number of time step $N_t\sim U(\{0,\cdots,20\})$, and the spatial discretizations from the follow methods:
   (a) $(2l)$th order central difference method, with $l=1,\cdots,4$.
   (b) $(2l-1)$th order upwind finite difference method, with $l=1,\cdots,5$.
   (c) $(2l-1)$th order finite difference WENO method, with $l=2,\cdots,5$.
2. (Generate u^0) Generate the exact initial function u^0 with the following steps.
   (a) Randomly select an integer $N_d\sim U(\{0,1,2,3\})$ as the number of discontinuities.
   (b) Generate N_d random points, subject to $U(D)$, as locations for discontinuities, which divide D into (N_d+1) subdomains.
   (c) Inside each subdomain, create a random Fourier series

$$a_0+\sum_{n=1}^{N_F}\left(a_n\cos(nx)+b_n\sin(nx)\right), \tag{3.2}$$

   where $N_F\sim U(\{0,\cdots,10\})$ and a_n,b_n are i.i.d. Gaussian random variables $N(0,1)$.

3. ($u^0 \to u_h^0$) Evaluate the grid function u_h^0.
4. ($u_h^0 \to u_h$) Pull u_h^0 into the range of the numerical solver, by solving $u_t + au_x = 0$ with the third-order Runge–Kutta method and the chosen spatial discretization for N_t steps. $h = 2/201$ and $\Delta t = h/10$ are used in the computation, and the final numerical solution is denoted by u_h.
5. ($u_h \to \eta$) Label the "discontinuity" of u_h, by setting $\eta_i = 1$ if the exact solution $u^0(x - aN_t\Delta t)$ has a discontinuity on $I_{i+\frac{1}{2}}$.

3.2 Network training

A one-step CNN is constructed to detect troubled cells from a grid of $N = 200$ cells. The architecture of the CNN is summarized in Table 1.

Table 1: Architecture of 1D-CNN.

Layer	input size	kernel size	num of kernel	stride	output size
conv1	202	2	24	1	201×24
conv2	201×24	2×24	24	1	200×24
conv3	200×24	2×24	24	1	199×24
conv4	199×24	2×24	24	1	198×24
conv5	198×24	2×24	24	2	99×24
fully connected	99×24				201

To train the parameters of the CNN, we generate a synthetic dataset of $n = 1{,}000{,}000$ grid functions and corresponding labels, $\{(u_h^m, \eta^m) : m = 1, \cdots, n\}$, using the generation procedure in the previous section. The network is trained with Keras API (https://keras.io/), by using mean squared loss function

$$\frac{1}{n}\sum_{m=1}^{n} \|\eta^m - \mathcal{N}(\tilde{u}_h^m)\|_{l_2}^2.$$

We use 90% of the synthetic data for training and the remaining 10% for validation. The input is a 202-length vector $\tilde{u}_h$ and the output is a 201-length vector which gives predictions on each cell (Fig. 1). The training is based on Adam optimization [15] with the mini-batch size of 5,000 and 1,000 epochs.

In the next subsection, we apply the CNN detector to locate nonsmooth regions (called buffer zones), of which troubled cells are near the centers. If the troubled cells or their adjacent cells can be identified, the buffer zones could be well constructed by intervals with detected cells in the centers. To demonstrate the detection performance in terms of the application, we generate 1,000 testing functions, produce $1000 \times 201 = 201{,}000$ cells, of which 1,481 cells are troubled cells, and then test if our detector can detect the troubled cells or their adjacent cells. We find that 98.3120% of those troubled cells are detected or

with at least one adjacent cell detected, and only 0.0546% of normal cells (non-troubled cells whose adjacent cells also have no discontinuity) are detected or with adjacent cells detected.

3.3 Hybrid numerical scheme for conservation laws

The semidiscrete finite difference approximation of (2.1) is given by

$$\frac{d}{dt}u_h(t,x_i)+\frac{\widehat{f}_{i+\frac{1}{2}}-\widehat{f}_{i-\frac{1}{2}}}{h}=0, \tag{3.3}$$

where $\widehat{f}_{i+\frac{1}{2}}$ is the high-order numerical flux. In a hybrid method, one first applies the discontinuity indicator to locate the troubled cells. In the smooth region, the flux construction of an efficient and simple-to-implement high-order method is used. For example, we will use the sixth-order central difference method in our numerical tests. In this case, we have

$$\widehat{f}_{i+\frac{1}{2}}=\frac{1}{60}f_{i-2}-\frac{2}{15}f_{i-1}+\frac{37}{60}f_i+\frac{37}{60}f_{i+1}-\frac{2}{15}f_{i+2}+\frac{1}{60}f_{i+3}, \tag{3.4}$$

where $f_{i+l}=f(u_h(t,x_{i+l}))$, $l=-2,\cdots,3$. In the nonsmooth region near the troubled cells, the more sophisticated WENO approximations will be used. We refer to [29] for detailed descriptions of WENO methods. Finally, the numerical solution is updated along with a strong-stability-preserving (SSP) time discretization of (3.3). For example, if we denote (3.3) as $\frac{d}{dt}u_h=L(u_h)$, its third-order SSP Runge–Kutta (SSPRK3) discretization is given by

$$u_h^{(1)}=u_h^n+\Delta tL(u_h^n), \tag{3.5a}$$

$$u_h^{(2)}=\frac{3}{4}u_h^n+\frac{1}{4}\left(u_h^{(1)}+\Delta tL(u_h^{(1)})\right), \tag{3.5b}$$

$$u_h^{n+1}=\frac{1}{3}u_h^n+\frac{2}{3}\left(u_h^{(2)}+\Delta tL(u_h^{(2)})\right), \tag{3.5c}$$

which consists of convex combinations of three Euler forward steps. Here we use the superscript to represent time steps and stages.

In summary, the following algorithm is used for discretizing (2.1).

1. Apply the CNN detector to mark the troubled cells $I_{i+\frac{1}{2}}$, only once at the beginning of each Runge–Kutta time discretization.
   (a) If the total number of cells exceeds 202, we standardize the data, divide the cells into several patches, and apply the indicator on each patch.
   (b) If the total number of cells is less than 202, we standardize the data, extend the data on both ends with constant values, and then apply the CNN detector. (Note that one can also use smoother extensions, for example, to make the derivatives continuous at the boundary.)

2. Define the buffer zone surrounding the troubled cell $I_{i+\frac{1}{2}}$ as $\bigcup_{l=-(N_b+1)}^{N_b} I_{i+\frac{1}{2}+l}$, with $N_b=2$ in our numerical tests.

3. Apply the SSPRK3 discretization. In each Euler forward stage, do the following.

   (a) Construct numerical flux at $x_{i+\frac{1}{2}}$ for all i. Use the WENO procedure if $x_{i+\frac{1}{2}}$ is inside the buffer zone of a troubled cell; otherwise, use the interpolation associated with the central difference method.

   (b) Update the Euler forward stage.

Remark 3.1. For sixth-order central difference method, the stencil for evaluation of numerical flux at $x_{i+\frac{1}{2}}$ is given by $\{x_{i-2},x_{i-1},x_i,x_{i+1},x_{i+2},x_{i+3}\}$. With the buffer size $N_b=2$, this stencil will not contain any troubled cell if $x_{i+\frac{1}{2}}$ is outside of the buffer zone.

4 Method description: Two-dimensional case

4.1 Randomized construction of training data

The two-dimensional data u_h is generated by solving the advection equation $u_t+au_x+bu_y=0$ on $D=[-1,1]\times[-1,1]$.

1. (Problem setups) Generate the random advection coefficient $(a,b)=(\cos\varphi,\sin\varphi)$ and the number of time steps N_t, where $\varphi\sim U([0,2\pi])$ and $N_t\sim U(\{0,\cdots,10\})$. Randomly chose one of the three spatial discretizations from Section 3.1.

2. (Generate u^0) Generate the exact initial function u^0 with following steps.

   (a) Generate a random curve, cutting the domain D into two subregions, as the location of discontinuity. We employ two cases:

      i. Random line cut, $\cos(\theta)(x-x_0)+\sin(\theta)(y-y_0)=0$, with $\theta\sim U([0,2\pi])$ and $(x_0,y_0)\sim U(D)$.

      ii. Random circle cut, $(x-x_0)^2+(y-y_0)^2=r^2$, with $r\sim U([1,3])$ and $(x_0,y_0)\sim U([-2,2]\times[-2,2])$.

   (b) On each smooth subregion, define the function as

$$\sum_{m+n\leq 4} a_{m,n}^{(i)} P_m(x)P_n(y), \tag{4.1}$$

   where P_n are the standard Legendre polynomials, $a_{m,n}$ are coefficients randomly generated from Gaussian distribution $N(0,1)$.

   (c) Rescale the function onto the domain of numerical dependence.

3. ($u^0\to u_h^0$) Evaluate the grid function u_h^0.

4. $(u_h^0 \to u_h)$ Pull u_h^0 into the range of numerical solver, by solving $u_t + au_x + bu_y = 0$ with the chosen spatial discretization scheme and with the third-order Runge–Kutta method for N_t steps. $h = h^x = h^y = 2/201$ and $\Delta t = h/5$ are used. The final solution is denoted by u_h.

5. $(u_h \to \eta)$ Label $\eta_{i,j} = 1$, if the exact solution $u^0(x - aN_t\Delta t, y - bN_t\Delta t)$ is discontinuous in the cell $I_{i+\frac{1}{2},j+\frac{1}{2}}$.

4.2 Network training

A two-step CNN detector is constructed to detect troubled cells over a 101×101 grid. The coarse grid for the step 1 CNN is of size 11×11 such that each coarse cell contains $10\times10=100$ subcells (original grid cells). The following two tables summarize the architectures of the two CNNs.

Table 2: Architecture of first step 2D-CNN.

Layer	input size	kernel/pooling size	num of kernel	stride	output size
conv1	101×101	4×4	32	1	$98\times98\times32$
maxpooling1	$98\times98\times32$	2×2		2	$49\times49\times32$
conv2	$49\times49\times32$	$2\times2\times32$	32	1	$48\times48\times32$
conv3	$48\times48\times32$	$2\times2\times32$	32	1	$47\times47\times32$
conv4	$47\times47\times32$	$2\times2\times32$	32	1	$46\times46\times32$
fully connected	$46\times46\times32$				10×10

Table 3: Architecture of second step 2D-CNN.

Layer	input size	kernel size	num of kernel	stride	output size
conv1	11×11	2×2	32	1	$10\times10\times32$
conv2	$10\times10\times32$	$2\times2\times32$	32	1	$9\times9\times32$
conv3	$9\times9\times32$	$2\times2\times32$	32	1	$8\times8\times32$
fully connected	$8\times8\times32$				10×10

To train the parameters of two CNNs, we generate a synthetic dataset of $n = 200{,}000$ grid functions and corresponding labels, $\{(u_h^m, \eta^m) : \ m = 1,\cdots,n\}$, using the generation procedure in the previous subsection. The training of the first CNN model is based on the Adam optimization with the mini-batch size of 2,000 and 5,000 epochs, where the training set $\{(u_h^m, \xi^m) : \ m = 1,\cdots,n\}$ is constructed using the synthetic dataset, and ξ^m is the set of binary labels corresponding to the coarse cells for the m-th function. The training of the second CNN model is based on the Adam optimization with the mini-batch size of 100,000 and 1,000 epochs, where the training set $\{(u_h^{m,(k,s)}, \eta^{m,(k,s)}) : \ m \in \{1,\cdots,n\},\ i,j \in \{1,\cdots,10\}\}$ is also constructed using the synthetic dataset, where $\eta^{m,(k,s)}$ is the set of binary labels corresponding to the (k,s)-th coarse cell for the m-th function.

The loss functions for these two CNN models are mean squared loss function as in the one-step CNN training.

To demonstrate the detection performance, we generate 1,000 testing functions, produce $1000\times100\times100=10{,}000{,}000$ cells, of which 107,480 cells are troubled cells, and therefore produce $1000\times100=100{,}000$ coarse cells, of which 10,721 coarse cells are troubled coarse cells. For the troubled coarse cell detection, 97.5469% of the coarse troubled cells are detected, and 0.2655% of the non-troubled coarse cells are detected. For the step 2 detection, 99.6278% of the troubled cells are detected or with at least one of the eight nearest cells detected, and 0.0877% of the normal cells are detected or with at lease one of the eight nearest cells detected.

4.3 Numerical scheme for conservation laws

The finite difference discretization of the two-dimensional conservation laws

$$u_t+f(u)_x+g(u)_y=0 \tag{4.2}$$

is given by

$$\frac{d}{dt}u_h(t,x_i,y_j)+\frac{\widehat{f}_{i+\frac{1}{2},j}-\widehat{f}_{i-\frac{1}{2},j}}{h^x}+\frac{\widehat{g}_{i,j+\frac{1}{2}}-\widehat{g}_{i,j-\frac{1}{2}}}{h^y}=0. \tag{4.3}$$

Here $\widehat{f}_{i+\frac{1}{2},j}$ and $\widehat{g}_{i,j+\frac{1}{2}}$ are numerical fluxes approximating $f(u)$ and $g(u)$ at $(x_{i+\frac{1}{2}},y_j)$ and $(x_i,y_{j+\frac{1}{2}})$, respectively, whose values are obtained from the one-dimensional procedures for either central or WENO approximations. As before, the SSPRK3 method is used for time discretization. The detailed numerical procedure is given below.

1. Apply the CNN detector to mark the troubled cells $I_{i+\frac{1}{2},j+\frac{1}{2}}$, only once at the beginning of each Runge–Kutta time discretization.
   (a) If the data set is larger than the designed inputs of the CNN detector, we standardize the data, divide the data into several subregions, and apply the CNN detector to them individually.
   (b) If the data set is smaller than the designed inputs, we standardize the data, extend the data set with constant values (or smoother extensions) on the boundary, and apply the CNN detector.
2. Define the buffer zone surrounding $I_{i+\frac{1}{2},j+\frac{1}{2}}$ as $\bigcup_{l,m=-(N_b+1)}^{N_b+1} I_{i+\frac{1}{2}+l,j+\frac{1}{2}+m}$. We take $N_b=2$ for two-dimensional tests.
3. Apply the SSPRK3 discretization. In each Euler forward stage, do the following.
   (a) Construct numerical flux at $(x_{i+\frac{1}{2}},y_j)$ and $(x_i,y_{j+\frac{1}{2}})$ for all i and j: Use the WENO procedure if the points are inside the buffer zone of a troubled cell; otherwise, use the interpolation associated with the central difference method.
   (b) Update the Euler forward stage.

5 Numerical examples

In this section, we conduct numerical tests to test the performance of the proposed CNN detector with hybrid methods. Examples in Section 5.1 with one-dimensional scalar equations are served as proofs-of-principle for examining the performance of the CNN detector with smeared discontinuities and shock waves. Detailed comparisons with KXRCF and MR indicators are provided in Section 5.2 and Section 5.3 for one- and two-dimensional Euler equations.

In all numerical tests, the SSPRK3 time discretization (3.5) is used. We take $N_b=2$ for the buffer zone unless otherwise stated. In numerical tests for Euler equations, only the density functions ρ are tested with the indicators for identifying troubled cells. In the figures showing the troubled cell histories, we only plot cells with signals greater than the thresholds, and the buffer cells are not included in these plots.

5.1 Scalar conservation laws in one dimension

The time step is set as $\Delta t=0.2h$ for all numerical tests in this subsection. We have run simulations with various mesh sizes with 200, 400 and 800 points used. The threshold for the CNN detector is taken as 0.2.

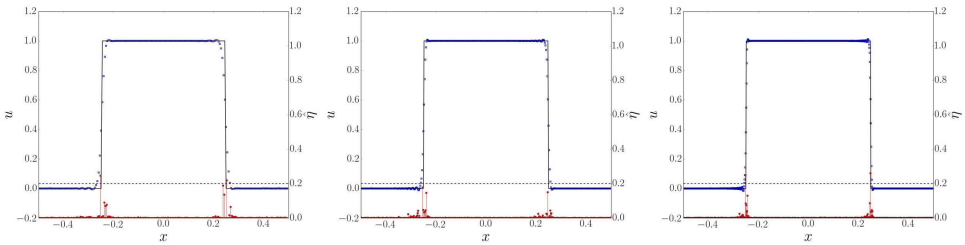
Example 5.1 (Linear advection equation)**.** In this numerical test, we consider the simple linear advection equation with a traveling square wave

$$u_t+u_x=0,\quad x\in\Big(-\frac{1}{2},\frac{1}{2}\Big),\quad u(x,0)=\begin{cases}1, & x\in(-\frac{1}{4},\frac{1}{4}),\\ 0, & \text{otherwise.}\end{cases} \tag{5.1}$$

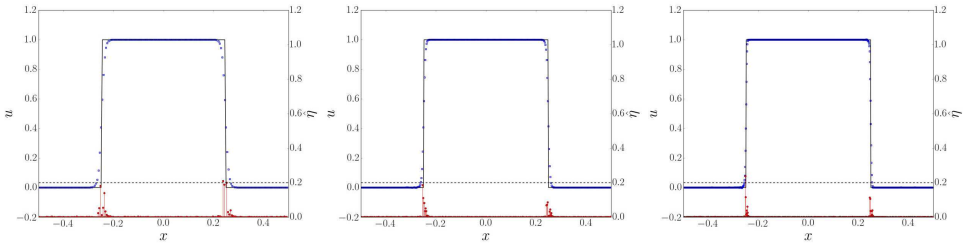
Periodic boundaries are applied. The solutions after 1 period, at $T=1$, together with the corresponding CNN outputs, are given in Fig. 3. The time histories of marked troubled cells are given in Fig. 4. We can see that the CNN detector does capture the moving discontinuity every now and then. But in the first row of Fig. 3, it seems there are small oscillations generated near the discontinuities. The possible cause of this numerical artifact, is that the discontinuities are too smeared. There is a wide region of transition points, and our CNN output marks one or two points in the middle of this region. With the buffer size of $N_b=2$, the transition region cannot be covered inside the buffer zone. If we expand the buffer zone from $N_b=2$ to $N_b=6$, one can see from the second rows of Fig. 3 and Fig. 4 that although similar troubled cells are identified during the computation, the oscillations nearing the discontinuities seem to be suppressed.

Example 5.2 (Burgers equation)**.** We then apply the scheme to solve the initial value problem of the Burgers equation

$$u_t+\left(\frac{u^2}{2}\right)_x=0,\quad x\in(0,2\pi),\quad u(x,0)=1+\sin(x). \tag{5.2}$$

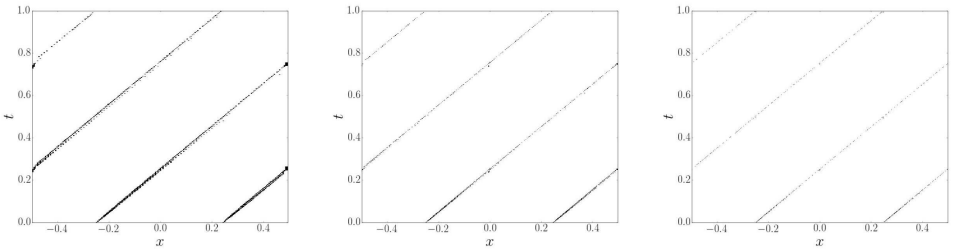


(a) $N_b = 2,\ N = 200$ (b) $N_b = 2,\ N = 400$ (c) $N_b = 2,\ N = 800$

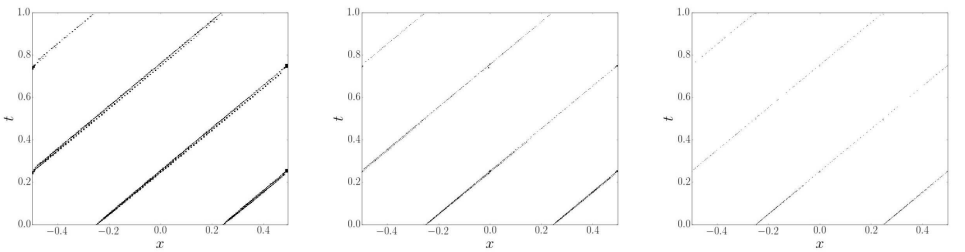


(d) $N_b = 6,\ N = 200$ (e) $N_b = 6,\ N = 400$ (f) $N_b = 6,\ N = 800$

Figure 3: Solution profiles and signal outputs of the CNN detector at $T=1$ for the linear advection equation in Example 5.1. First row: $N_b=2$. Second row: $N_b=6$. Solid black lines: exact solution. Blue squares: numerical solution. Red circles with stems: indicator output. Black dashed lines: threshold for the indicator.

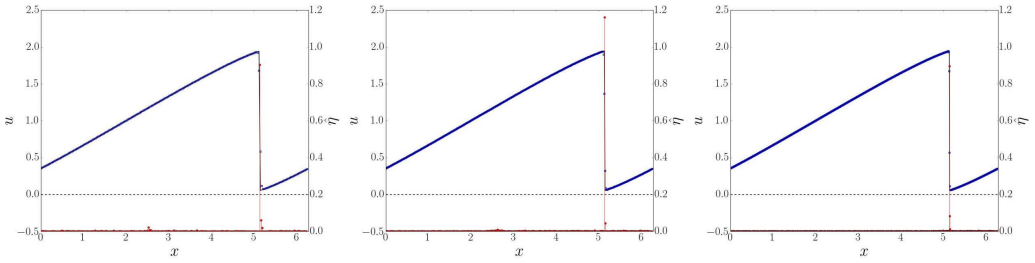


(a) $N_b = 2,\ N = 200$ (b) $N_b = 2,\ N = 400$ (c) $N_b = 2,\ N = 800$



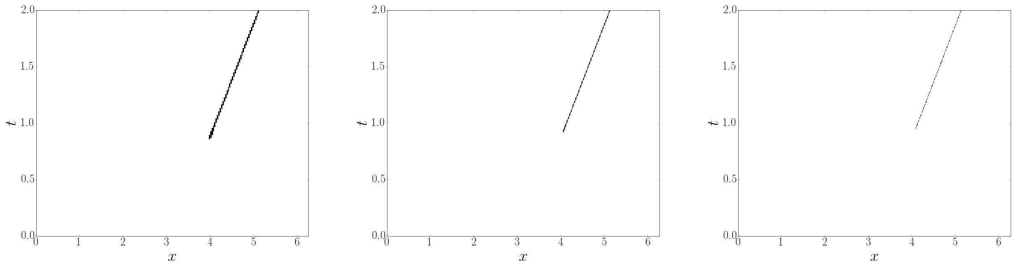
(d) $N_b = 6,\ N = 200$ (e) $N_b = 6,\ N = 400$ (f) $N_b = 6,\ N = 800$

Figure 4: Troubled cell histories of the CNN detector for the advection equation in Example 5.1. First row: $N_b=2$. Second row: $N_b=6$.



(a) $N=200$ (b) $N=400$ (c) $N=800$

Figure 5: Solution profiles and signal outputs of the CNN detector at $T=2$ for the Burgers equation in Example 5.2. Solid black lines: exact solution. Blue squares: numerical solution. Red circles with stems: indicator output. Black dashed lines: threshold for the indicator.



(a) $N=200$ (b) $N=400$ (c) $N=800$

Figure 6: Troubled cell histories of the CNN detector for the Burgers equation in Example 5.2.

Periodic boundary conditions are applied. The solution develops into a right-moving shock wave starting from $T=1$. The final solution at $T=2$ and the corresponding outputs of the CNN detector are given in Fig. 5. The marked troubled cells during the simulations are reported in Fig. 6. In this numerical test with a shock wave, since the sharp profiles of the discontinuities are maintained with fewer transition points, there is no observable oscillation with $N_b=2$. The CNN detector does capture the formation of the shock wave starting from $T=1$.

5.2 Euler equations in one dimension

In this subsection, we apply the hybrid method to solve Euler equations in one dimension

$$\begin{pmatrix} \rho \\ \rho\mu \\ E \end{pmatrix}_t + \begin{pmatrix} \rho\mu \\ \rho\mu^2+p \\ \mu(E+p) \end{pmatrix}_x = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \tag{5.3}$$

where ρ is the density, μ is the velocity, E is the internal energy, and $p=(\gamma-1)(E-\frac{\rho}{2}\mu^2)$, with $\gamma=1.4$, is the pressure. We use 200, 400 and 800 mesh points for simulations,

and the time step size is taken as $\Delta t = 0.6h$. Comparisons between the CNN detector with the KXRCF indicator and the MR indicator are also provided. All these indicators produce numeric signals for troubled cells and there is always a threshold parameter that one can tune with: a larger threshold tends to include fewer troubled cells while suffering the danger to generate oscillations. Here we look into the possibility of having problem-independent and predetermined thresholds. We have tested the problems with various parameters, chosen the ones with overall good performance and fixed them in the following tests: 0.2 for the CNN detector, 0.5 for the KXRCF indicator and 1 for the MR indicator. We remark that for KXRCF and MR indicators, a finely tuned and problem-dependent threshold may provide better results for each particular test.

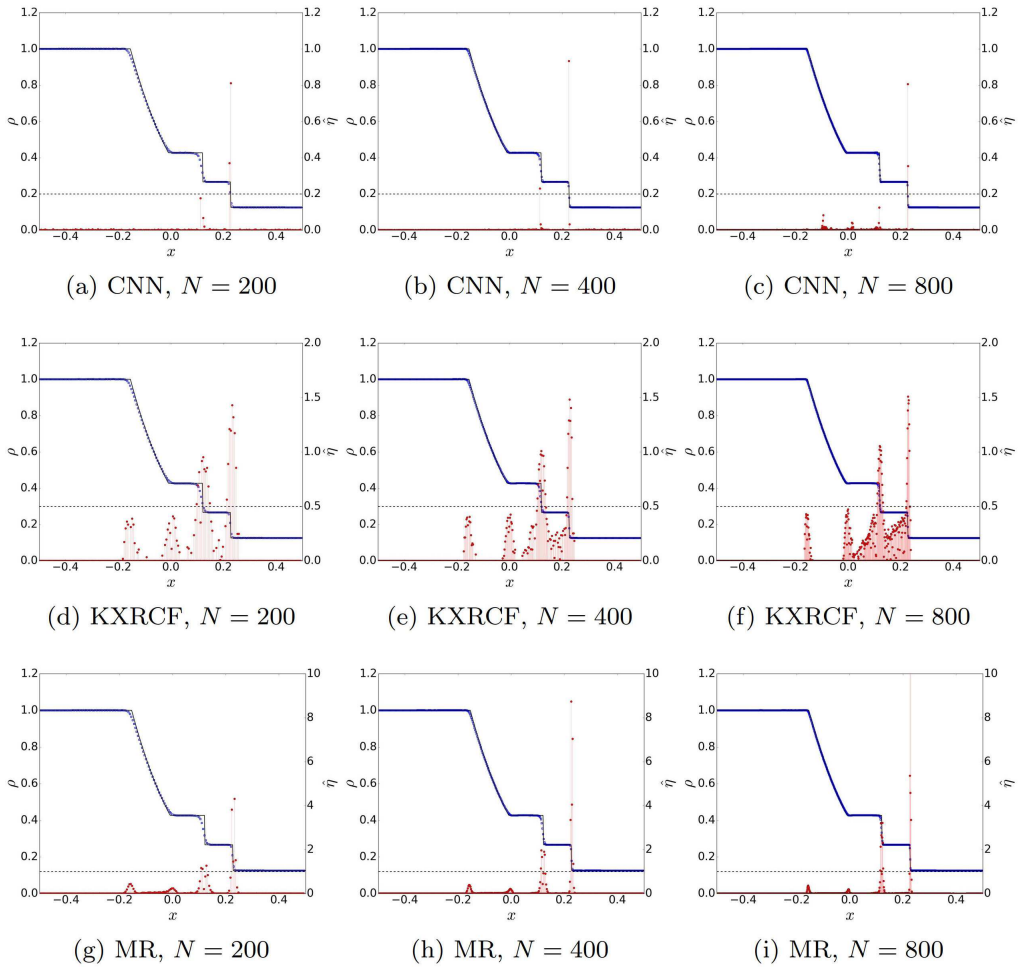
Example 5.3 (Sod problem)**.** In this classical Riemann problem test, the initial condition is taken as

$$(\rho,\mu,p) = \begin{cases} (1,0,1), & x \le 0, \\ (0.125,0,0.1), & x > 0. \end{cases} \tag{5.4}$$

The solution profiles of the hybrid methods with CNN, KXRCF and MR indicators at $T = 0.13$, along with corresponding indicator signals, are given in Fig. 7. The time histories of the marked troubled cells are provided in Fig. 8. It can be seen from Fig. 7, CNN detector seems to produce sharper signal at the discontinuities, while the outputs of KXRCF and MR indicators seem to have a spread into the neighboring cells. From the troubled cell histories, we can also see that the CNN detector captures thinner and finer trajectories of the moving discontinuities, while the troubled cells for KXRCF and MR indicators are wider. We also summarize the percentages of troubled cells, both with and without buffers, in Table 4, from which one can see that the percentages of marked troubled cells decrease when the mesh sizes N increase. Also, CNN detector marks the least amount of cells, followed by MR indicator, and then KXRCF indicator.

Table 4: Troubled cell percentages of the Sod problem in Example 5.3. "Avg" means the averaged percentages of marked cells for all time steps. "Max" corresponds to the maximum percentages that ever occurred among all time steps. "nbf" means the buffer cells are excluded in the counting, and "bf" means the buffer cells are included in the counting.

N	indicator	Avg(nbf)	Max(nbf)	Avg(bf)	Max(bf)
200	CNN	1.09	2.50	5.00	8.50
	KXRCF	9.81	14.00	15.37	22.00
	MR	4.39	7.50	10.40	15.50
400	CNN	0.50	1.25	2.32	4.75
	KXRCF	6.64	9.75	10.19	14.25
	MR	3.08	5.25	6.20	9.50
800	CNN	0.21	0.62	0.97	2.00
	KXRCF	4.40	6.12	6.46	8.62
	MR	1.98	3.62	3.57	5.62



(a) CNN, $N=200$ (b) CNN, $N=400$ (c) CNN, $N=800$

(d) KXRCF, $N=200$ (e) KXRCF, $N=400$ (f) KXRCF, $N=800$

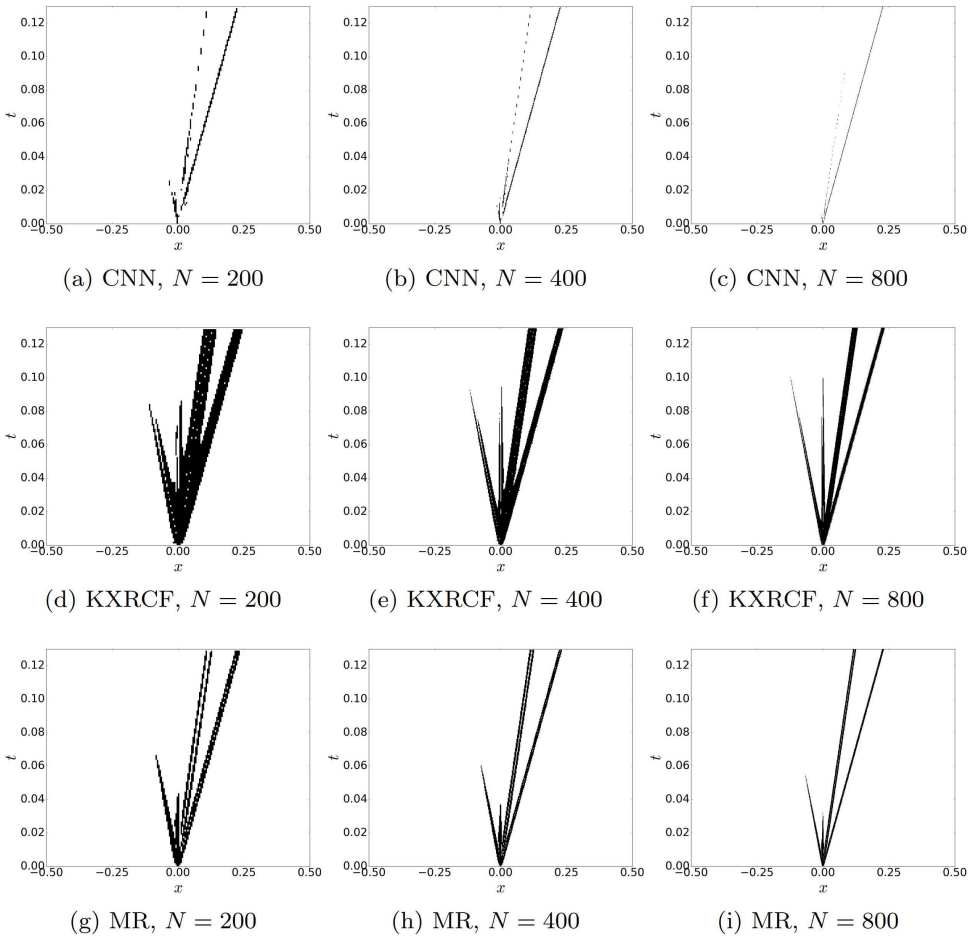
(g) MR, $N=200$ (h) MR, $N=400$ (i) MR, $N=800$

Figure 7: Signals of different troubled cell indicators for the Sod problem in Example 5.3. First row: CNN detector. Second row: KXRCF indicator. Third row: MR indicator. First column: $N=200$. Second column: $N=400$. Third column: $N=800$. Solid black lines: exact solution. Blue squares: numerical solution. Red circles with stems: indicator output. Black dashed lines: threshold for the indicator.

Example 5.4 (Interacting blast waves)**.** In this example, we consider the interaction of two blast waves with the initial data

$$(\rho,\mu,p)=\begin{cases}(1,0,1000), & x\leq -0.1,\\ (1,0,0.01), & 0.1<x\leq 0.9,\\ (1,0,100), & x>0.9.\end{cases} \tag{5.5}$$

Reflective boundaries are imposed both at $x=0$ and $x=1$, and we refer to [36] for detailed descriptions of this example. We solve the problem up to $T=0.4$. The numerical density, and the indicator outputs, on $[0,1]$ are given in Fig. 9. The troubled cell histories and their percentages are reported in Fig. 10 and Table 5. For this problem, it seems that the



(a) CNN, $N = 200$ (b) CNN, $N = 400$ (c) CNN, $N = 800$

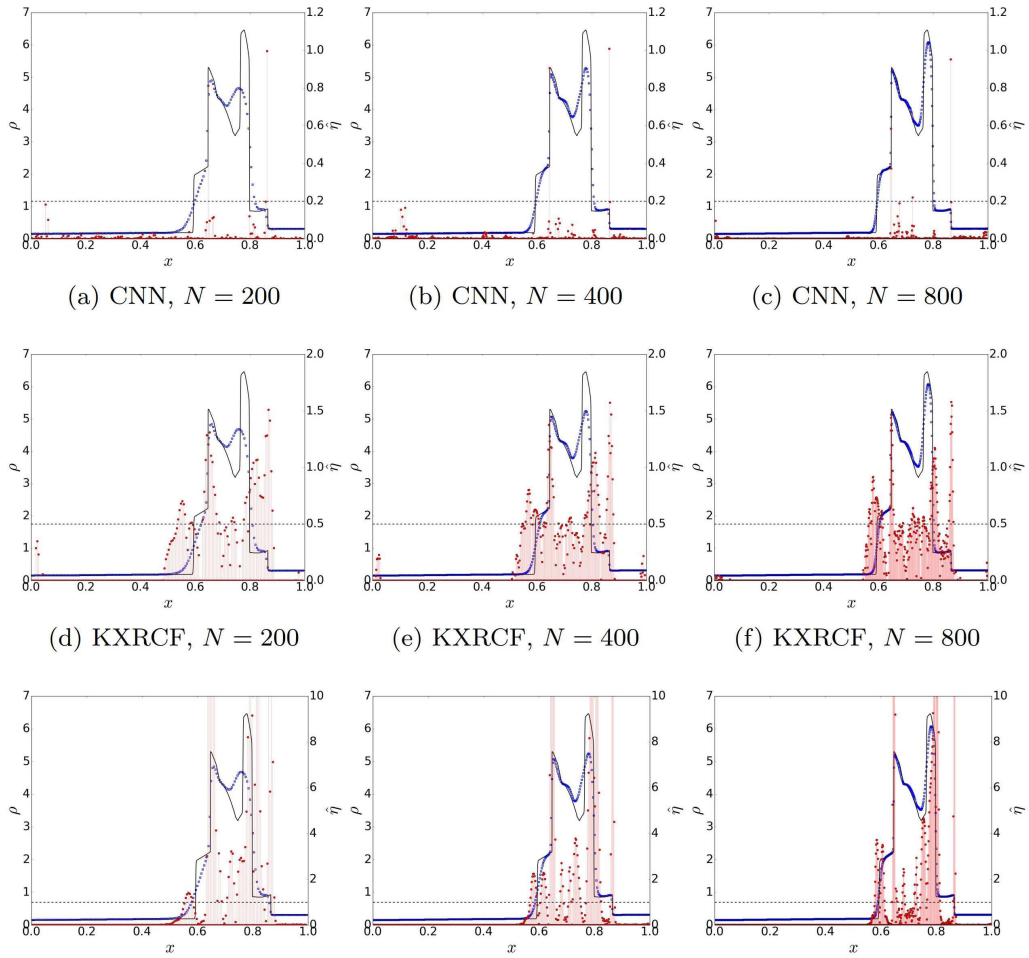
(d) KXRCF, $N = 200$ (e) KXRCF, $N = 400$ (f) KXRCF, $N = 800$

(g) MR, $N = 200$ (h) MR, $N = 400$ (i) MR, $N = 800$

Figure 8: Troubled cell histories of different indicators for the Sod problem in Example 5.3. First row: CNN detector. Second row: KXRCF indicator. Third row: MR indicator. First column: $N=200$. Second column: $N=400$. Third column: $N=800$.

Table 5: Troubled cell percentages for the interacting blast waves in Example 5.4. See Table 4 for explanations of "Avg", "Max", "nbf" and "bf".

N	indicator	Avg(nbf)	Max(nbf)	Avg(bf)	Max(bf)
200	CNN	2.26	4.50	9.30	15.50
	KXRCF	21.01	26.00	30.23	39.00
	MR	19.26	26.00	27.16	36.00
400	CNN	1.05	2.75	4.37	7.75
	KXRCF	15.70	20.75	20.91	30.75
	MR	14.04	21.25	18.65	29.00
800	CNN	0.51	1.38	2.13	4.00
	KXRCF	10.79	15.75	14.03	21.75
	MR	9.42	14.88	12.31	20.25



(a) CNN, $N=200$ (b) CNN, $N=400$ (c) CNN, $N=800$

(d) KXRCF, $N=200$ (e) KXRCF, $N=400$ (f) KXRCF, $N=800$

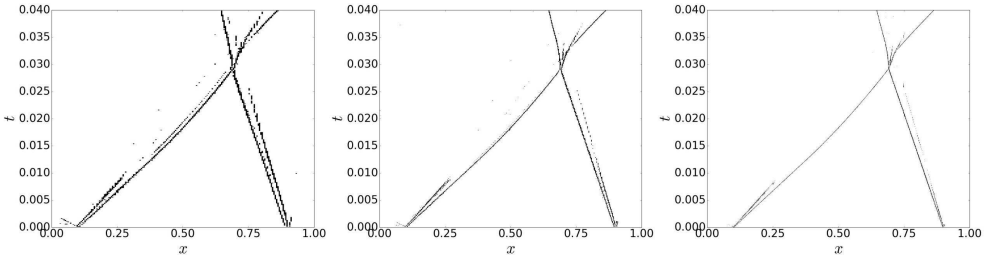
(g) MR, $N=200$ (h) MR, $N=400$ (i) MR, $N=800$

Figure 9: Signals of different troubled cell indicators for the interacting blast waves in Example 5.4. First row: CNN detector. Second row: KXRCF indicator. Third row: MR indicator. First column: $N=200$. Second column: $N=400$. Third column: $N=800$. Solid black lines: reference solution. Blue squares: numerical solution. Red circles with stems: indicator output. Black dashed lines: threshold for the indicator.

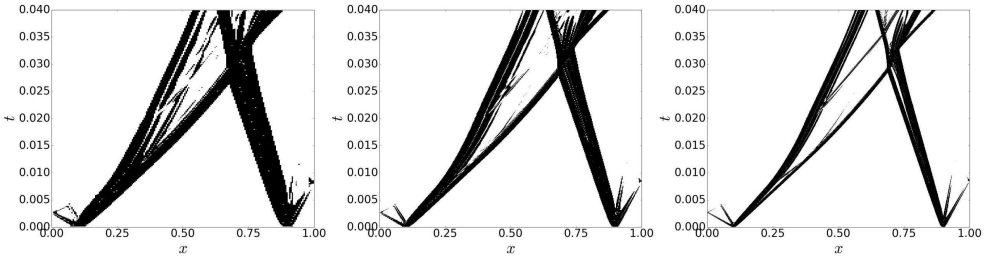
signals by KXRCF and MR indicators are much stronger than those in previous tests, and many cells are marked as troubled cells. At the same time, CNN seems to be able to produce clean signals. It marks much fewer cells (see Table 5) while maintaining acceptable profiles of numerical densities.

Example 5.5 (Shu–Osher problem)**.** This test was introduced in [31], describing a Mach 3 shock interacting with sine waves in density. The initial condition is set as

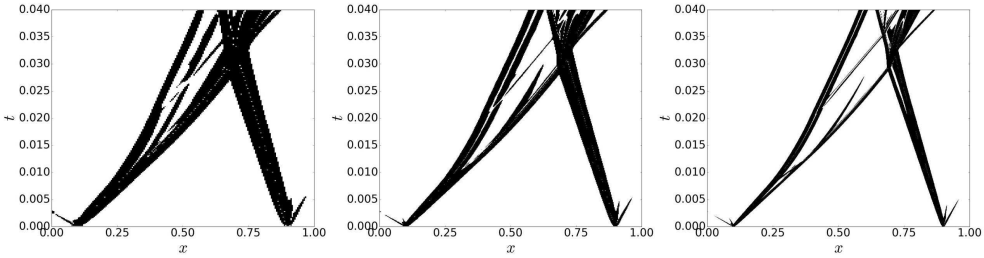
$$(\rho,\mu,p)=\begin{cases}(3.857143,2.629369,10.333333), & x\leq -4,\\ (1+0.2\sin(5x),0,1), & x>-4.\end{cases} \tag{5.6}$$



(a) CNN, $N=200$ (b) CNN, $N=400$ (c) CNN, $N=800$



(d) KXRCF, $N=200$ (e) KXRCF, $N=400$ (f) KXRCF, $N=800$



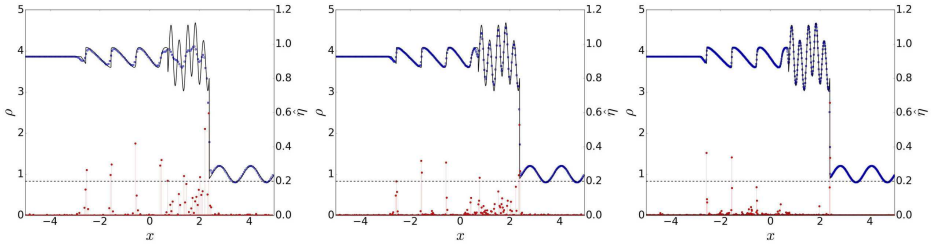
(g) MR, $N=200$ (h) MR, $N=400$ (i) MR, $N=800$

Figure 10: Troubled cell histories of different indicators for the interacting blast waves in Example 5.4. First row: CNN detector. Second row: KXRCF indicator. Third row: MR indicator. First column: $N=200$. Second column: $N=400$. Third column: $N=800$.

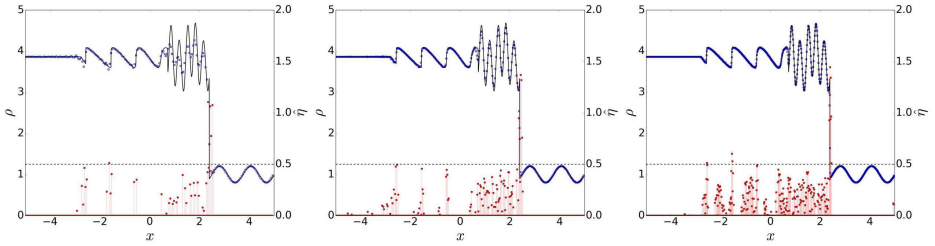
The density plots at $T=1.8$ on the domain $[-5,5]$, as well as the corresponding indicator outputs, are provided in Fig. 11. The troubled cell histories and the percentages are reported in Fig. 12 and Table 6, respectively. With a refined mesh, it seems that the CNN detector can better tell the high frequency waves in the region $[0,2]$ from the actual discontinuities. While the KXRCF and MR indicators consistently produce signals of medium strength in the high frequency region, and their inclusion of the high frequency region as troubled cells may depends heavily on the choice of thresholds.

5.3 Euler equations in two dimensions

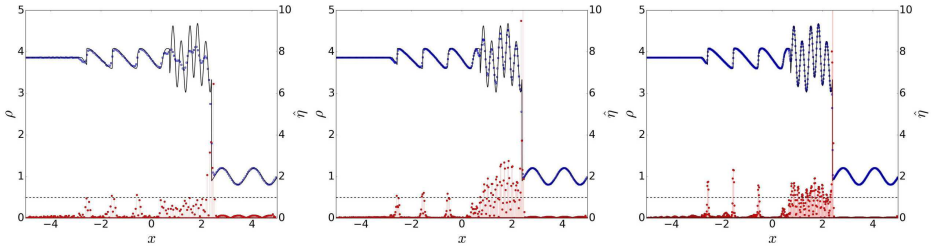
In this subsection, we consider Euler equations in two dimensions



(a) CNN, $N=200$ (b) CNN, $N=400$ (c) CNN, $N=800$



(d) KXRCF, $N=200$ (e) KXRCF, $N=400$ (f) KXRCF, $N=800$

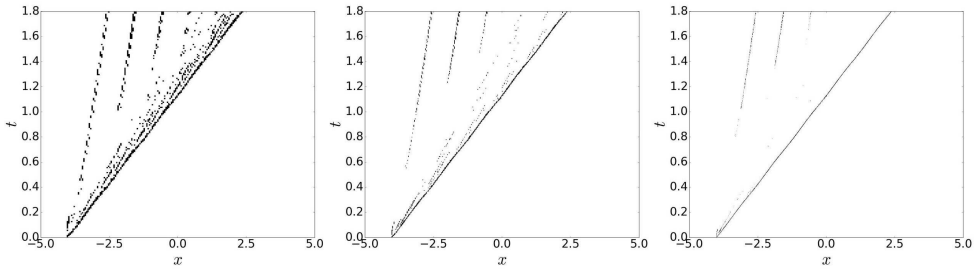


(g) MR, $N=200$ (h) MR, $N=400$ (i) MR, $N=800$

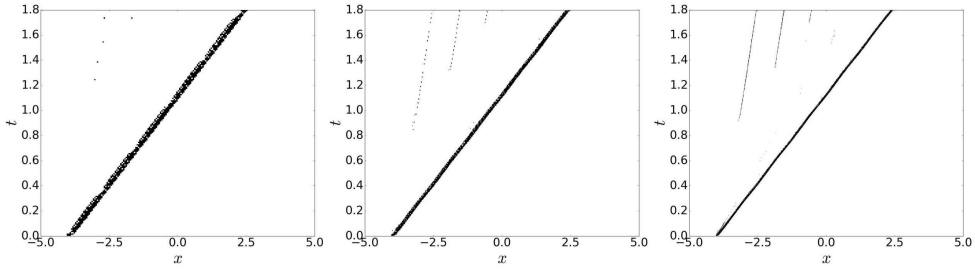
Figure 11: Signals of different troubled cell indicators for the Shu–Osher problem in Example 5.5. First row: CNN detector. Second row: KXRCF indicator. Third row: MR indicator. First column: $N=200$. Second column: $N=400$. Third column: $N=800$. Solid black lines: reference solution. Blue squares: numerical solution. Red circles with stems: indicator output. Black dashed lines: threshold for the indicator.

Table 6: Troubled cell percentages for the Shu–Osher problem in Example 5.5. See Table 4 for explanations of "Avg", "Max", "nbf" and "bf".

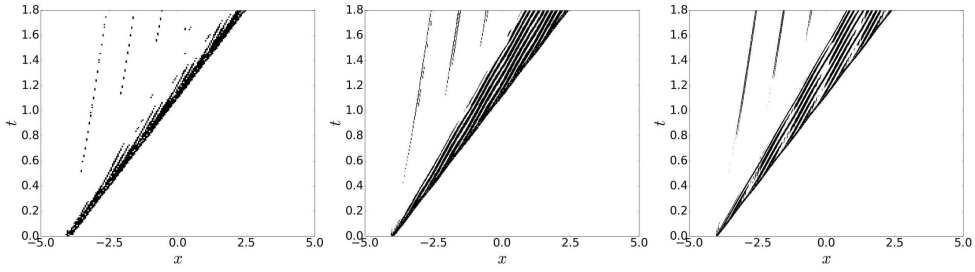
N	indicator	Avg(nbf)	Max(nbf)	Avg(bf)	Max(bf)
200	CNN	2.39	6.50	10.14	25.00
	KXRCF	2.29	3.00	5.13	10.50
	MR	3.55	6.50	8.04	20.00
400	CNN	0.79	2.00	3.60	9.50
	KXRCF	1.55	2.25	3.20	6.00
	MR	6.55	12.75	11.73	24.50
800	CNN	0.30	0.88	1.31	3.38
	KXRCF	1.02	1.50	2.20	4.00
	MR	5.29	10.62	9.64	19.38



(a) CNN, $N=200$ (b) CNN, $N=400$ (c) CNN, $N=800$



(d) KXRCF, $N=200$ (e) KXRCF, $N=400$ (f) KXRCF, $N=800$



(g) MR, $N=200$ (h) MR, $N=400$ (i) MR, $N=800$

Figure 12: Troubled cell histories of different indicators for the Shu–Osher problem in Example 5.5. First row: CNN detector. Second row: KXRCF indicator. Third row: MR indicator. First column: $N=200$. Second column: $N=400$. Third column: $N=800$.

$$\begin{pmatrix}\rho\\ \rho\mu\\ \rho\nu\\ E\end{pmatrix}_t+\begin{pmatrix}\rho\mu\\ \rho\mu^2+p\\ \rho\mu\nu\\ \mu(E+p)\end{pmatrix}_x+\begin{pmatrix}\rho\nu\\ \rho\mu\nu\\ \rho\nu^2+p\\ \nu(E+p)\end{pmatrix}_y=\begin{pmatrix}0\\0\\0\\0\end{pmatrix}, \tag{5.7}$$

where ρ is the density, (μ,ν) is the velocity vector, E is the total energy, and $p=(\gamma-1)(E-\frac{\rho}{2}(\mu^2+\nu^2))$, with $\gamma=1.4$, is the pressure. In all two-dimensional tests, the CFL number is chosen as 0.6. For the CNN detector, we choose the threshold to be 0.5 for the first step model and 0.2 for the second step model. The thresholds for the KXRCF indicator and MR indicator are both taken as 1. For a direct comparison with the CNN detector, we

consider tensor product implementations of the KXRCF and MR indicators, which are outlined in Appendix A.

Example 5.6 (Riemann problem I)**.** We start with a Riemann problem, and the initial data is set as (see, for example, [19])

$$(\rho,\mu,\nu,p)=\begin{cases}(1.5,0,0,1.5), & x>0.5,\ y>0.5,\\ (0.5323,1.206,0,0.3), & x<0.5,\ y>0.5,\\ (0.138,1.206,1.206,0.029), & x<0.5,\ y<0.5,\\ (0.5323,0,1.206,0.3), & x>0.5,\ y<0.5.\end{cases} \tag{5.8}$$

The numerical densities, the indicator outputs and the troubled cells at $T=0.3$ are reported in Fig. 13 when 200×200 mesh points are used, and in Fig. 14 when 400×400 mesh points are used. The percentages of troubled cells during the computation, when CNN, KXRCF and MR indicators are used, are summarized in Table 7. It can be seen that, the CNN detector itself marks fewer cells (about half of the number) compared with the KXRCF and MR indicators. While the percentages of buffer cells for the CNN detector are similar to those of the MR indicator. This is because a wider buffer zone is applied due to the way the troubled cells are labeled. See Appendix A for details. Also, the indicator outputs of KXRCF indicator seem to be much more noisy, when compared with those of the other indicators.

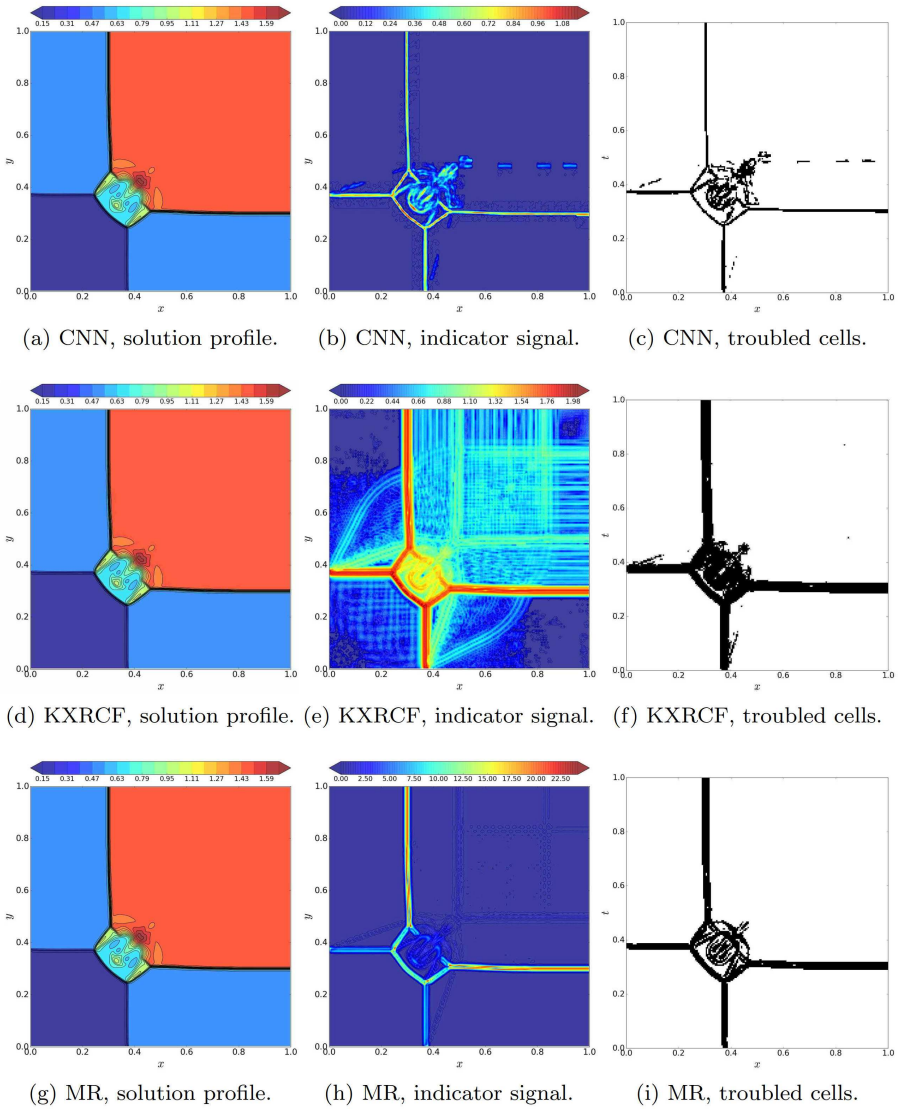
Table 7: Troubled cell percentages for the Riemann problem in Example 5.6. See Table 4 for explanations of "Avg", "Max", "nbf" and "bf".

$N^x\times N^y$	indicator	Avg(nbf)	Max(nbf)	Avg(bf)	Max(bf)
200×200	CNN	2.70	4.36	9.94	14.46
	KXRCF	8.32	9.86	14.30	17.29
	MR	5.64	7.04	10.96	13.19
400×400	CNN	1.46	2.17	5.76	7.98
	KXRCF	6.42	8.18	11.29	14.25
	MR	3.70	4.75	7.01	9.35

Example 5.7 (Riemann problem II)**.** Next, we consider the Riemann problem that is initially set as (see, for example, [19])

$$(\rho,\mu,\nu,p)=\begin{cases}(1.1,0,0,1.1), & x>0.5,\ y>0.5,\\ (0.5065,0.8939,0,0.35), & x<0.5,\ y>0.5,\\ (1.1,0.8939,0.8939,1.1), & x<0.5,\ y<0.5,\\ (0.5065,0,0.8939,0.35), & x>0.5,\ y<0.5.\end{cases} \tag{5.9}$$

The solutions, indicator outputs and the marked cells at $T=0.2$ are given in Fig. 15. (We provide the plots with 400×400 mesh points only, and those with 200×200 points are



(a) CNN, solution profile. (b) CNN, indicator signal. (c) CNN, troubled cells.

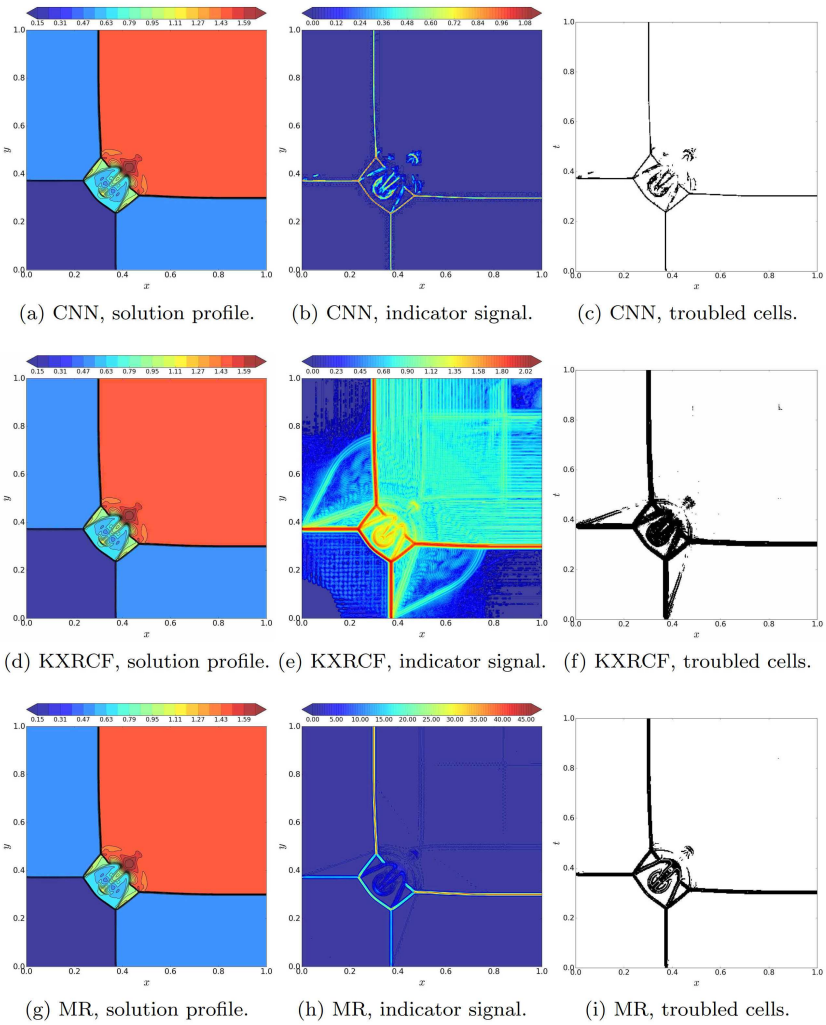
(d) KXRCF, solution profile. (e) KXRCF, indicator signal. (f) KXRCF, troubled cells.

(g) MR, solution profile. (h) MR, indicator signal. (i) MR, troubled cells.

Figure 13: Numerical outputs for Riemann problem in Example 5.6 with 200×200 mesh cells at $T=0.3$. First row: CNN detector. Second row: KXRCF indicator. Third row: MR indicator. First column: numerical density, with 20 equally spaced contour lines ranging from 0.15 to 1.75. Second column: indicator signal. Third column: marked troubled cells.

.

omitted to save space.) Percentages of troubled cells are summarized in Table 8. Again, we can observe that the CNN detector gives sharp and concentrated signals. Compared with the KXRCF and MR indicators, it marks fewer number of cells if the buffer zone is not considered.



(a) CNN, solution profile. (b) CNN, indicator signal. (c) CNN, troubled cells.

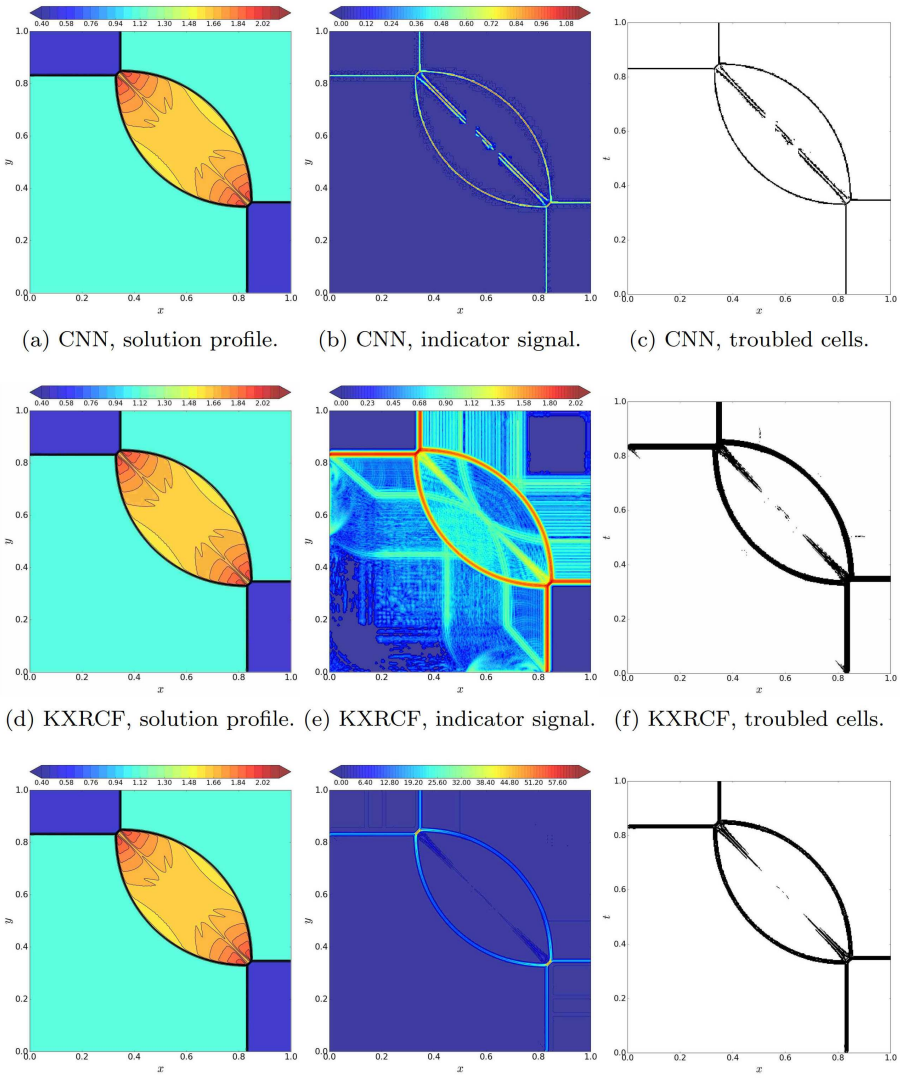
(d) KXRCF, solution profile. (e) KXRCF, indicator signal. (f) KXRCF, troubled cells.

(g) MR, solution profile. (h) MR, indicator signal. (i) MR, troubled cells.

Figure 14: Numerical outputs for Riemann problem in Example 5.6 with 400×400 mesh cells at $T=0.3$. First row: CNN detector. Second row: KXRCF indicator. Third row: MR indicator. First column: numerical density, with 20 equally spaced contour lines ranging from 0.15 to 1.75. Second column: indicator signal. Third column: marked troubled cells.

Table 8: Troubled cell percentages for the Riemann problem in Example 5.7. See Table 4 for explanations of "Avg", "Max", "nbf" and "bf".

$N^x\times N^y$	indicator	Avg(nbf)	Max(nbf)	Avg(bf)	Max(bf)
200×200	CNN	3.26	4.95	12.55	17.77
	KXRCF	8.19	9.40	14.69	18.13
	MR	5.98	7.35	12.39	15.49
400×400	CNN	1.64	2.43	6.52	8.64
	KXRCF	5.53	6.45	9.73	11.45
	MR	3.82	4.83	7.40	9.55



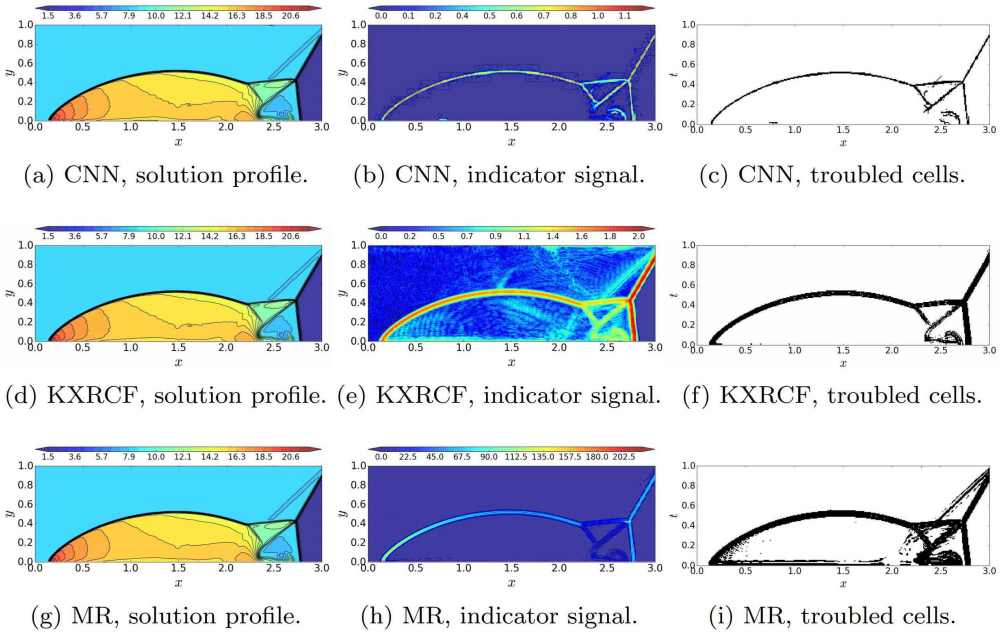
(a) CNN, solution profile. (b) CNN, indicator signal. (c) CNN, troubled cells.

(d) KXRCF, solution profile. (e) KXRCF, indicator signal. (f) KXRCF, troubled cells.

(g) MR, solution profile. (h) MR, indicator signal. (i) MR, troubled cells.

Figure 15: Numerical outputs for Riemann problem in Example 5.7 with 400×400 mesh cells at $T=0.2$. First row: CNN detector. Second row: KXRCF indicator. Third row: MR indicator. First column: numerical density, with 30 equally spaced contour lines ranging from 0.4 to 2.2. Second column: indicator signal. Third column: marked troubled cells.

Example 5.8 (Double Mach reflection)**.** This problem describes reflections of planar shocks in air from wedges [36]. The computational domain of this problem is set as $[0,4]\times[0,1]$. Initially, a right-moving Mach 10 shock is positioned at $x=1/6$ and makes a 60^o angle with the x-axis. It moves into the undisturbed air with a density of 1.4 and a pressure of 1. At the bottom boundary, the exact post-shock condition is imposed from $x=0$ to $x=1/6$ and a reflective boundary condition is used in the rest part for the rigid wall.



(a) CNN, solution profile. (b) CNN, indicator signal. (c) CNN, troubled cells.

(d) KXRCF, solution profile. (e) KXRCF, indicator signal. (f) KXRCF, troubled cells.

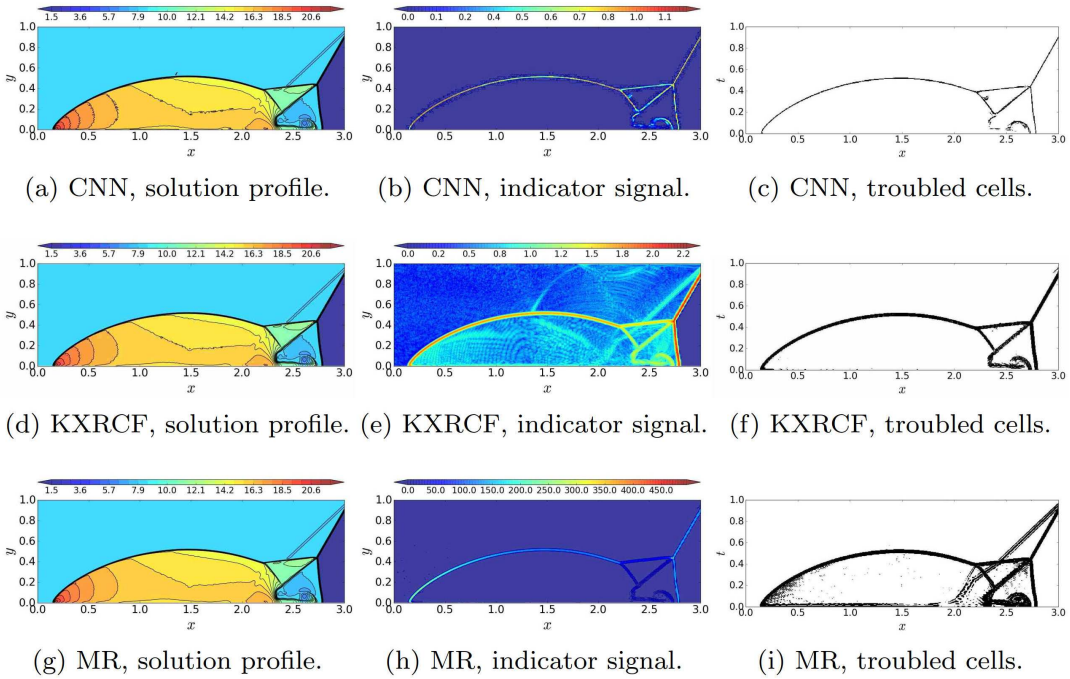
(g) MR, solution profile. (h) MR, indicator signal. (i) MR, troubled cells.

Figure 16: Numerical outputs for double Mach reflection in Example 5.8 with 480×120 mesh cells at $T=0.2$. First row: CNN detector. Second row: KXRCF indicator. Third row: MR indicator. First column: numerical density, with 30 equally spaced contour lines ranging from 1.5 to 22.7. Second column: indicator signal. Third column: marked troubled cells.

Table 9: Troubled cell percentages for the double Mach reflection in Example 5.8. See Table 4 for explanations of "Avg", "Max", "nbf" and "bf".

$N^x\times N^y$	indicator	Avg(nbf)	Max(nbf)	Avg(bf)	Max(bf)
480×120	CNN	1.57	2.57	6.71	10.78
	KXRCF	4.06	6.30	7.94	12.93
	MR	7.04	11.64	13.05	21.45
960×240	CNN	0.80	1.32	3.45	5.48
	KXRCF	2.71	4.31	5.08	8.16
	MR	4.97	8.11	9.52	15.86

At the top boundary, the flow values are set to describe the exact motion of the Mach 10 shock. We compute the solution up to $T=0.2$. Solution profiles of numerical densities, indicator outputs and marked cells are given in Fig. 16 and Fig. 17. Percentages of troubled cells are summarized in Table 9. In this test, it seems that the MR indicator produces very strong signals, and marks a lot more cells than the other two indicators, if we maintain the threshold to be 1. For the CNN and KXRCF indicators, troubled cells marked by the CNN detector seem to be fewer than half of the KXRCF indicator before applying the buffer zone. With the buffer zone, this difference is reduced, but the CNN detector still marks fewer cells. Again, the indicator outputs of KXRCF indicator seem to be much more noisy, when compared with those of the CNN detector.



(a) CNN, solution profile. (b) CNN, indicator signal. (c) CNN, troubled cells.

(d) KXRCF, solution profile. (e) KXRCF, indicator signal. (f) KXRCF, troubled cells.

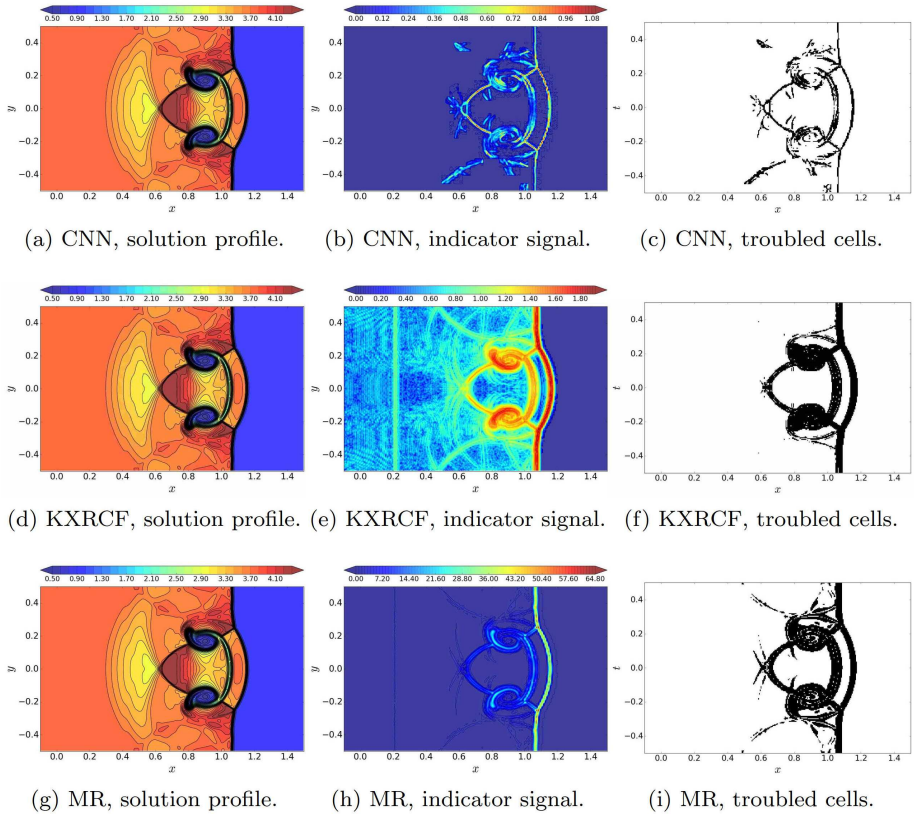
(g) MR, solution profile. (h) MR, indicator signal. (i) MR, troubled cells.

Figure 17: Numerical outputs for double Mach reflection in Example 5.8 with 960×240 mesh cells at $T=0.2$. First row: CNN detector. Second row: KXRCF indicator. Third row: MR indicator. First column: numerical density, with 30 equally spaced contour lines ranging from 1.5 to 22.7. Second column: indicator signal. Third column: marked troubled cells.

Example 5.9 (Bubble-shock interaction)**.** This last example simulates the interaction between a planar shock wave and a circular region of low density [13, 18]. The computational domain is set as $[-0.1,1.5]\times[-0.5,0.5]$. An incoming Mach 2.95 shock is initially positioned at $x=0$, and move rightward into the undisturbed gas with unit density and pressure. A circular bubble is centered at $(0.3,0)$ with radius 0.2, inside of which the density is 0.1. Solution profiles of numerical densities, indicator outputs and marked cells are given in Fig. 16 and Fig. 17. The percentages of troubled cells are summarized in Table

Table 10: Troubled cell percentages for the bubble-shock interaction in Example 5.9. See Table 4 for explanations of "Avg", "Max", "nbf" and "bf".

$N^x\times N^y$	indicator	Avg(nbf)	Max(nbf)	Avg(bf)	Max(bf)
320×200	CNN	3.11	5.47	11.21	20.49
	KXRCF	7.92	10.25	12.47	17.35
	MR	8.28	12.25	14.23	22.18
640×400	CNN	1.90	3.39	7.07	12.70
	KXRCF	6.51	9.20	10.44	15.68
	MR	6.77	11.26	11.72	20.25



(a) CNN, solution profile. (b) CNN, indicator signal. (c) CNN, troubled cells.

(d) KXRCF, solution profile. (e) KXRCF, indicator signal. (f) KXRCF, troubled cells.

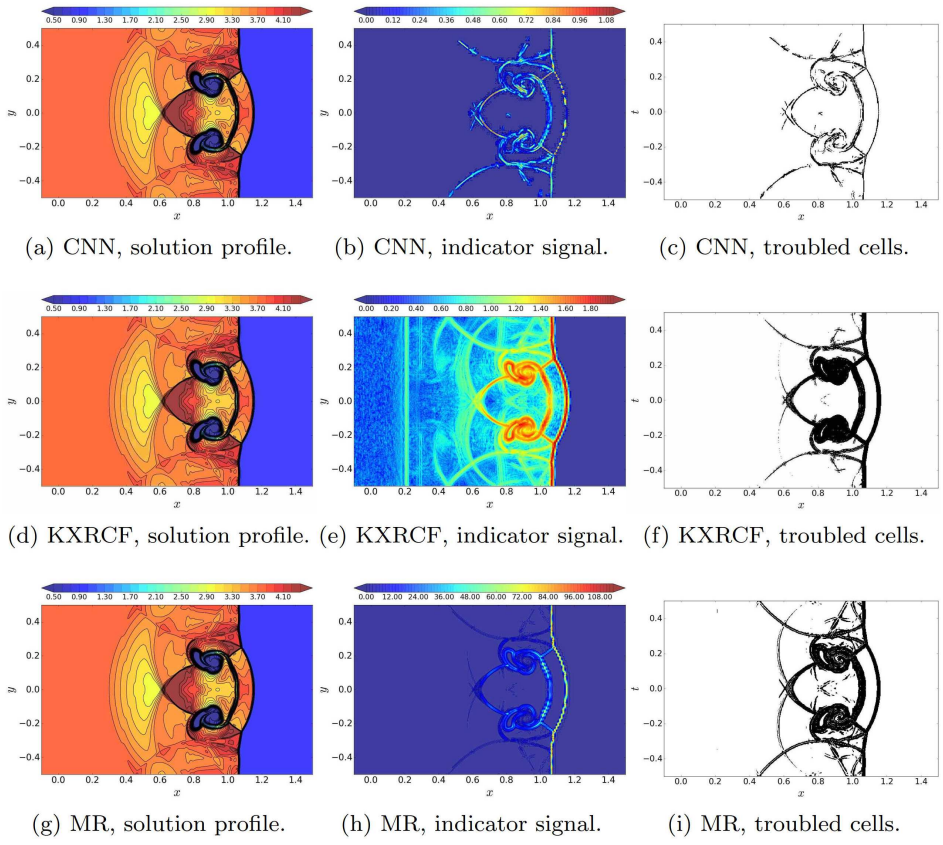
(g) MR, solution profile. (h) MR, indicator signal. (i) MR, troubled cells.

Figure 18: Numerical outputs for bubble-shock interaction in Example 5.9 with 320×200 mesh cells at $T=0.3$. First row: CNN detector. Second row: KXRCF indicator. Third row: MR indicator. First column: numerical density, with 30 equally spaced contour lines ranging from 0.5 to 4.5. Second column: indicator signal. Third column: marked troubled cells.

10. From the pictures and the table, it can be seen that the CNN detector is capturing fewer troubled cells, while maintaining acceptable solution profiles.

6 Conclusion

We proposed a troubled cell detector using CNNs. The CNN can be trained offline and become increasingly more accurate as one keeps training it. A well trained CNN detector can then be used in online computations of conservation laws, where discontinuities often occur. In this paper, we demonstrate the effectiveness of the approach, where fifth-order WENO is used on troubled cells and sixth-order central difference in smooth regions. The computational results are encouraging, as we obtained cleaner and sharper resolutions in the results, when compared to other existing methods. More in-depth study of the method, such as its computational efficiency and coupling of other numerical schemes, will follow in future studies.



(a) CNN, solution profile. (b) CNN, indicator signal. (c) CNN, troubled cells.

(d) KXRCF, solution profile. (e) KXRCF, indicator signal. (f) KXRCF, troubled cells.

(g) MR, solution profile. (h) MR, indicator signal. (i) MR, troubled cells.

Figure 19: Numerical outputs for bubble-shock interaction in Example 5.9 with 640×400 mesh cells at $T=0.3$. First row: CNN detector. Second row: KXRCF indicator. Third row: MR indicator. First column: numerical density, with 30 equally spaced contour lines ranging from 0.5 to 4.5. Second column: indicator signal. Third column: marked troubled cells.

Acknowledgments

The work of Y. Xing was partially supported by the NSF grant DMS-1753581. The work of D. Xiu was partially supported by AFOSR FA9550-18-1-0102.

A KXRCF and MR indicators in two dimensions

Again in this section, we drop the subscript h and use u for numerical solutions. Let $I_{i,j}$ be the mesh cell centered at (x_i,y_j). We assume $h_x=x_{i+\frac{1}{2}}-x_{i-\frac{1}{2}}=h_y=y_{j+\frac{1}{2}}-y_{j-\frac{1}{2}}=h$.

KXRCF Indicator. For each cell $I_{i,j}$, we first use the nine nodal values $u_{i+i',j+j'}$, $-1\leq i',j'\leq 1$ to find a Q^2 interpolating polynomial, denoted by v. For Euler equations, we use the

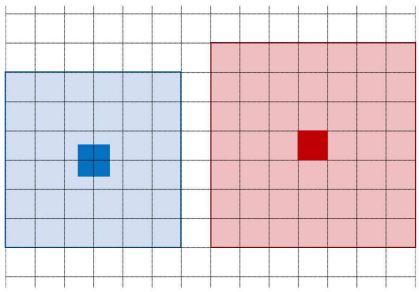


Figure 20: Troubled cell and buffer zones of indicators. The crosses of lines corresponds to grid points (x_i,y_j). Red cell: troubled cell marked by the CNN detector. Red region: buffer zone for the CNN detector. Blue cell: troubled cell marked by the KXRCF/MR indicator. Blue region: buffer zone for the KXRCF/MR indicator.

velocity (μ,ν) to determine the flow direction in the cell $I_{i,j}$. Then define

$$\kappa_{i,j}^{\text{KXRCF}}=\frac{\left|\int_{\partial I_{i,j}^-}\left(v|_{I_{i,j}}-v|_{I_{\text{nb},i,j}}\right)ds\right|}{h^{\frac{3}{2}}|\partial I_{i,j}^-|\|v\|_{I_{i,j}}}, \tag{A.1}$$

where $\partial I_{i,j}^-$ is the inflow portion of the cell boundary, $I_{\text{nb},i,j}$ is the neighboring cell sharing the edge $\partial I_{i,j}^-$. The output of KXRCF indicator on $I_{i,j}$ is set as

$$\eta_{i,j}^{\text{KXRCF}}=-\frac{\log\kappa_{i,j}^{\text{KXRCF}}}{\log h}. \tag{A.2}$$

MR Indicator. For the mesh cell $I_{i,j}$, we define

$$\kappa_{i,j}^{\text{MR}}=\left(\frac{1}{9}\sum_{i',j'=-1}^{1}u_{i+i',j+j'}\right)-u_{i,j}. \tag{A.3}$$

The output of MR indicator on $I_{i+\frac{1}{2},j+\frac{1}{2}}$ is set as

$$\eta_{i,j}^{\text{MR}}=\frac{\kappa_{i,j}^{\text{MR}}}{h}. \tag{A.4}$$

Buffer Zone. We remark that the signal of traditional indicators are given at a cell centered at grid points, while that of CNN detector is given at the actual mesh cell. In the case that closed cells are considered, the buffer zone of the CNN detector is slightly wider than the KXRCF and MR indicators. See Fig. 20. We note that this is not an issue in one dimension, since the discontinuity is marked on a half-closed interval.

References

[1] D. S. Balsara and C.-W. Shu, *Monotonicity preserving weighted essentially non-oscillatory schemes with increasingly high order of accuracy*, Journal of Computational Physics, 160 (2000), pp. 405–452.

[2] R. Biswas, K. D. Devine, and J. E. Flaherty, *Parallel, adaptive finite element methods for conservation laws*, Applied Numerical Mathematics, 14 (1994), pp. 255–283.

[3] A. Burbeau, P. Sagaut, and C.-H. Bruneau, *A problem-independent limiter for high-order Runge–Kutta discontinuous Galerkin methods*, Journal of Computational Physics, 169 (2001), pp. 111–150.

[4] B. Cockburn and C.-W. Shu, *TVB Runge–Kutta local projection discontinuous Galerkin finite element method for conservation laws. II. General framework*, Mathematics of Computation, 52 (1989), pp. 411–435.

[5] B. Costa and W. S. Don, *High order hybrid central-WENO finite difference scheme for conservation laws*, Journal of Computational and Applied Mathematics, 204 (2007), pp. 209–218.

[6] W.-S. Don, Z. Gao, P. Li, and X. Wen, *Hybrid compact-WENO finite difference scheme with conjugate Fourier shock detection algorithm for hyperbolic conservation laws*, SIAM Journal on Scientific Computing, 38 (2016), pp. A691–A711.

[7] Y. Feng, T. Liu, and K. Wang, *A characteristic-featured shock wave indicator for conservation laws based on training an artificial neuron*, Journal of Scientific Computing, 83 (2020), p. 21.

[8] G. Fu and C.-W. Shu, *A new troubled-cell indicator for discontinuous Galerkin methods for hyperbolic conservation laws*, Journal of Computational Physics, 347 (2017), pp. 305–327.

[9] Z. Gao and W. S. Don, *Mapped hybrid central-WENO finite difference scheme for detonation waves simulations*, Journal of Scientific Computing, 55 (2013), pp. 351–371.

[10] A. Harten, *Adaptive multiresolution schemes for shock computations*, Journal of Computational Physics, 115 (1994), pp. 319–338.

[11] A. Harten, *Multiresolution algorithms for the numerical solution of hyperbolic conservation laws*, Communications on Pure and Applied Mathematics, 48 (1995), pp. 1305–1342.

[12] A. Harten, *Multiresolution representation of data: A general framework*, SIAM Journal on Numerical Analysis, 33 (1996), pp. 1205–1256.

[13] H. Holden, K.-A. Lie, and N. H. Risebro, *An unconditionally stable method for the Euler equations*, Journal of Computational Physics, 150 (1999), pp. 76–96.

[14] G.-S. Jiang and C.-W. Shu, *Efficient implementation of weighted ENO schemes*, Journal of Computational Physics, 126 (1996), pp. 202–228.

[15] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, arXiv preprint arXiv:1412.6980, (2014).

[16] L. Krivodonova, *Limiters for high-order discontinuous Galerkin methods*, Journal of Computational Physics, 226 (2007), pp. 879–896.

[17] L. Krivodonova, J. Xin, J.-F. Remacle, N. Chevaugeon, and J. E. Flaherty, *Shock detection and limiting with discontinuous Galerkin methods for hyperbolic conservation laws*, Applied Numerical Mathematics, 48 (2004), pp. 323–338.

[18] J. O. Langseth and R. J. LeVeque, *A wave propagation method for three-dimensional hyperbolic conservation laws*, Journal of Computational Physics, 165 (2000), pp. 126–166.

[19] P. D. Lax and X.-D. Liu, *Solution of two-dimensional Riemann problems of gas dynamics by positive schemes*, SIAM Journal on Scientific Computing, 19 (1998), pp. 319–340.

[20] G. Li, C. Lu, and J. Qiu, *Hybrid well-balanced WENO schemes with different indicators for shallow water equations*, Journal of Scientific Computing, 51 (2012), pp. 527–559.

[21] G. Li and J. Qiu, *Hybrid weighted essentially non-oscillatory schemes with different indicators*, Journal of Computational Physics, 229 (2010), pp. 8105–8129.
[22] G. Li and J. Qiu, *Hybrid WENO schemes with different indicators on curvilinear grids*, Advances in Computational Mathematics, 40 (2014), pp. 747–772.
[23] S. Pirozzoli, *Conservative hybrid compact-WENO schemes for shock-turbulence interaction*, Journal of Computational Physics, 178 (2002), pp. 81–117.
[24] J. Qiu and C.-W. Shu, *A comparison of troubled-cell indicators for Runge–Kutta discontinuous Galerkin methods using weighted essentially nonoscillatory limiters*, SIAM Journal on Scientific Computing, 27 (2005), pp. 995–1013.
[25] D. Ray and J. S. Hesthaven, *An artificial neural network as a troubled-cell indicator*, Journal of Computational Physics, 367 (2018), pp. 166–191.
[26] D. Ray and J. S. Hesthaven, *Detecting troubled-cells on two-dimensional unstructured grids using a neural network*, Journal of Computational Physics, 397 (2019), p. 108845.
[27] W. J. Rider and L. G. Margolin, *Simple modifications of monotonicity-preserving limiter*, Journal of Computational Physics, 174 (2001), pp. 473–488.
[28] K. Shahbazi, J. S. Hesthaven, and X. Zhu, *Multi-dimensional hybrid Fourier continuation–WENO solvers for conservation laws*, Journal of Computational Physics, 253 (2013), pp. 209–225.
[29] C.-W. Shu, *Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws*, in Advanced Numerical Approximation of Nonlinear Hyperbolic Equations, Springer, 1998, pp. 325–432.
[30] C.-W. Shu, *High order weighted essentially nonoscillatory schemes for convection dominated problems*, SIAM Review, 51 (2009), pp. 82–126.
[31] C.-W. Shu and S. Osher, *Efficient implementation of essentially non-oscillatory shock-capturing schemes, II*, in Upwind and High-Resolution Schemes, Springer, 1989, pp. 328–374.
[32] A. Suresh, H. Huynth, A. Suresh, and H. Huynth, *Accurate monotonicity-preserving schemes with Runge–Kutta time stepping*, in 13th Computational Fluid Dynamics Conference, 1997, p. 2037.
[33] M. H. Veiga and R. Abgrall, *Neural network based limiter with transfer learning*, Communications on Applied Mathematics and Computation, (2020), https://doi.org/10.1007/s42967-020-00087-1.
[34] S. Wang, Z. Zhou, L.-B. Chang, and D. Xiu, *Construction of a universal discontinuity detector using convolution neural networks*, submitted for publication, (2020).
[35] X. Wen, W. S. Don, Z. Gao, and J. S. Hesthaven, *An edge detector based on artificial neural network with application to hybrid compact-WENO finite difference scheme*, Journal of Scientific Computing, 83 (2020), p. 49.
[36] P. Woodward and P. Colella, *The numerical simulation of two-dimensional fluid flow with strong shocks*, Journal of Computational Physics, 54 (1984), pp. 115–173.
[37] Z. Xu and C.-W. Shu, *Anti-diffusive flux corrections for high order finite difference WENO schemes*, Journal of Computational Physics, 205 (2005), pp. 458–485.