Bergische Universität Wuppertal

Fakultät für Mathematik und Naturwissenschaften

Institute of Mathematical Modelling, Analysis and Computational Mathematics (IMACM)

Preprint BUW-IMACM 23/14

Tatiana Kossaczká, Ameya D. Jagtap and Matthias Ehrhardt

Deep smoothness WENO scheme for two-dimensional hyperbolic conservation laws: A deep learning approach for learning smoothness indicators

September 18, 2023

http://www.imacm.uni-wuppertal.de

Deep smoothness WENO scheme for two-dimensional hyperbolic conservation laws: A deep learning approach for learning smoothness indicators

Tatiana Kossaczká[a,*], Ameya D. Jagtap[b], Matthias Ehrhardt[a]

[a]*Chair of Applied and Computational Mathematics, Bergische Universität Wuppertal, Gaußstrasse 20, Wuppertal, 42119, Germany*
[b]*Division of Applied Mathematics, Brown University, 182 George Street, Providence, RI 02912, USA*

Abstract

In this paper we introduce an improved version of the fifth-order weighted essentially non-oscillatory (WENO) shock-capturing scheme by incorporating deep learning techniques. The established WENO algorithm is improved by training a compact neural network to adjust the smoothness indicators within the WENO scheme. This modification enhances the accuracy of the numerical results, particularly near abrupt shocks. Unlike previous deep learning based methods, no additional post-processing steps are necessary for maintaining the consistency. We demonstrate the superiority of our new approach using several examples from the literature for the two-dimensional Euler equations of gas dynamics. Through intensive study of these test problems, which involve various shocks and rarefaction waves, the new technique is shown to outperform traditional fifth-order WENO schemes, especially in cases where the numerical solutions exhibit excessive diffusion or overshoot around shocks.

Keywords: Weighted essentially non-oscillatory method, Hyperbolic conservation laws, Smoothness indicators, Deep Learning, Euler equations
2020 MSC: 65M06, 68T05, 76M20

1. Introduction

It has long been a challenge to adequately simulate complex flow problems using numerical methods. Recently, this has been further improved using machine learning techniques. As an example, in [1, 2, 3], the concept of physics-informed neural networks (PINNs) for the solution of complex fluid flow problems was proposed, which seamlessly combines the

*corresponding author
Email addresses: kossaczka@uni-wuppertal.de (Tatiana Kossaczká), ameya_jagtap@brown.edu (Ameya D. Jagtap), ehrhardt@uni-wuppertal.de (Matthias Ehrhardt)

data and the mathematical models; see [1, 4, 5, 6, 7, 8] for more details. Similarly, a new method using a U-Net-like convolutional neural network (CNN) along with established finite difference discretization techniques was proposed to learn approximate solutions for the NSE without the need for parameterization [9]. Also, recently, a framework called *local transfer function analysis* (LTA) for optimizing numerical methods for convection problems using a graph neural network (GNN) was proposed [10].

The work [11] investigated the use of PINNs to approximate the hyperbolic Euler equations of gas dynamics. The Euler equations and initial and boundary conditions are used to create a loss function that solves scenarios with smooth solutions and those with discontinuities. Next, in [4], a novel approach, called *conservative PINNs*, for solving nonlinear conservation laws, such as the compressible Euler equations, was presented. In the recent paper [12], another novel approach has been proposed where machine learning improves finite-difference-based approximations of PDEs while maintaining high-order convergence through node refinement.

This research area is also the context of our work. Recently, improvements to the standard finite difference methods (FDMs) have been developed [13]. By adding a small convolutional neural network, the solutions of the standard PDEs are improved, while the convergence and consistency properties of the original methods are preserved. We aim to further improve modern FDMs, such as WENO schemes, for nonlinear hyperbolic systems using machine learning. For this type of PDEs, it is known that discontinuities (shocks) can occur despite initial smoothness, which makes specialized numerical methods mandatory. Therefore, the focus of our attention is on the behavior of numerical solutions in the vicinity of shocks.

To better frame our current work, let us very briefly sketch the historical development of WENO schemes. Crandall and Majda [14] introduced *monotone schemes* in 1980 that maintain stability and satisfy entropy conditions, but are only exactly first order due to Godunov's theorem. Next, *shock-capturing schemes* were developed to accurately handle shocks and gradients without excessive diffusion [15]. The *essentially non-oscillatory (ENO) schemes* [16] were outstanding, achieving high accuracy in smooth regions and effective shock resolution using smoothness indicators, e.g. [17, 18]. Extensions such as the *Hermite WENO (HWENO)* schemes [19, 20] and *hybrid methods* [21, 22] were introduced for higher accuracy and efficiency. A gas-kinetic theory based KWENO scheme was proposed in [23] for hyperbolic conservation laws. Moreover, further modifications of WENO scheme have been developed, e.g. [24, 25, 26, 27, 28, 29].

And then machine learning for solving PDEs entered the scene. Neural networks approximated the solutions of PDEs and improved numerical methods for PDEs. While the data-driven approach is promising for improving modern numerical methods, it is always important to maintain a balance between new data-driven insights and established mathematical structures, i.e., the basic numerical scheme (here based on physical principles), e.g., for hyperbolic problems, the resulting hybrid scheme should be conservative in any case. We have maintained this balance, and next we will briefly describe our approach.

Recent approaches to solving numerical PDEs include neural network-based WENO methods that modify coefficients and smoothness indicators of established state-of-the-art numerical methods to further improve these schemes, especially near shocks. However, some methods achieve only first-order accuracy [30].

In this paper, we present a new approach called "WENO-DS", a Deep learning-based extension of the family of WENO methods and extend it to solving a general two-dimensional system of hyperbolic conservation laws

$$U_t + F(U)_x + G(U)_y = 0. \tag{1}$$

To this end, we modify the smoothness indicators of the WENO schemes using a small neural network, maintaining high accuracy in smooth regions and reducing diffusion and overshoots (oscillatory behaviour) near shocks. The resulting machine learning enhanced WENO scheme combines accuracy and improved qualitative behavior for both smooth and discontinuous solutions.

The paper is organized as follows. In Section 2, we introduce two underlying WENO schemes and explain the basic ideas, such as the smoothness indicators, on a 1D conservation law. In Section 3, we present our method for improving these schemes using a deep learning approach to modify the smoothness indicators accordingly. This novel idea does not destroy the basic structure of the WENO schemes, such as the conservative property, and qualitatively improves the solution near shocks with only small additional computational costs. In this section, we also elaborate on implementation aspects, such as adaptive activation functions and the design of the small network, and the training procedure. In Section 4, we briefly describe our application example of the 2D Euler equations of gas dynamics. Subsequently, in Section 5 we present in detail the numerical results with a wide range of test configurations. Finally, in Section 6 we conclude our work and give a brief overview of future research directions.

2. The WENO scheme

We first introduce the standard fifth-order WENO scheme for solving one-dimensional hyperbolic conservation laws

$$u_t + f(u)_x = 0, \tag{2}$$

as developed by Jiang and Shu [17, 18]. For this purpose, we consider the uniform grid defined by the points $x_i = x_0 + i\Delta x$ with cell boundaries $x_{i+\frac{1}{2}} = x_i + \frac{\Delta x}{2}$, $i = 0, \ldots, I$. The semi-discrete formulation of (2) can be written as

$$\frac{\mathrm{d}u_i(t)}{\mathrm{d}t} = -\frac{1}{\Delta x}\big(\hat{f}_{i+\frac{1}{2}} - \hat{f}_{i-\frac{1}{2}}\big), \tag{3}$$

where $u_i(t)$ approximates $u(x_i, t)$ pointwise and $\hat{f}$ is a numerical approximation of the flux function f, i.e. $\hat{f}_{i+\frac{1}{2}}$ and $\hat{f}_{i-\frac{1}{2}}$ are numerical flux approximations at the cell boundaries $x_{i+\frac{1}{2}}$

and $x_{i-\frac{1}{2}}$, respectively. The numerical flux $\hat{f}_{i+\frac{1}{2}}$ is chosen such that for all sufficiently smooth u

$$\frac{1}{\Delta x}\Big(\hat{f}_{i+\frac{1}{2}} - \hat{f}_{i-\frac{1}{2}}\Big) = \big(f(u)\big)_x\big|_{x=x_i} + O(\Delta x^5), \tag{4}$$

with fifth-order of accuracy. Defining a function h implicitly by

$$f\big(u(x)\big) = \frac{1}{\Delta x}\int_{x-\frac{\Delta x}{2}}^{x+\frac{\Delta x}{2}} h(\xi)\,d\xi, \tag{5}$$

we obtain

$$f'\big(u(x_i)\big) = \frac{1}{\Delta x}\big(h_{i+\frac{1}{2}} - h_{i-\frac{1}{2}}\big), \qquad h_{i\pm\frac{1}{2}} = h(x_{i\pm\frac{1}{2}}), \tag{6}$$

where $h_{i\pm\frac{1}{2}}$ approximates the numerical flux $\hat{f}_{\pm\frac{1}{2}}$ with the fifth-order of accuracy in a sense that

$$\hat{f}_{i\pm\frac{1}{2}} = h_{i\pm\frac{1}{2}} + O(\Delta x^5). \tag{7}$$

This procedure results in a *conservative* numerical scheme.

To ensure numerical stability, the *flux splitting method* is applied. We therefore write the flux in the form

$$f(u) = f^+(u) + f^-(u), \quad \text{where} \quad \frac{\mathrm{d}f^+(u)}{\mathrm{d}u} \geq 0 \quad \text{and} \quad \frac{\mathrm{d}f^-(u)}{\mathrm{d}u} \leq 0. \tag{8}$$

The numerical flux $\hat{f}_{i\pm\frac{1}{2}}$ is then given by $\hat{f}_{i\pm\frac{1}{2}} = \hat{f}^+_{i\pm\frac{1}{2}} + \hat{f}^-_{i\pm\frac{1}{2}}$ and we get the final approximation

$$\frac{\mathrm{d}u_i}{\mathrm{d}t} = -\frac{1}{\Delta x}\left[\Big(\hat{f}^+_{i+\frac{1}{2}} - \hat{f}^+_{i-\frac{1}{2}}\Big) + \Big(\hat{f}^-_{i+\frac{1}{2}} - \hat{f}^-_{i-\frac{1}{2}}\Big)\right]. \tag{9}$$

Remark 1. *In our implementation we use the Lax-Friedrichs flux splitting*

$$f^\pm(u) = \frac{1}{2}\big(f(u) \pm \alpha u\big), \tag{10}$$

with $\alpha = \max_u |f'(u)|$.

2.1. The fifth order WENO scheme

First, we consider the construction of $\hat{f}^+_{i+\frac{1}{2}}$ and drop the superscript $^+$ for simplicity. For this approximation a 5-point stencil

$$S(i) = \{x_{i-2}, \ldots, x_{i+2}\} \tag{11}$$

is used. The main idea of fifth-order WENO scheme is to divide this stencil (11) into three candidate substencils, which are given by

$$S^m(i) = \{x_{i+m-2}, x_{i+m-1}, x_{i+m}\}, \quad m = 0, 1, 2. \tag{12}$$

The numerical fluxes $\hat{f}^m(x_{i+\frac{1}{2}}) = \hat{f}^m_{i+\frac{1}{2}} = h_{i+\frac{1}{2}} + O(\Delta x^3)$ are then calculated for each of the small substencils (12). Let $\hat{f}^m(x)$ be the polynomial approximation of $h(x)$ on each of the substencils (12). By evaluation of these polynomials at $x = x_{i+\frac{1}{2}}$ the following explicit formulas can be obtained [18]

$$\begin{aligned}
\hat{f}^0_{i+\frac{1}{2}} &= \frac{2f(u_{i-2}) - 7f(u_{i-1}) + 11f(u_i)}{6}, \\
\hat{f}^1_{i+\frac{1}{2}} &= \frac{-f(u_{i-1}) + 5f(u_i) + 2f(u_{i+1})}{6}, \\
\hat{f}^2_{i+\frac{1}{2}} &= \frac{2f(u_i) + 5f(u_{i+1}) - f(u_{i+2})}{6},
\end{aligned} \tag{13}$$

where the value of a function f at $u(x_i)$ is indicated by $f(u_i) = f(u(x_i))$. Then, we obtain a final approximation on a big stencil (11) as a linear combination of the fluxes (13)

$$\hat{f}_{i+\frac{1}{2}} = \sum_{m=0}^{2} d_m \hat{f}^m_{i+\frac{1}{2}}, \tag{14}$$

where the coefficients d_m are the linear weights, which would form the upstream fifth order central scheme for the 5-point stencil and their values are

$$d_0 = \frac{1}{10}, \quad d_1 = \frac{6}{10}, \quad d_2 = \frac{3}{10}. \tag{15}$$

As described in [17, 18], the linear weights can be replaced by *nonlinear weights* ω^{JS}_m, $m = 0, 1, 2$, such that

$$\hat{f}_{i+\frac{1}{2}} = \sum_{m=0}^{2} \omega^{JS}_m \hat{f}^m_{i+\frac{1}{2}}, \tag{16}$$

with

$$\omega^{JS}_m = \frac{\alpha^{JS}_m}{\sum_{i=0}^{2} \alpha^{JS}_i}, \qquad \text{where} \qquad \alpha^{JS}_m = \frac{d_m}{(\epsilon + \beta_m)^2}. \tag{17}$$

The parameter β_m is crucial for deciding which substencils to include in the final flux approximation. It is referred to as *smoothness indicator* and its main role is to reduce or remove the contribution of the substencil S^m, which contains the discontinuity. In this case the corresponding nonlinear weight ω^{JS}_m becomes smaller. For smooth parts of the solution, the indicators are designed to come closer to zero, so that the nonlinear weights ω^{JS}_m comes closer to the ideal weights d_m. We will further analyze the smoothness indicators in the next section. The parameter ϵ is used to prevent the denominator from becoming zero. In all our experiments, we set the value of ϵ to 10^{-6}.

2.2. Smoothness indicators

In [17], the smoothness indicators have been developed as:

$$\beta_m = \sum_{q=1}^{2} \Delta x^{2q-1} \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \Big(\frac{\mathrm{d}^q \hat{f}^m(x)}{\mathrm{d}x^q} \Big)^2 dx, \tag{18}$$

with $\hat{f}^m(x)$ being the polynomial approximation in each of three substencils. Their explicit form corresponding to the flux approximation $\hat{f}_{i+\frac{1}{2}}$ can be obtained as

$$\begin{aligned}
\beta_0 &= \frac{13}{12}\big(f(u_{i-2}) - 2f(u_{i-1}) + f(u_i)\big)^2 + \frac{1}{4}\big(f(u_{i-2}) - 4f(u_{i-1}) + 3f(u_i)\big)^2, \\
\beta_1 &= \frac{13}{12}\big(f(u_{i-1}) - 2f(u_i) + f(u_{i+1})\big)^2 + \frac{1}{4}\big(-f(u_{i-1}) + f(u_{i+1})\big)^2, \\
\beta_2 &= \frac{13}{12}\big(f(u_i) - 2f(u_{i+1}) + f(u_{i+2})\big)^2 + \frac{1}{4}\big(3f(u_i) - 4f(u_{i+1}) + f(u_{i+2})\big)^2.
\end{aligned} \tag{19}$$

Remark 2. *As mentioned before, we only considered the construction of the numerical flux $\hat{f}^+_{i+\frac{1}{2}}$. For the numerical approximation of the flux $\hat{f}^+_{i-\frac{1}{2}}$ we can use formulas* (13)–(17) *and* (19) *and shift each index by* -1.

The negative part of the flux splitting can be obtained using symmetry (see, e.g., [31]), and we briefly summarize the formulas for $\hat{f}^-_{i+\frac{1}{2}}$ and omit the superscript $^-$:

$$\begin{aligned}
\hat{f}^0_{i+\frac{1}{2}} &= \frac{11f(u_{i+1}) - 7f(u_{i+2}) + 2f(u_{i+3})}{6}, \\
\hat{f}^1_{i+\frac{1}{2}} &= \frac{2f(u_i) + 5f(u_{i+1}) - f(u_{i+2})}{6}, \\
\hat{f}^2_{i+\frac{1}{2}} &= \frac{-f(u_{i-1}) + 5f(u_i) + 2f(u_{i+1})}{6},
\end{aligned} \tag{20}$$

where the weights ω_m^{JS} are computed as in (17) using the smoothness indicators given by

$$\begin{aligned}
\beta_0 &= \frac{13}{12}\big(f(u_{i+1}) - 2f(u_{i+2}) + f(u_{i+3})\big)^2 + \frac{1}{4}\big(3f(u_{i+1}) - 4f(u_{i+2}) + f(u_{i+3})\big)^2, \\
\beta_1 &= \frac{13}{12}\big(f(u_i) - 2f(u_{i+1}) + f(u_{i+2})\big)^2 + \frac{1}{4}\big(f(u_i) - f(u_{i+2})\big)^2, \\
\beta_2 &= \frac{13}{12}\big(f(u_{i-1}) - 2f(u_i) + f(u_{i+1})\big)^2 + \frac{1}{4}\big(f(u_{i-1}) - 4f(u_i) + 3f(u_{i+1})\big)^2.
\end{aligned} \tag{21}$$

In the next section, where the deep learning algorithm will be introduced, this will help to understand how the improved smoothness indicators will be constructed.

2.3. The WENO-Z scheme

Borges et al. [25] pointed out that the classical WENO-JS scheme described in previous sections looses the fifth-order accuracy at the critical points where $f'(u) = 0$, and proposed new nonlinear weights defined by

$$\omega_m^Z = \frac{\alpha_m^Z}{\sum_{i=0}^{2} \alpha_i^Z}, \quad \text{where} \quad \alpha_m^Z = d_m \left[1 + \left(\frac{\tau_5}{\beta_m + \epsilon} \right)^2 \right] \tag{22}$$

and

$$\tau_5 = |\beta_0 - \beta_2| \tag{23}$$

is a new global smoothness indicator.

3. Deep smoothness WENO scheme

In [32, 33, 34] the new WENO-DS scheme based on the improvement of the smoothness indicators was developed. The smoothness indicators β_m, $m = 0, 1, 2$, are multiplied by the perturbations δ_m, which are the outputs of the respective neural network algorithm. The new smoothness indicators are denoted by β_m^{DS}:

$$\beta_m^{DS} = \beta_m(\delta_m + C), \qquad m = 0, 1, 2, \tag{24}$$

where C is a constant that ensures the consistency and accuracy of the new method. In all our experiments we set $C = 0.1$. For more details and corresponding theoretical proofs of accuracy and consistency we refer to [32, 33].

Note that the formulation of the new smoothness indicators β_m^{DS} as a multiplication of the original ones with the perturbations δ_m is very favorable. In a case where the original smoothness indicator converges to zero, the improved β_m^{DS} behaves in the same way. On the other hand, if a subset S^m contains a discontinuity, the perturbation δ_m can improve the original smoothness indicator so that the final scheme exhibits better numerical approximations. Moreover, the theoretical convergence properties are not lost, see [32, 33].

In [32] the algorithm was successfully applied to one-dimensional benchmark examples such as the Burgers' equation, the Buckley-Leverett equation, the one-dimensional Euler system, and the two-dimensional Burgers' equation. In [33], the algorithm was extended to nonlinear degenerate parabolic equations and further applied to computational finance problems in [34]. The theoretical order of convergence was demonstrated on the smooth solutions and the large numerical improvements were obtained when comparing the WENO-DS method with the original WENO methods.

3.1. Preservation of a conservative property for WENO-DS scheme

However, the multipliers introduced for the smoothness indicators in [32] were cell-based (not interface-based). This means that although the high numerical accuracy was theoretically demonstrated and numerically confirmed, the guarantee of the conservative property was lost. As stated in [33], the conservative property can be easily recovered by defining the multipliers such that

$$\begin{aligned} \beta^{DS}_{m,i+\frac{1}{2}} &= \beta_{m,i+\frac{1}{2}}(\delta_{m,i+\frac{1}{2}} + C), \\ \beta^{DS}_{m,i-\frac{1}{2}} &= \beta_{m,i-\frac{1}{2}}(\delta_{m,i-\frac{1}{2}} + C), \end{aligned} \tag{25}$$

with

$$\delta_{0,i+\frac{3}{2}} = \delta_{1,i+\frac{1}{2}} = \delta_{2,i-\frac{1}{2}}, \quad i = 0, \dots, N. \tag{26}$$

This makes the multipliers depend on the location of the substencils corresponding to $\beta_{m,i+\frac{1}{2}}$ and $\beta_{m,i-\frac{1}{2}}$. This ensures that the values $\hat{f}^{\pm}_{i-\frac{1}{2}}$ can be obtained from the values $\hat{f}^{\pm}_{i+\frac{1}{2}}$ by simple index shifting and that the conservative property is preserved.

3.2. Structure of neural network

To ensure the consistency of a numerical method, the Convolutional Neural Network (CNN) is used. This is crucial to ensure the spatial invariance of the resulting numerical method. This means that the multipliers δ_m are independent of their position in the spatial grid and only depend on the solution itself.

Let us formulate the CNN as a function $H(\cdot) \in \mathbb{R}^{2k+1} \to \mathbb{R}$, where $2k+1$ is the size of the receptive field of the CNN:

$$H\big(\bar{f}(\bar{u}_i)\big) = \mathrm{CNN}\big(\bar{f}(\bar{u}_i)\big). \tag{27}$$

As an input we define a vector

$$\begin{aligned} \bar{f}(\bar{u}_i) &= \big(f(u(x_{i-k})), f(u(x_{i-k+1})), \dots, f(u(x_{i+k}))\big), \\ \bar{u}_i = \bar{u}(\bar{x}_i) &= \big(u(x_{i-k}), u(x_{i-k+1}), \dots, u(x_{i+k})\big). \end{aligned} \tag{28}$$

The Figure 1 shows the values from which the multipliers δ_m, $m = 0, 1, 2$ are constructed, assuming $2k + 1 = 3$ for the receptive field. In this case, the values used to compute the original smoothness indicators are also used to compute the multipliers δ_m, $m = 0, 1, 2$, (see equations (19) and (21)). If we enlarge the receptive field of the CNN, we also enlarge the stencil for computing the multipliers δ_m, $m = 0, 1, 2$. In this way, the smoothness indicators are basically computed from a wider stencil, which can lead to better numerical approximations. In this case, we just need to add more bounds before feeding the values (28) to the CNN.

As we are improving the existing numerical scheme and adding a neural network part to it, it is important that the new numerical scheme remains computationally efficient. The neural

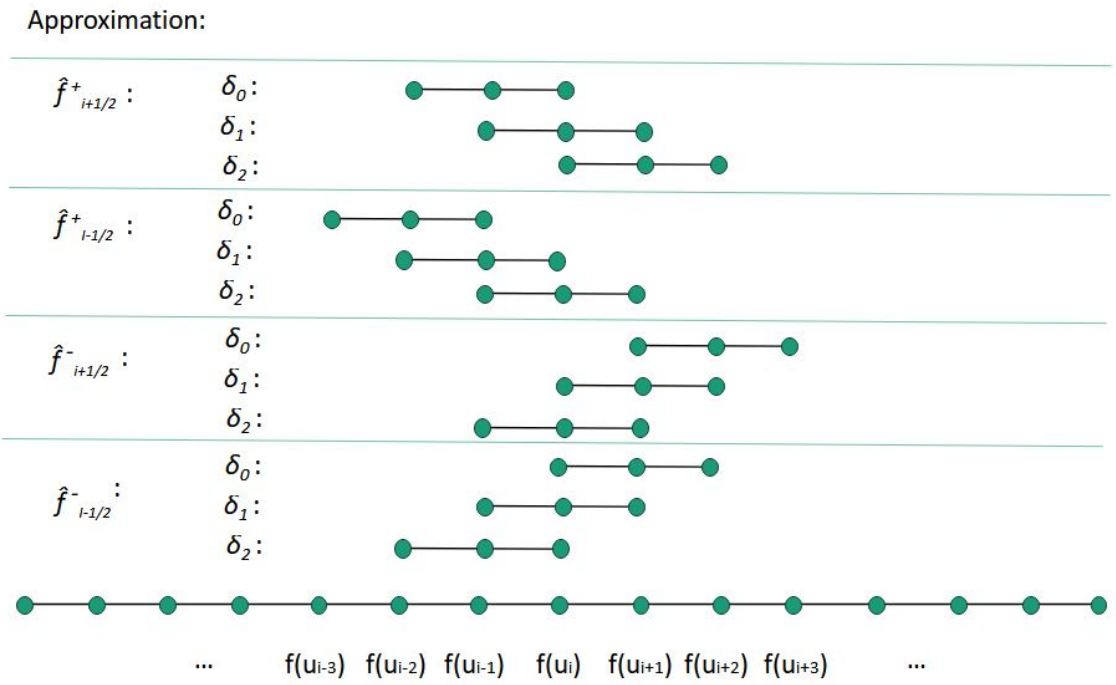


Figure 1: The substencils used for computation of multipliers δ_m, $m = 0, 1, 2$ corresponding to the flux approximations $\hat{f}^{\pm}_{i\pm\frac{1}{2}}$, assuming that for the receptive field of the CNN holds $2k + 1 = 3$.

network part added to the numerical scheme could be computationally expensive. However, we propose to use only a small CNN, which would not have such high computational costs. The detailed structure of the CNN can be found in Section 4.1.

It was pointed out in [32] that better numerical results were obtained using two different neural networks for the positive and negative part of a flux. We experimentally found that we can avoid using more neural networks and use only one CNN. On the other hand, we can achieve better results by using a superior training procedure and adaptive activation functions. More details will be discussed in the next subsections.

For convergence and consistency of the numerical scheme, all hidden layers of the CNN must be differentiable functions and the activation function in the last CNN layer must be bounded from below [33]. Experimentally, we found that the use of a *softplus* activation function in the last CNN layer is more effective and gives better numerical results compared to e.g. *sigmoid* as used in [32].

3.2.1. Adaptive activation functions

We can make the training more effective and get better numerical results by using *adaptive activation functions*. [35, 36, 37]. The activation function is one of the most important hyperparameters in neural network architectures. The purpose of this hyperparameter is to introduce nonlinearity into the prediction. There are many activation functions proposed in the literature; see the comprehensive survey [38] for more details. However, there is no basic rule for the choice of the activation function. This is the motivation to use an adaptive activation function that can adapt to the problem at hand. In this work, we used global adaptive activation functions [35], where the additional slope parameter is introduced in the activation function as follows.

For the ELU activation function, we train the additional parameter α:

$$\text{ELU} = \begin{cases} x, & \text{if} \quad x > 0, \\ \alpha(\exp(x) - 1) & \text{if} \quad x \leq 0 \end{cases} \tag{29}$$

and we denote the adaptive ELU as *aELU*. For the softplus activation function, we train the additional parameter β:

$$\text{Softplus}(x) = \frac{1}{\beta}\log(1 + \exp(\beta x)) \tag{30}$$

and we denote the adaptive softplus as *aSoftplus*.

3.3. Training procedure

In this section we describe how the training procedure for WENO-DS is carried out. We have experimented with different training procedures and have found experimentally that following the training procedure described in [33] gives the best numerical results. First we have to create the data set. For this purpose we compute the reference solutions using the WENO-Z method on a fine grid of $I \times J = 400 \times 400$ space points up to the given final time T, where t_n represents the time points, $n = 0, \ldots, N$. More details on the construction of the reference solutions are given in Sections 5.1, 5.2, 5.3.

During a training we compute the numerical solutions on a grid of $I \times J = 100 \times 100$ space points. At the beginning of a training we randomly select a problem from a data set and perform a single time step to get to the time t_{n+1}, using CNN to predict the multipliers δ_m. However, by performing a single time step on a coarse grid, we do not match the time step size of the fine precomputed solutions, as the adaptive time step size is used. So we simply take the closest reference solution from the data set, use it as an initial condition and do another small time step to get a reference solution in time t_{n+1}. Then we compute the loss and its gradient with respect to the weights of the CNN.

We then decide whether to proceed to the next time step of a current problem or to choose another problem from our dataset and run a time step of that problem. The probability of choosing the new problem has to be determined at the beginning of the training session and

we set it to $\varphi = 0.5$ in our experiments. We set the maximum number of opened problems to 150. We remember all opened problems, and if no new problem is opened (with probability $1 - \varphi$), or if the maximum number of opened problems is reached, we execute the next time step of a problem uniformly chosen from the set of already opened problems. Keeping the solution from the previous time step as initial data, we repeat the same procedure until we reach the maximum number of training steps. This training procedure gives us a great opportunity to mix the solutions with different initial data and in different time points, which makes the training more effective.

3.3.1. Optimizer and the optimal learning rate

To train the network, we used a gradient-based optimiser, namely a variant of stochastic gradient descent, the Adam optimiser [39].

The learning rate is another important hyperparameter to choose. A larger learning rate may miss the local minima, and a smaller learning rate may require a large number of iterations to reach convergence. Therefore, it is important to find a near-optimal learning rate. In this work, the learning rate is 0.001 to update the weights of the CNN. This near-optimal learning rate was found through experiments.

3.3.2. Loss function

In this work, the loss function consists of the data mismatch term between the solution predicted by the networks and the reference solution. For the loss function, we use the mean square error loss as follows:

$$LOSS_{\mathrm{MSE}}(u) = \frac{1}{I}\sum_{i=0}^{I}(u_i - u_i^{\mathrm{ref}})^2, \tag{31}$$

where u_i is a numerical approximation of $u(x_i)$ and u_i^{ref} is the corresponding reference solution. The L_2 norm based loss function has the advantage of stronger gradients with respect to u_i, resulting in faster training. However, in our examples we use the L_1 norm as the main error measure, which is more typical for measuring errors for hyperbolic conservation laws. Thus, for validation during training, we use the metrics

$$L_1(u) = \frac{1}{I}\sum_{i=0}^{I}|u_i - u_i^{\mathrm{ref}}|. \tag{32}$$

4. Application of our approach to the 2D Euler equations

We consider the two-dimensional Euler equations of gas dynamics in the form (1) with

$$U = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ E \end{pmatrix} \qquad F(U) = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(E+p) \end{pmatrix} \qquad G(U) = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(E+p) \end{pmatrix} \tag{33}$$

for polytropic gas. Here, the variable ρ is the density, u the x-velocity component, v the y-velocity component, E the total energy and p the pressure. Further, it holds

$$p = (\gamma - 1)\Big[E - \frac{\rho}{2}(u^2 + v^2)\Big]. \tag{34}$$

γ denotes the ratio of the specific heats and we will use $\gamma \in (1.1, 1.67)$ in this paper.

We consider the spatial domain $[0,1] \times [0,1]$ and solve the Riemann problem with the following initial condition

$$(\rho, u, v, p) = \begin{cases} (\rho_1, u_1, v_1, p_1) & x > 0.5 \quad \text{and} \quad y > 0.5, \\ (\rho_2, u_2, v_2, p_2) & x < 0.5 \quad \text{and} \quad y > 0.5, \\ (\rho_3, u_3, v_3, p_3) & x < 0.5 \quad \text{and} \quad y < 0.5, \\ (\rho_4, u_4, v_4, p_4) & x > 0.5 \quad \text{and} \quad y < 0.5. \end{cases} \tag{35}$$

The combination of four elementary planar waves is used to define the classification of the Riemann problem. A detailed study of these configurations has been done in [40, 41, 42, 43, 44, 45] and there are 19 different possible configurations for polytropic gas. These are defined by three types of elementary waves, namely a backward rarefaction wave $\overleftarrow{R}$, a backward shock wave $\overleftarrow{S}$, a forward rarefaction wave $\overrightarrow{R}$, a forward shock wave $\overrightarrow{S}$ and a contact discontinuity $J^{\pm}$, where the superscript $\pm$ refers to negative and positive contacts.

To obtain the WENO approximations in the two-dimensional example, we apply the procedure described in Section 2 using the dimension-by-dimension principle. Thus we obtain the flux approximations for (1) as

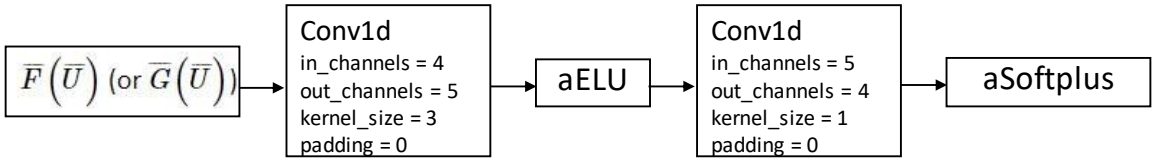
$$\begin{aligned} \frac{1}{\Delta x}\big(\hat{f}_{i+\frac{1}{2}} - \hat{f}_{i-\frac{1}{2}}\big) &= \big(F(U)\big)_x\big|_{(x_i,y_j)} + O\big(\Delta x^5\big), \\ \frac{1}{\Delta y}\big(\hat{g}_{i+\frac{1}{2}} - \hat{g}_{i-\frac{1}{2}}\big) &= \big(G(U)\big)_y\big|_{(x_i,y_j)} + O\big(\Delta y^5\big), \end{aligned} \tag{36}$$

with the uniform grid defined by the nodes (x_i, y_j), $\Delta x = x_{i+1} - x_i$, $\Delta y = y_{j+1} - y_j$, $i = 0, \ldots, I$, $j = 0, \ldots, J$.

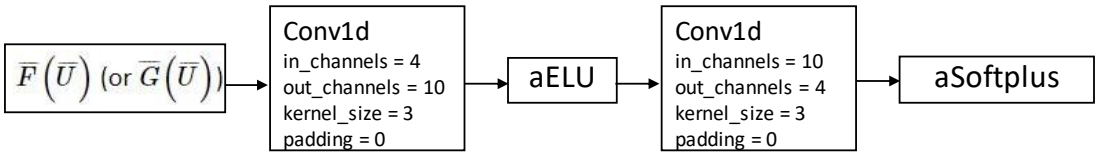
In our examples we proceed with the implementation of the Euler system using characteristic decomposition. This means that we first project the solution and the flux onto the characteristic fields using left eigenvectors. Then we apply the Lax-Friedrichs flux splitting (10) for each component of the characteristic variables. These values are fed into the CNN and the enhanced smoothness indicators are computed. After obtaining the final WENO approximation, the projection back to physical space is done using right eigenvectors, see [46] for more details on this procedure.

4.1. Size of the neural network

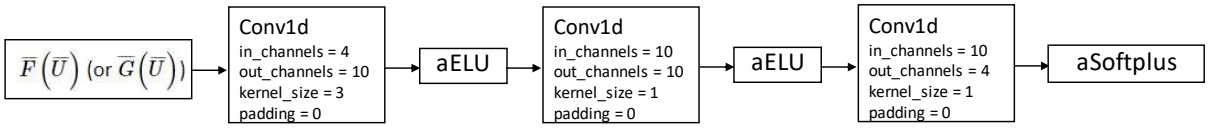
In our paper, we considered different structures of neural networks and carried out numerous experiments with them. First, we used a rather simple CNN with only two layers and a receptive field of width 3. The structure is shown in Figure 2a. The advantage of this is its computational efficiency. Second, we used a CNN with the same number of layers, but we increased the number of channels and made the receptive field wider. The structure is shown in Figure 2b. Finally, we used only a receptive field of width 3, but added one more layer and used a more complex neural network, as shown in Figure 2c. Each of these neural networks gave interesting results and we summarize them in Section 5.



(a) Two hidden layers, lower number of channels, receptive field of size 3.



(b) Two hidden layers, higher number of channels, receptive field of size 5.



(c) Three hidden layers, higher number of channels, receptive field of size 3.

Figure 2: Different structures of the convolutional neural network.

As can be seen, we have 4 input channels in the first hidden layer and 4 output channels in the last hidden layer in each CNN. These represent the dimension of the solution U from (33). In this way, the neural network also takes in information from other variables, which can be useful for improving the numerical solution. The input $\bar{F}(\bar{U})$, respectively $\bar{G}(\bar{U})$ represents the numerical approximation after the projection using left eigenvectors and after applying the flux splitting method.

We also have to adapt the loss function from (31) and use for training

$$LOSS_{\mathrm{MSE}}(\rho, u, v, p) = LOSS_{\mathrm{MSE}}(\rho) + LOSS_{\mathrm{MSE}}(u) + LOSS_{\mathrm{MSE}}(v) + LOSS_{\mathrm{MSE}}(p) \quad (37)$$

and for the validation during training from (32)

$$L_1(\rho, u, v, p) = L_1(\rho) + L_1(u) + L_1(v) + L_1(p). \quad (38)$$

When we plot the error on validation problems, we rescale the values for each validation problem to be in the interval $[0, 1]$ using the relationship

$$L_1^*(\rho, u, v, p) = \frac{L_1^l(\rho, u, v, p)}{\max_l(L_1^l(\rho, u, v, p))}, \qquad l = 0, \ldots, L, \tag{39}$$

where L denotes the total number of training steps.

4.2. Construction of the data set for the CNN training procedure

For each of the 19 configurations of the Riemann problem, the specific relations must be satisfied by the initial data and the symmetry properties of the solution. We present the formulas given in [41] and create the data sets for the CNN training according to these formulas.

We define

$$\Phi_{lr} := \frac{2\sqrt{\gamma}}{\gamma - 1}\Big(\sqrt{\frac{p_l}{\rho_l}} - \sqrt{\frac{p_r}{\rho_r}}\Big), \quad \Psi_{lr}^2 := \frac{(p_l - p_r)(\rho_l - \rho_r)}{\rho_l \rho_r}, \quad (\Psi_{lr} > 0) \tag{40}$$

and

$$\Pi_{lr} := \Big(\frac{p_l}{p_r} + \frac{(\gamma - 1)}{(\gamma + 1)}\Big) \Big/ \Big(1 + \frac{(\gamma - 1)}{(\gamma + 1)}\frac{p_l}{p_r}\Big). \tag{41}$$

In Sections 5.1, 5.2, 5.3 we list the specific relations for given examples that are sufficient to uniquely define the solution. Following these relations, we randomly generate the initial data and construct our data sets.

5. Numerical results

To demonstrate the efficiency of the proposed method, in this section we present the numerical results obtained with the WENO-DS method after the CNN training procedure. Note that the CNN training procedure only needs to be performed once as *offline* training for each of the examples presented in Sections 5.1, 5.2, 5.3. No additional training was performed for the examples in Section 5.4 as we show the results using the same trained CNN from the previous examples. In Section 5.5 we perform two more trainings with larger CNN and illustrate the results. Further details can be found in the respective sections.

For the following system of ordinary differential equations (ODEs)

$$\frac{\mathrm{d}U(t)}{\mathrm{d}t} = L(U), \tag{42}$$

we use a third-order *total variation diminishing* (TVD) Runge-Kutta method [17] given by

$$\begin{aligned} U^{(1)} &= U^n + \Delta t\, L(U^n), \\ U^{(2)} &= \frac{3}{4}U^n + \frac{1}{4}U^{(1)} + \frac{1}{4}\Delta t\, L(U^{(1)}), \\ U^{n+1} &= \frac{1}{3}U^n + \frac{2}{3}U^{(2)} + \frac{2}{3}\Delta t\, L(U^{(2)}), \end{aligned} \tag{43}$$

where U^n is the numerical solution at the time step n.

For the scheme (43) we use an adaptive step size

$$\Delta t = 0.6 \min\Big(\frac{\Delta x}{a}, \frac{\Delta y}{a}\Big), \tag{44}$$

with

$$a = \max_{\substack{i=0,\dots,I \\ j=0,\dots,J}} (|\lambda^+_{i,j}|, |\lambda^-_{i,j}|) \quad \lambda^\pm = V \pm c, \quad V = \sqrt{u^2 + v^2} \quad c^2 = \gamma \frac{p}{\rho}, \tag{45}$$

where u, v are the velocities and c is the local speed of sound.

In the sequel we enumerate the different configurations of initial conditions according to [44].

5.1. Configuration 2

This is the configuration with four rarefaction waves: $\overrightarrow{R}_{21}$, $\overleftarrow{R}_{32}$, $\overleftarrow{R}_{34}$, $\overrightarrow{R}_{41}$. The detailed analysis was done in [45, 41] and we have to satisfy the following relations for this case:

$$\begin{aligned} &u_2 - u_1 = \Phi_{21}, \quad u_4 - u_3 = \Phi_{34}, \quad u_3 = u_2, \quad u_4 = u_1, \\ &v_4 - v_1 = \Phi_{41}, \quad v_2 - v_3 = \Phi_{32}, \quad v_2 = v_1, \quad v_3 = v_4 \end{aligned} \tag{46}$$

with the compatibility conditions $\Phi_{21} = -\Phi_{34}$ and $\Phi_{41} = -\Phi_{32}$. Moreover, for a polytropic gas the equations

$$\rho_l/\rho_r = (p_l/p_r)^{1/\gamma} \quad \text{for} \quad (l, r) \in \{(2, 1), (3, 4), (3, 2), (4, 1)\} \tag{47}$$

have to be included. Furthermore, we have $\rho_2 = \rho_4$, $\rho_1 = \rho_3$, $p_1 = p_3$, $p_2 = p_4$, $u_2 - u_1 = v_4 - v_1$ and $u_4 - u_3 = v_2 - v_3$.

We use for creating of the data set the values

$$\begin{aligned} &\rho_1 \in \mathcal{U}[0.7, 2], \quad \rho_2 \in \mathcal{U}[0.5, \rho_1], \quad p_1 \in \mathcal{U}[0.2, 1.5], \\ &u_1 \in \mathcal{U}[-1, 1], \quad v_1 = u_1, \quad \gamma \in (1.1, 1.67) \end{aligned} \tag{48}$$

and for the other values we use the relations (46), (47) with (40). We also compute the reference solutions using the WENO-Z method on a grid $I \times J = 400 \times 400$ space points up to the final time $T \in \mathcal{U}[0.1, 0.2]$ and create the data set consisting of 50 reference solutions.

For training, we use the training procedure described in Section 3.3. First, we use the simplest neural network structure shown in Figure 2a and perform the training for the total number of 4000 training steps. We plot the evolution of the L_1^* error (39) for the validation problems in Figure 3. Note that these problems were not included in the training data, and the initial conditions of these problems were generated analogously to the construction of the training data set. For these problems, we measured the error every 100 training steps

and at a randomly chosen final time T. We select the final model based on the evolution of the error of the validation set. We see that the error decreases up to a certain point for all problems and then starts to increase for some problems. Longer training would lead to overfitting of the training data. Finally, we choose the final model from the 2800 training step and present the results using this model.

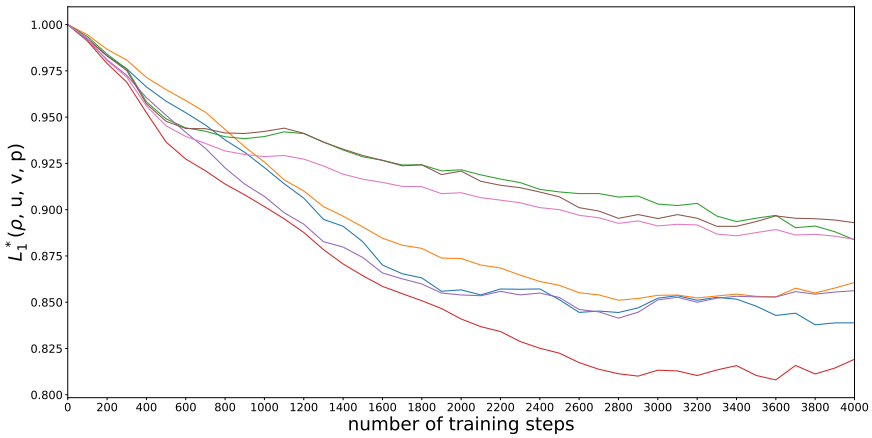


Figure 3: The values (39) for different validation problems evaluated each 100 training steps.

As a test problem we use the problem from [44] with $\gamma = 1.4$, $T = 0.2$ and the initial condition

$$(\rho, u, v, p) = \begin{cases} (1, 0, 0, 1) & x > 0.5 \quad \text{and} \quad y > 0.5, \\ (0.5197, -0.7259, 0, 0.4) & x < 0.5 \quad \text{and} \quad y > 0.5, \\ (1, -0.7259, -0.7259, 1) & x < 0.5 \quad \text{and} \quad y < 0.5, \\ (0.5197, 0, -0.7259, 0.4) & x > 0.5 \quad \text{and} \quad y < 0.5. \end{cases} \tag{49}$$

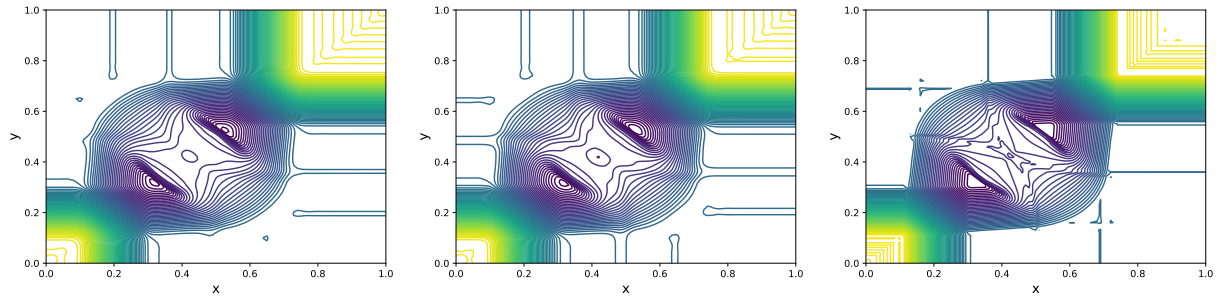
The results are shown in Table 1. As can be seen, we achieve a significant error improvement for all four variables and for different discretizations. It should be noted that we trained only with the discretization 100×100 space points and did not retrain the neural network for different discretizations. We refer to the error of the WENO-Z method divided by the error of WENO-DS (rounded to 2 decimal points) as the 'ratio'. The density contour plots are shown in Figure 4 and the absolute pointwise errors for the density are shown in Figure 5.

Finally, we want to compare the computational cost of WENO-DS compared to the original WENO scheme in solving the problem shown in Figure 6. Using a logarithmic scale, we plot the computation time against the L_1 error averaged over the four variables ρ, u, v, p.

It should be noted that if we were to test the method on another unseen test problem using the initial data from the previously described range, we would obtain very similar error improvements in those cases.

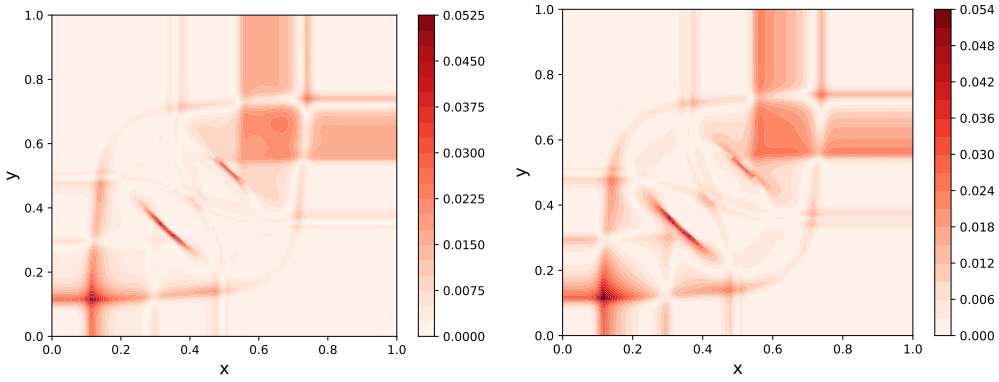
$I \times J$	50×50			100×100			200×200		
	WENO-Z	WENO-DS	ratio	WENO-Z	WENO-DS	ratio	WENO-Z	WENO-DS	ratio
ρ	0.012488	0.010722	1.16	0.005465	0.004648	1.18	0.001862	0.001547	1.20
u	0.014363	0.011986	1.20	0.006153	0.005066	1.21	0.002053	0.001627	1.26
v	0.014363	0.011986	1.20	0.006153	0.005066	1.21	0.002053	0.001627	1.26
p	0.013113	0.011510	1.14	0.005619	0.004899	1.15	0.001879	0.001587	1.18

Table 1: Comparison of L_1 error of WENO-Z and WENO-DS methods for the solution of the Euler system with the initial condition (49) for different spatial discretizations, $T = 0.2$.



(a) WENO-DS (b) WENO-Z (c) reference solution

Figure 4: Density contour plot for the solution of the Riemann problem with the initial condition (49), $I \times J = 100 \times 100$, $T = 0.2$.



(a) WENO-DS (b) WENO-Z

Figure 5: Absolute pointwise errors for the density solution of the Riemann problem with the initial condition (49), $I \times J = 100 \times 100$, $T = 0.2$.

5.2. Configuration 3

This is the configuration with four shock waves: $\overleftarrow{S}_{21}$, $\overleftarrow{S}_{32}$, $\overleftarrow{S}_{34}$, $\overleftarrow{S}_{41}$. According to [41], in this case we have the following equations that must be satisfied:

$$\begin{aligned} &u_2 - u_1 = \Psi_{21}, \quad u_3 - u_4 = \Psi_{34}, \quad u_3 = u_2, \quad u_4 = u_1, \\ &v_4 - v_1 = \Psi_{41}, \quad v_3 - v_2 = \Psi_{32}, \quad v_2 = v_1, \quad v_3 = v_4 \end{aligned} \tag{50}$$

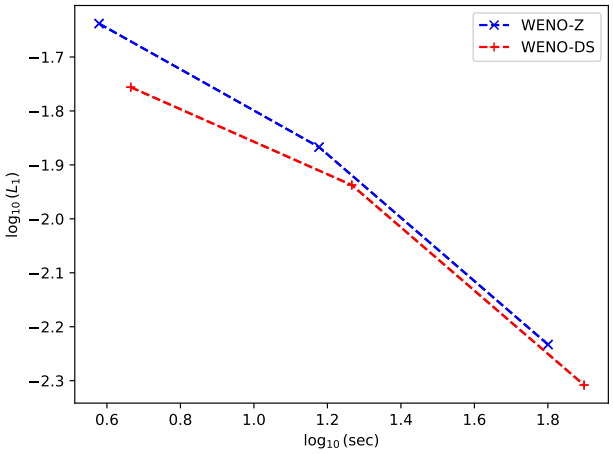


Figure 6: Comparison of computational cost against L_1-error of the solution of the Riemann problem with the initial condition (49).

and for polytropic gas the equations

$$\rho_l/\rho_r = \Pi_{lr} \quad \text{for} \quad (l,r) \in \{(2,1),(3,4),(3,2),(4,1)\} \tag{51}$$

are added. This gives the compatibility conditions $\Psi_{21} = \Psi_{34}$ and $\Psi_{41} = \Psi_{32}$. Furthermore, we have $\rho_2 = \rho_4$, $p_2 = p_4$ and $u_2 - u_1 = v_4 - v_1$.

In this case we use for creating the data set the values

$$\begin{gathered} \rho_1 \in \mathcal{U}[1,2], \quad \rho_2 \in \mathcal{U}[0.5,1], \quad p_1 \in \mathcal{U}[1,2], \\ u_1 \in \mathcal{U}[-0.25,0.25], \quad v_1 = u_1, \quad \gamma \in (1.1,1.67) \end{gathered} \tag{52}$$

and for the other values we use the relations (50), (51) with (40) and (41). Similar to the previous example, we compute the reference solutions using the WENO-Z method on a grid $I \times J = 400 \times 400$ space points up to the final time $T \in \mathcal{U}[0.1,0.3]$ and create the data set consisting of 50 reference solutions.

We proceed with training as described in the previous section, using the same neural network structure as shown in Figure 2a. Again, we train only on the discretization $I \times J = 100 \times 100$ space steps. We run the training for 4000 training steps and plot the evolution of the validation metrics (39) for the validation problems in Figure 7. We measured the error every 100 training steps and at the randomly chosen final time T. Based on this, we choose the final model from training step 3200 and present the results for the test problem with

$\gamma = 1.4$, $T = 0.3$, and initial condition [44]

$$(\rho, u, v, p) = \begin{cases} (1.5, 0, 0, 1.5) & x > 0.5 \quad \text{and} \quad y > 0.5, \\ (0.5323.1.206, 0, 0.3) & x < 0.5 \quad \text{and} \quad y > 0.5, \\ (0.138, 1.206, 1.206, 0.029) & x < 0.5 \quad \text{and} \quad y < 0.5, \\ (0.5323, 0, 1.206, 0.3) & x > 0.5 \quad \text{and} \quad y < 0.5. \end{cases} \tag{53}$$

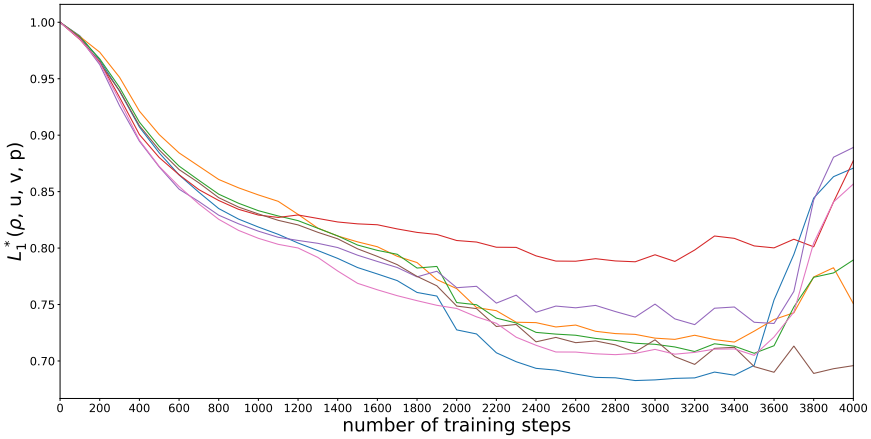


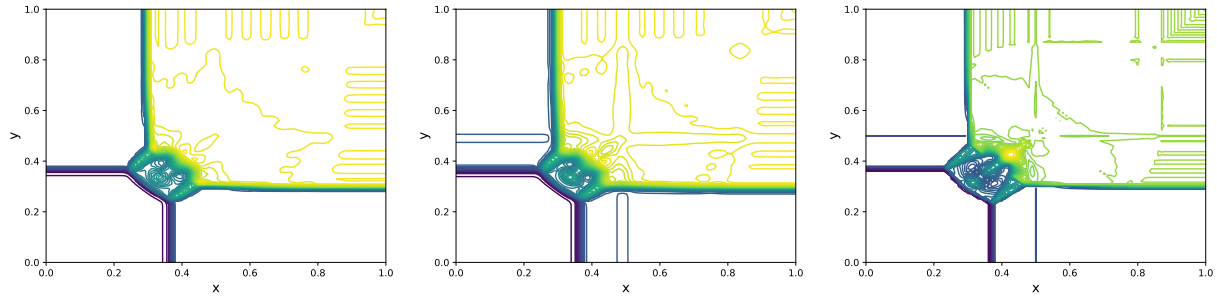
Figure 7: The values (39) for different validation problems evaluated each 100 training steps.

We compare the results in Table 2. As can be seen, we achieve a large error improvement for all discretizations listed. The density contour plots can be found in Figure 8 and the absolute pointwise errors for the density in Figure 9. Here it can be seen that the error of WENO-DS is significantly lower in the areas of the shock contacts.

$I \times J$	50×50			100×100			200×200		
	WENO-Z	WENO-DS	ratio	WENO-Z	WENO-DS	ratio	WENO-Z	WENO-DS	ratio
ρ	0.038682	0.027906	1.39	0.019232	0.012817	1.50	0.007454	0.004657	1.60
u	0.034692	0.027638	1.26	0.019588	0.015043	1.30	0.008249	0.005810	1.42
v	0.034692	0.027638	1.26	0.019588	0.015043	1.30	0.008249	0.005810	1.42
p	0.038920	0.030888	1.26	0.018666	0.014041	1.33	0.007275	0.005001	1.45

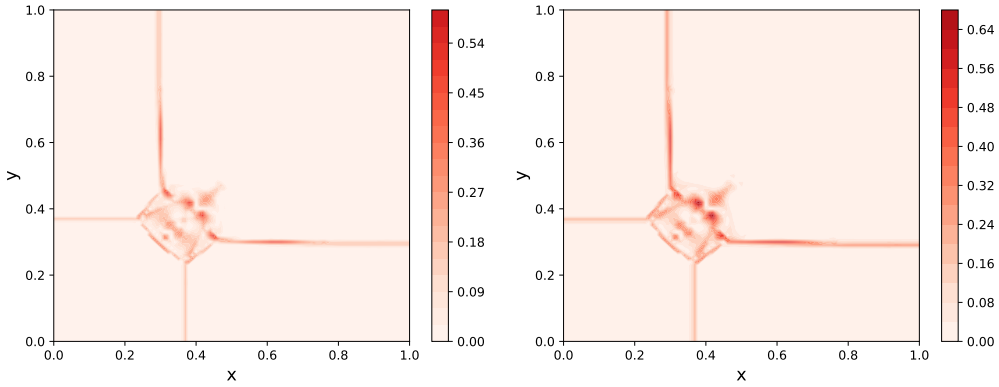
Table 2: Comparison of L_1 error of WENO-Z and WENO-DS methods for the solution of the Euler system with the initial condition (53) for different spatial discretizations, $T = 0.3$.

We also compare the weights ω_m^Z, $m = 0, 1, 2$ (22) and the updated weights ω_m^{DS}, $m = 0, 1, 2$ with the improved smoothness indicators (25). We plot these weights, corresponding to the positive part of a flux $\hat{f}^+$ from the flux splitting, using WENO-Z and WENO-DS for the previous test problem at the final time $T = 0.3$. Since we apply the principle dimension-by-dimension, we present the weights only for the approximation of the flux $F(U)$. For



(a) WENO-DS (b) WENO-Z (c) reference solution

Figure 8: Density contour plot for the solution of the Riemann problem with the initial condition (53), $I \times J = 100 \times 100$, $T = 0.3$.



(a) WENO-DS (b) WENO-Z

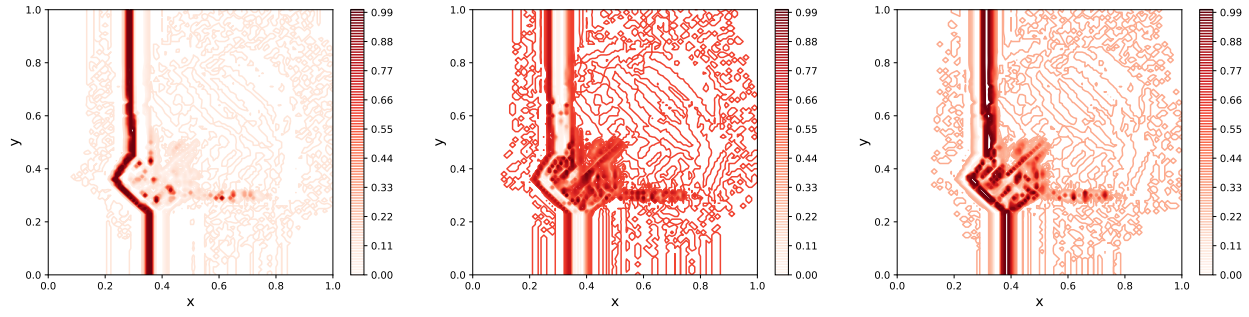
Figure 9: Absolute pointwise errors for the density solution of the Riemann problem with the initial condition (53), $I \times J = 100 \times 100$, $T = 0.3$.

the approximations of the flux $G(U)$, we could obtain these weights in this example using symmetry. As can be seen, WENO-DS is much better at localizing the shock from the other direction as well, which has a significant impact on error improvement.

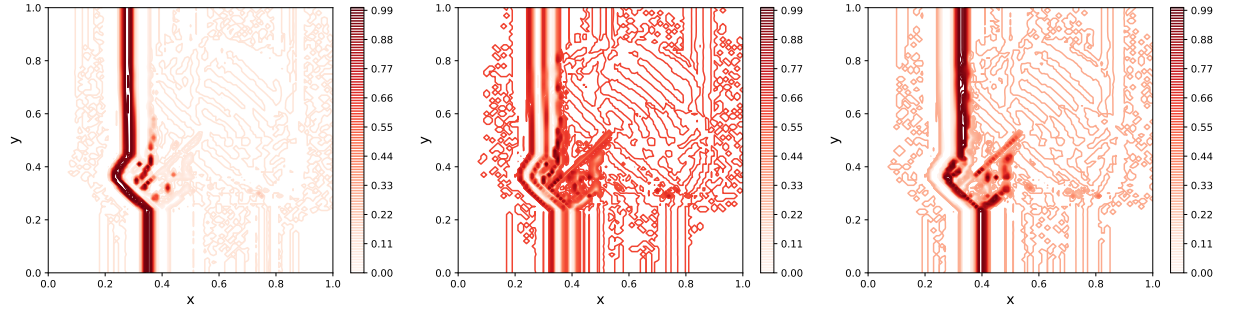
Finally, let us compare the computational cost of WENO-DS for the problem shown in Figure 11. We see that WENO-DS is much more computationally intensive compared to WENO-DS. Again, if we tested the method on the unseen problems, but with the same initial configuration, we would get analogous significant error improvements.

5.3. Configuration 16

This is the configuration with the combination of rarefaction wave, shock wave and contact discontinuities: $\overleftarrow{R}_{21}$, J^{-}_{32}, J^{+}_{34}, $\overrightarrow{S}_{41}$. As shown in [41], the following relations must



(a) WENO-DS, ω_0^{DS} (b) WENO-DS, ω_1^{DS} (c) WENO-DS, ω_2^{DS}



(d) WENO-Z, ω_0^{Z} (e) WENO-Z, ω_1^{Z} (f) WENO-Z, ω_2^{Z}

Figure 10: Comparison of the nonlinear weights $\omega_m^Z, m = 0, 1, 2$ and ω_m^{DS}, $m = 0, 1, 2$.

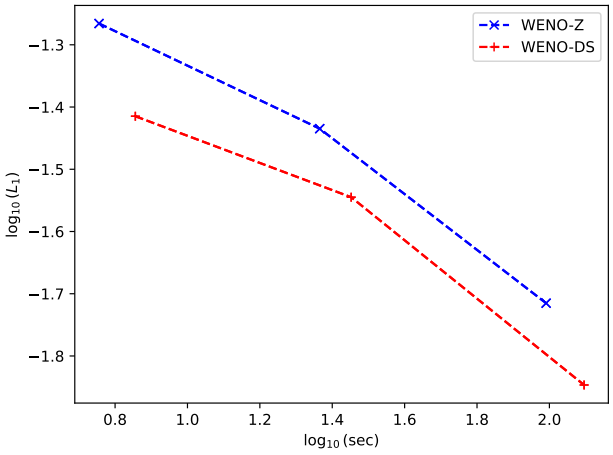


Figure 11: Comparison of computational cost against L_1-error of the solution of the Riemann problem with the initial condition (53).

hold for this case

$$\begin{aligned} u_1 - u_2 =& \Phi_{21}, \quad u_3 = u_4 = u_1, \\ v_4 - v_1 = \Psi_{41}, \quad & v_3 = v_2 = v_1, \quad p_1 < p_2 = p_3 = p_4 \end{aligned} \tag{54}$$

and for polytropic gas we add the equation (47) for a rarefaction and (51) for a shock wave between the lth and rth quadrants.

For our data set we use the values

$$\begin{gathered}\rho_4 \in \mathcal{U}[1,2], \quad \rho_3 \in \mathcal{U}[0.5,\rho_4], \quad p_1 \in \mathcal{U}[0.3,1], \quad p_2 \in \mathcal{U}[1,1.5], \\ u_1 \in \mathcal{U}[-0.25,0.25], \quad v_1 = u_1, \quad \gamma \in (1.1,1.67)\end{gathered} \tag{55}$$

To compute the data set consisting of 50 reference solutions, we use the WENO-Z method on a grid $I \times J = 400 \times 400$ space points up to the final time $T \in \mathcal{U}[0.1,0.2]$.

We train the CNN with the structure shown in Figure 2a as in the previous examples on the discretization $I \times J = 100 \times 100$ space steps for the total number of 2000 training steps. We show the evolution of the validation metrics (39) in Figure 12 and choose the model from training step 1900.

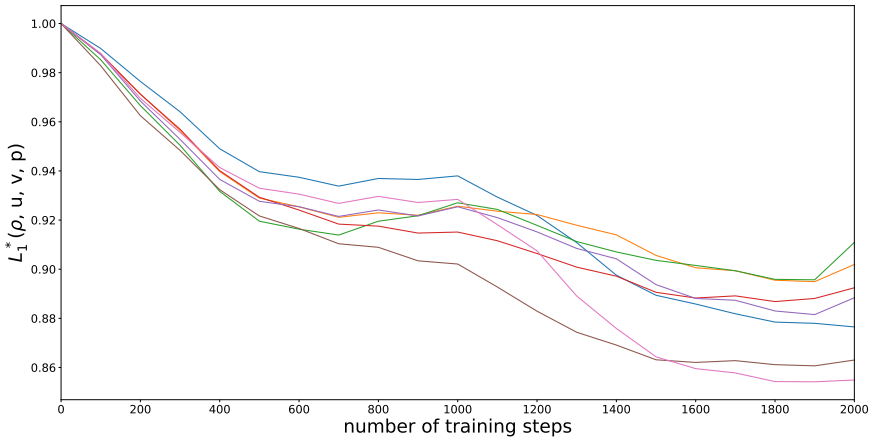


Figure 12: The values (39) for different validation problems evaluated each 100 training steps.

We test the trained WENO-DS on a test problem [44] with $\gamma = 1.4$, $T = 0.2$ and the initial condition

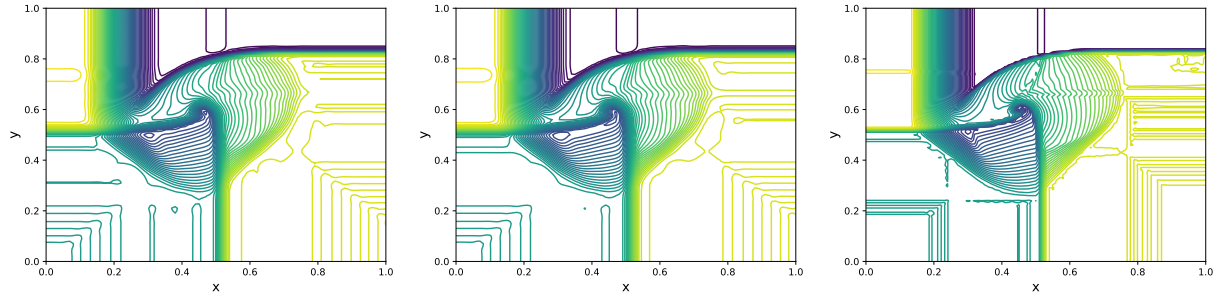
$$(\rho,u,v,p) = \begin{cases} (0.5313, 0.1, 0.1, 0.4) & x > 0.5 \quad \text{and} \quad y > 0.5, \\ (1.0222, -0.6179, 0.1, 1) & x < 0.5 \quad \text{and} \quad y > 0.5, \\ (0.8, 0.1, 0.1, 1) & x < 0.5 \quad \text{and} \quad y < 0.5, \\ (1, 0.1, 0.8276, 1) & x > 0.5 \quad \text{and} \quad y < 0.5. \end{cases} \tag{56}$$

We compare the results in Table 3 and the density contour plots can be found in Figure 13. As can be seen, WENO-DS outperforms WENO-Z and has smaller L_1 errors in all cases. In addition, we plot the absolute pointwise errors for the density solution and show them in Figure 14.

For another unseen test problem with the same initial configurations, we would again obtain analogous significant error improvements.

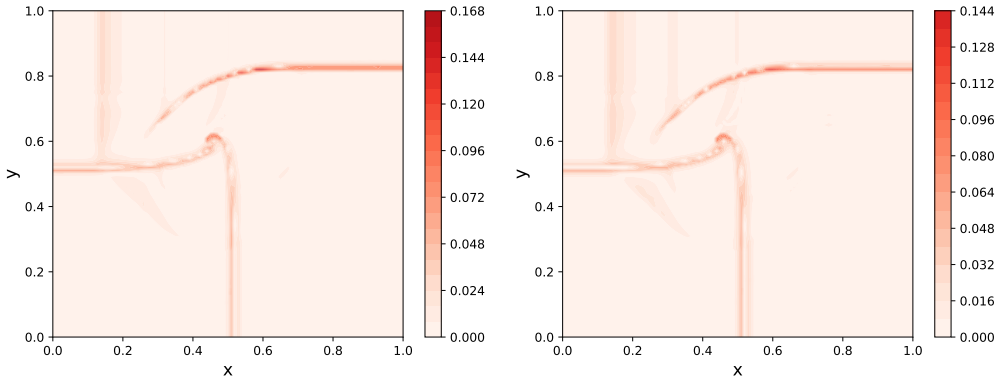
$I \times J$	50×50			100×100			200×200		
	WENO-Z	WENO-DS	ratio	WENO-Z	WENO-DS	ratio	WENO-Z	WENO-DS	ratio
ρ	0.010980	0.009877	1.11	0.004834	0.004327	1.12	0.001827	0.001624	1.12
u	0.012464	0.011287	1.10	0.005989	0.005326	1.12	0.002223	0.001913	1.16
v	0.015020	0.013932	1.08	0.006609	0.006172	1.07	0.002527	0.002298	1.10
p	0.010594	0.009644	1.10	0.004236	0.003820	1.11	0.001576	0.001392	1.13

Table 3: Comparison of L_1 error of WENO-Z and WENO-DS methods for the solution of the Euler system with the initial condition (56) for different spatial discretizations, $T = 0.2$.



(a) WENO-DS (b) WENO-Z (c) reference solution

Figure 13: Density contour plot for the solution of the Riemann problem with the initial condition (56), $I \times J = 100 \times 100$, $T = 0.2$.



(a) WENO-DS (b) WENO-Z

Figure 14: Absolute pointwise errors for density solution of the Riemann problem with the initial condition (56), $I \times J = 100 \times 100$, $T = 0.2$.

5.4. Configuration 11 and Configuration 19

In the previous sections, we trained three WENO-DS methods for three different types of configurations. We denote by WENO-DS (C2), WENO-DS (C3), and WENO-DS (C16) the methods from Sections 5.1, 5.2, and 5.3, respectively. In this section, we test these methods on the unseen problems containing the combination of rarefaction wave, shock wave, and

contact discontinuities. First, we consider Configuration 11 ($\overleftarrow{S}_{21}$, J^{+}_{32}, J^{+}_{34}, $\overleftarrow{S}_{41}$) with the test problem with $\gamma = 1.4$, $T = 0.3$, and the initial condition [44]

$$(\rho, u, v, p) = \begin{cases} (1, 0.1, 0, 1) & x > 0.5 \quad \text{and} \quad y > 0.5, \\ (0.5313, 0.8276, 0, 0.4) & x < 0.5 \quad \text{and} \quad y > 0.5, \\ (0.8, 0.1, 0, 0.4) & x < 0.5 \quad \text{and} \quad y < 0.5, \\ (0.5313, 0.1, 0.7276, 0.4) & x > 0.5 \quad \text{and} \quad y < 0.5. \end{cases} \tag{57}$$

Second, we test the models on the configuration (J^{+}_{21}, $\overleftarrow{S}_{32}$, J^{-}_{34}, $\overrightarrow{R}_{41}$) with the test problem with $\gamma = 1.4$, $T = 0.3$ and the initial condition [44]

$$(\rho, u, v, p) = \begin{cases} (1, 0, 0.3, 1) & x > 0.5 \quad \text{and} \quad y > 0.5, \\ (2, 0, -0.3, 1) & x < 0.5 \quad \text{and} \quad y > 0.5, \\ (1.0625, 0, 0.2145, 0.4) & x < 0.5 \quad \text{and} \quad y < 0.5, \\ (0.5197, 0, -0.4259, 0.4) & x > 0.5 \quad \text{and} \quad y < 0.5. \end{cases} \tag{58}$$

We summarize the results in Tables 4 and 5. As can be seen, the method trained on problems containing only rarefaction waves has the worst ability to generalize to unseen problems. On the other hand, by using methods trained on problems containing shocks or a combination of contact discontinuities, rarefaction, and shock waves, we obtain the error improvements even on unseen problems with different initial configurations. We would like to emphasize that the test problems in this section are far from the problems included in the training and validation sets. This is not only due to the choice of initial data, but also to the combination of rarefaction, shock waves and their direction, and positive and negative contact discontinuities.

	Configuration 11						
	WENO-Z	WENO-DS (C2)	ratio	WENO-DS (C3)	ratio	WENO-DS (C16)	ratio
ρ	0.007792	0.008000	0.97	**0.006783**	**1.15**	0.007538	1.03
u	0.008003	0.008701	0.92	0.007846	1.02	**0.007840**	**1.02**
v	0.007692	0.008300	0.93	**0.007161**	**1.07**	0.007370	1.04
p	0.005883	0.006467	0.91	**0.005115**	**1.15**	0.005776	1.02

Table 4: Comparison of L_1 error of WENO-Z and WENO-DS methods trained on data in Sections 5.1, 5.2 and 5.3 for the solution of the Euler system with the initial condition (57), $I \times J = 100 \times 100$, $T = 0.3$.

5.5. Bigger CNN and ability to generalize on unseen configurations

As can be seen from the previous Section 5.4, the models trained using the data from Section 5.2 and Section 5.3 are able to generalize very well to unseen problems. The WENO-DS method is able to properly localize the shocks and discontinuities, leading to a better

	Configuration 19						
	WENO-Z	WENO-DS (C2)	ratio	WENO-DS (C3)	ratio	WENO-DS (C16)	ratio
ρ	0.014844	0.014463	1.03	**0.013702**	**1.08**	0.013841	1.07
u	0.003749	**0.003562**	**1.05**	0.003689	1.02	0.003574	1.05
v	0.009891	0.009502	1.04	0.009791	1.01	**0.009245**	**1.07**
p	0.006123	0.005922	1.03	**0.005595**	**1.09**	0.005844	1.05

Table 5: Comparison of L_1 error of WENO-Z and WENO-DS methods trained on data in Sections 5.1, 5.2 and 5.3 for the solution of the Euler system with the initial condition (58), $I \times J = 100 \times 100$, $T = 0.3$.

numerical solution. Let us now increase the size of the CNN and use the structures shown in Figures 2b, increasing the size of the receptive field and the number of channels, and Figure 2c, increasing the number of channels and adding another CNN layer. Experimentally, we found that only increasing the size of the receptive field and the number of channels leads to similar results as described in the previous sections. In addition, increasing the receptive field makes the WENO-DS computationally more expensive. This is because we need to prepare wider inputs for the CNN, which also need to be projected onto the characteristic fields using left eigenvectors, and the matrix multiplications are more expensive here. On the other hand, if we use the CNN structure described in Figure 2c, we obtain a trained WENO-DS method that provides a much better numerical solution even for unseen problems with significantly different initial configurations.

Let us now train the method on two data sets. First, we use the dataset from the Section 5.2, train the CNN, and denote the final method as WENO-DS (C3c). Second, we train the CNN on the data set from the Section 5.3 and denote the final method as WENO-DS (C16c). We test the methods on even more different configurations and compare the results in Tables 6 and 7. We use boldface to indicate the configuration on which the method was actually trained.

With the number of configurations listed in the tables, we cover a wide range of possible combinations of contact discontinuities, rarefaction and shock waves. For all of them we use the test examples from the literature, see, e.g. [44]. We treat the possibility with four contact discontinuities with Configuration 6: J^-_{21}, J^-_{32}, J^-_{34}, J^-_{41}, two contact discontinuities and two rarefaction waves with Configuration 8: $\overleftarrow{R}_{21}$, J^-_{32}, J^-_{34}, $\overleftarrow{R}_{41}$, two shock waves and two contact discontinuities using Configuration 14: J^+_{21}, $\overleftarrow{S}_{32}$, J^-_{34}, $\overleftarrow{S}_{41}$ and Configuration 11 from Section 5.4. Finally, the combination of contact discontinuities, rarefaction, and shock waves using Configuration 18: J^+_{21}, $\overleftarrow{S}_{32}$, J^+_{34}, $\overrightarrow{R}_{41}$, and Configuration 19 form the Section 5.4.

As one can see, we obtain significant error improvements with both methods. Comparing both methods, even better results are obtained when the CNN was trained on a data set from Section 5.2 on a configuration with four shock waves. Compared to the Table 2, the improvement for Configuration 3 is smaller but still significant. However, the method is able to generalize much better to unknown configurations. For example, for Configuration 14, we obtain an average improvement rate of 1.30 for all four variables. In addition, we use

WENO-DS (C3c) to illustrate the density contour plots and absolute pointwise errors in Figures 15, 16, and 17. Here we also show the difference from Configuration 3, with which the model was actually trained.

The WENO-DS (C3c) method achieves large error improvements not only for problems from the same configuration, but also for problems from significantly different configurations. Since we used a larger CNN, the question is what is the actual numerical cost of these improvements. We illustrate the computational costs in Figure 18. As can be seen from the shift of the red dots to the right, the method involves larger computational costs. However, it is still more effective or not worse than the original method in most cases. We would like to emphasize that here we are comparing results with significantly different initial problems than those on which the method was actually trained. Machine learning models are generally not expected to give such better results on unseen problems.

	Configuration 3			Configuration 6			Configuration 8			Configuration 11		
	WENO-Z	**WENO-DS**	**ratio**	WENO-Z	WENO-DS	ratio	WENO-Z	WENO-DS	ratio	WENO-Z	WENO-DS	ratio
ρ	**0.019232**	**0.015033**	**1.28**	0.038616	0.032696	1.18	0.005711	0.004975	1.15	0.007792	0.006316	1.23
u	**0.019588**	**0.016359**	**1.20**	0.019662	0.016144	1.22	0.008488	0.007396	1.15	0.008003	0.006487	1.23
v	**0.019588**	**0.016359**	**1.20**	0.022582	0.018951	1.19	0.008488	0.007396	1.15	0.007692	0.006282	1.22
p	**0.018666**	**0.015214**	**1.23**	0.010525	0.008821	1.19	0.005350	0.004844	1.10	0.005883	0.004813	1.22

	Configuration 14			Configuration 18			Configuration 19		
	WENO-Z	WENO-DS	ratio	WENO-Z	WENO-DS	ratio	WENO-Z	WENO-DS	ratio
ρ	0.013169	0.010333	1.27	0.014918	0.012519	1.19	0.014844	0.012390	1.20
u	0.004835	0.003732	1.30	0.003534	0.003063	1.15	0.003749	0.003339	1.12
v	0.021299	0.016512	1.29	0.010315	0.009077	1.14	0.009891	0.008641	1.14
p	0.034996	0.026008	1.35	0.006795	0.005961	1.14	0.006123	0.005393	1.14

Table 6: Comparison of L_1 error of WENO-Z and WENO-DS (C3c) methods for the solution of the Euler system with various initial configurations, $I \times J = 100 \times 100$.

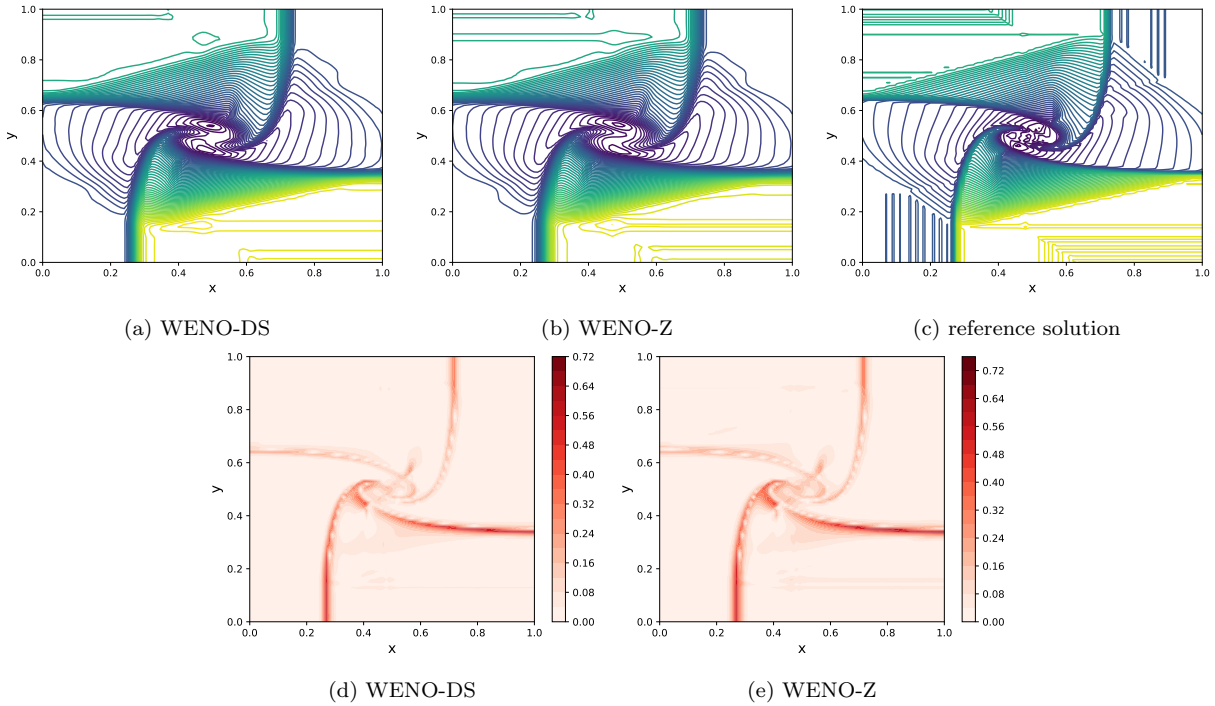
	Configuration 16			Configuration 6			Configuration 8			Configuration 11		
	WENO-Z	**WENO-DS**	**ratio**	WENO-Z	WENO-DS	ratio	WENO-Z	WENO-DS	ratio	WENO-Z	WENO-DS	ratio
ρ	**0.004834**	**0.004127**	**1.17**	0.038616	0.036329	1.06	0.005711	0.004777	1.20	0.007792	0.007695	1.01
u	**0.005989**	**0.004981**	**1.20**	0.019662	0.019575	1.00	0.008488	0.007056	1.20	0.008003	0.007824	1.02
v	**0.006609**	**0.005776**	**1.14**	0.022582	0.019974	1.13	0.008488	0.007056	1.20	0.007692	0.007482	1.03
p	**0.004236**	**0.003663**	**1.16**	0.010525	0.010216	1.03	0.005350	0.004624	1.16	0.005883	0.006295	0.93

	Configuration 14			Configuration 18			Configuration 19		
	WENO-Z	WENO-DS	ratio	WENO-Z	WENO-DS	ratio	WENO-Z	WENO-DS	ratio
ρ	0.013169	0.011718	1.12	0.014918	0.013447	1.11	0.014844	0.013198	1.12
u	0.004835	0.004042	1.20	0.003534	0.002975	1.19	0.003749	0.003256	1.15
v	0.021299	0.020330	1.05	0.010315	0.009302	1.11	0.009891	0.008796	1.12
p	0.034996	0.036038	0.97	0.006795	0.006535	1.04	0.006123	0.005752	1.06

Table 7: Comparison of L_1 error of WENO-Z and WENO-DS (C16c) methods for the solution of the Euler system with various initial configurations, $I \times J = 100 \times 100$.

6. Conclusion

In this paper we introduced a novel approach, WENO-DS, which leverages the power of deep learning to enhance the performance of the well-established Weighted Essentially



(a) WENO-DS (b) WENO-Z (c) reference solution

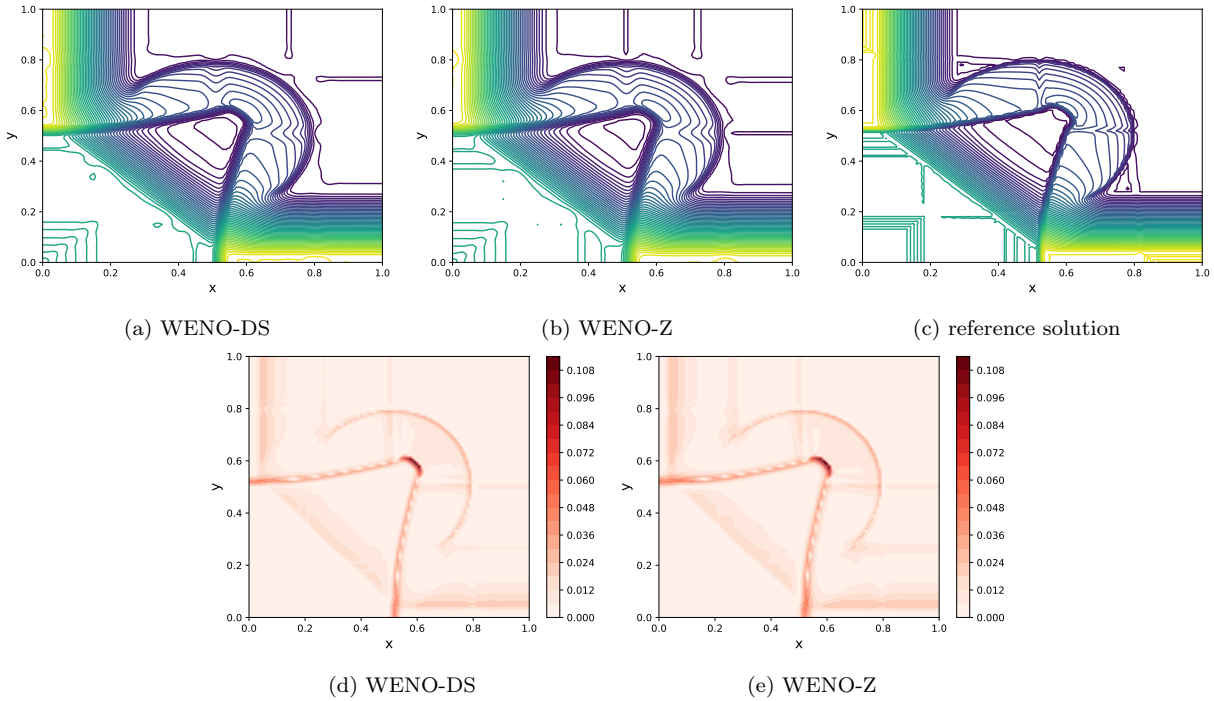
(d) WENO-DS (e) WENO-Z

Figure 15: Density contour plots and absolute pointwise errors for the solution of the Riemann problem with initial Configuration 6, $I \times J = 100 \times 100$, $T = 0.3$.

Non-Oscillatory (WENO) scheme in the context of solving hyperbolic conservation laws, particularly exemplified by the two-dimensional Euler equations of gas dynamics. By seamlessly integrating deep learning techniques into the WENO algorithm, we have successfully improved the accuracy of numerical solutions, particularly in regions near abrupt shocks. Unlike previous attempts at incorporating deep learning into numerical methods, this approach stands out by eliminating the need for additional post-processing steps, ensuring consistency throughout.

This study demonstrates the superiority of the WENO-DS approach through an extensive examination of various test problems, including scenarios featuring shocks and rarefaction waves. The results consistently showcase the newfound capabilities of the approach, outperforming traditional fifth-order WENO schemes, especially when dealing with challenges like excessive diffusion or overshooting around shocks.

The introduction of machine learning into the realm of solving partial differential equations (PDEs) has brought about promising improvements in numerical methods. However, it is crucial to strike a balance between these data-driven insights and the foundational mathematical principles underpinning the numerical scheme. This study successfully maintains this equilibrium, building upon the physical principles of the Euler equations while incorporating deep learning enhancements.



(a) WENO-DS (b) WENO-Z (c) reference solution

(d) WENO-DS (e) WENO-Z

Figure 16: Density contour plots and absolute pointwise errors for the solution of the Riemann problem with initial Configuration 8, $I \times J = 100 \times 100$, $T = 0.25$.

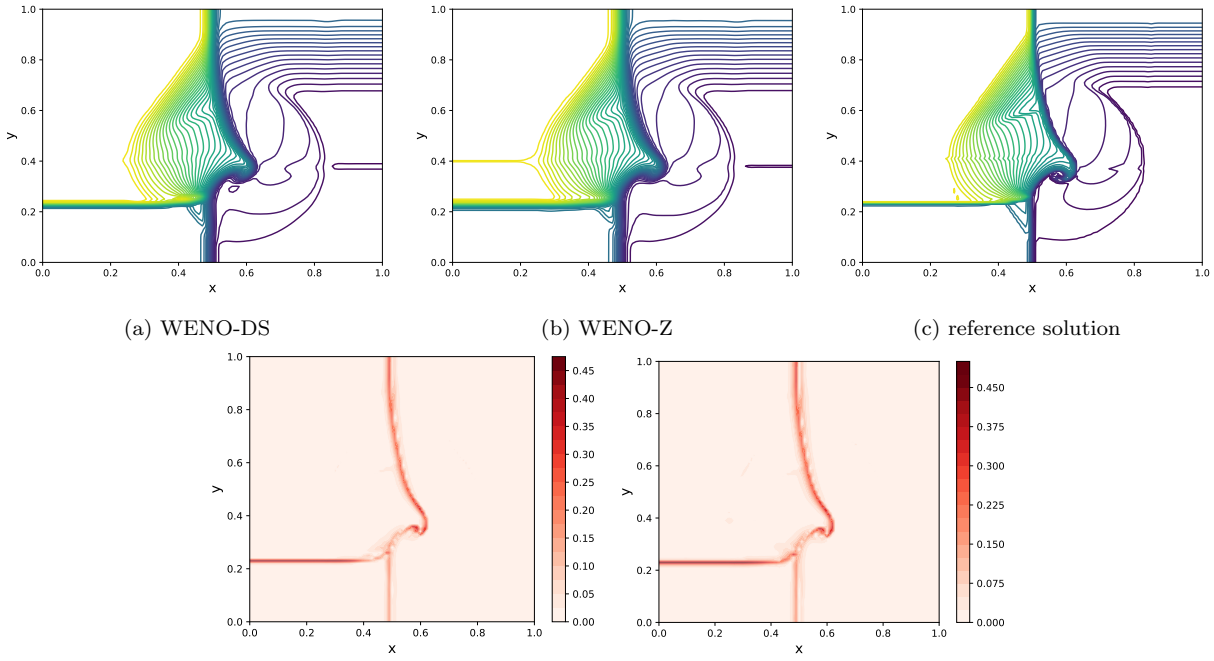
In summary, the WENO-DS approach represents a significant advancement in the field of numerical methods for hyperbolic conservation laws, where the incorporation of deep learning techniques has not only enhanced the accuracy but also improved the qualitative behavior of solutions, both in smooth regions and near discontinuities. This research paves the way for future developments in the intersection of traditional numerical methods and machine learning, offering a promising direction for further advancements in solving complex PDEs like the Euler equations.

References

[1] M. Raissi, P. Perdikaris, G. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.

[2] A. D. Jagtap, Z. Mao, N. Adams, G. E. Karniadakis, Physics-informed neural networks for inverse problems in supersonic flows, J. Comput. Phys. 466 (2022) 111402.

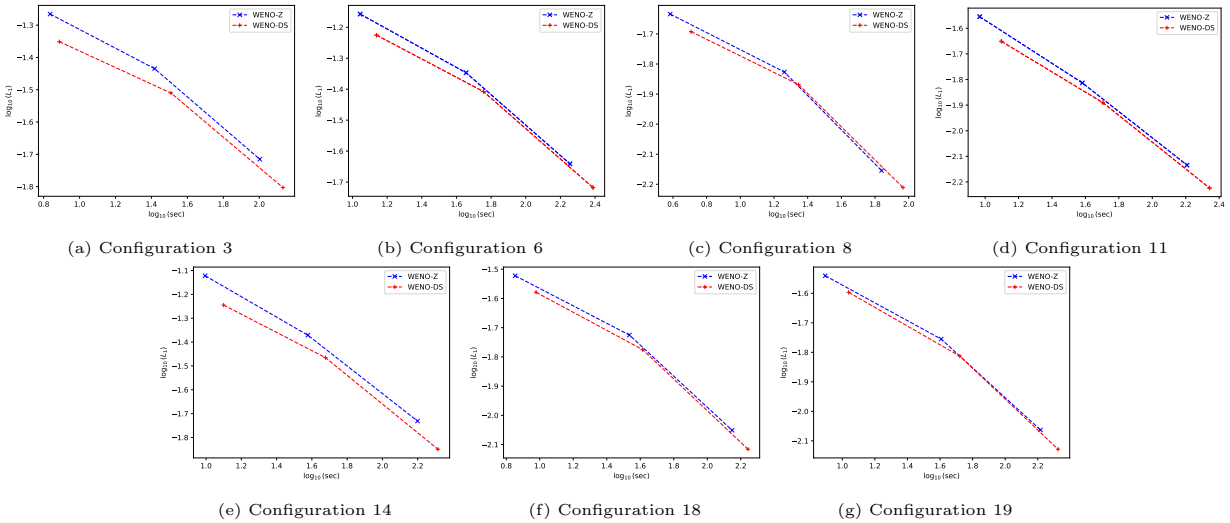
[3] A. D. Jagtap, D. Mitsotakis, G. E. Karniadakis, Deep learning of inverse water waves



(a) WENO-DS (b) WENO-Z (c) reference solution

(d) WENO-DS (e) WENO-Z

Figure 17: Density contour plots and absolute pointwise errors for the solution of the Riemann problem with initial Configuration 19, $I \times J = 100 \times 100$, $T = 0.3$.



(a) Configuration 3 (b) Configuration 6 (c) Configuration 8 (d) Configuration 11

(e) Configuration 14 (f) Configuration 18 (g) Configuration 19

Figure 18: Comparison of computational cost against L_1-error of the solution of Riemann problem with various initial configurations using WENO-Z and WENO-DS (C3c) methods.

problems using multi-fidelity data: Application to Serre–Green–Naghdi equations, Ocean Engineering 248 (2022) 110775.

[4] A. D. Jagtap, E. Kharazmi, G. E. Karniadakis, Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems, Comput. Meth. Appl. Mech. Engrg. 365 (2020) 113028.

[5] A. D. Jagtap, G. E. Karniadakis, Extended physics-informed neural networks (XPINNs): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations, Commun. Comput. Phys. 28 (5) (2020) 2002–2041.

[6] K. Shukla, A. D. Jagtap, G. E. Karniadakis, Parallel physics-informed neural networks via domain decomposition, J. Comput. Phys. 447 (2021) 110683.

[7] M. Penwarden, A. D. Jagtap, S. Zhe, G. E. Karniadakis, R. M. Kirby, A unified scalable framework for causal sweeping strategies for physics-informed neural networks (PINNs) and their temporal decompositions, J. Comput. Phys. (2023) 112464.

[8] T. De Ryck, A. D. Jagtap, S. Mishra, Error estimates for physics informed neural networks approximating the Navier-Stokes equations, arXiv preprint arXiv:2203.09346 (2022).

[9] V. Grimm, A. Heinlein, A. Klawonn, Learning the solution operator of two-dimensional incompressible Navier-Stokes equations using physics-aware convolutional neural networks, arXiv preprint arXiv:2308.02137 (2023).

[10] L. Drozda, P. Mohanamuraly, L. Cheng, C. Lapeyre, G. Daviller, Y. Realpe, A. Adler, G. Staffelbach, T. Poinsot, Learning an optimised stable Taylor-Galerkin convection scheme based on a local spectral model for the numerical error dynamics, J. Comput. Phys. (2023) 112430.

[11] Z. Mao, A. D. Jagtap, G. E. Karniadakis, Physics-informed neural networks for high-speed flows, Comput. Meth. Appl. Mech. Engrg. 360 (2020) 112789.

[12] K. van der Sande, N. Flyer, B. Fornberg, Accelerating explicit time-stepping with spatially variable time steps through machine learning, J. Sci. Comput. 96 (1) (2023) 31.

[13] T. Kossaczká, M. Ehrhardt, M. Günther, Deep FDM: Enhanced finite difference methods by deep learning, Franklin Open (2023) 100039.

[14] M. Crandall, A. Majda, Monotone difference approximations for scalar conservation laws, Math. Comput. 34 (1980) 1–21.

[15] A. Harten, High resolution schemes for hyperbolic conservation laws, J. Comput. Phys. 49 (3) (1983) 357–393.

[16] A. Harten, B. Engquist, S. Osher, S. Chakravarthy, Uniformly high order accurate essentially non-oscillatory schemes, III, in: M. Hussaini, B. van Leer, J. Van Rosendale (Eds.), Upwind and High-Resolution Schemes, Springer, 1987, pp. 218–290.

[17] G.-S. Jiang, C.-W. Shu, Efficient implementation of weighted ENO schemes, J. Comput. Phys. 126 (1) (1996) 202–228.

[18] C.-W. Shu, Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws, in: A. Quarteroni (Ed.), Advanced Numerical Approximation of Nonlinear Hyperbolic Equations: Lectures given at the 2nd Session of the Centro Internazionale Matematico Estivo (C.I.M.E.) held in Cetraro, Italy, June 23–28, 1997, Springer, Berlin, 1998, pp. 325–432.

[19] J. Qiu, C.-W. Shu, Hermite WENO schemes and their application as limiters for Runge-Kutta discontinuous Galerkin method: one-dimensional case, J. Comput. Phys. 193 (1) (2004) 115–135.

[20] J. Qiu, C.-W. Shu, Hermite WENO schemes and their application as limiters for Runge-Kutta discontinuous Galerkin method II: Two dimensional case, Computers & Fluids 34 (6) (2005) 642–663.

[21] S. Pirozzoli, Conservative hybrid compact-WENO schemes for shock-turbulence interaction, J. Comput. Phys. 178 (1) (2002) 81–117.

[22] D. J. Hill, D. I. Pullin, Hybrid tuned center-difference-WENO method for large eddy simulations in the presence of strong shocks, J. Comput. Phys. 194 (2) (2004) 435–450.

[23] A. D. Jagtap, R. Kumar, Kinetic theory based multi-level adaptive finite difference WENO schemes for compressible Euler equations, Wave Motion 98 (2020) 102626.

[24] A. K. Henrick, T. D. Aslam, J. M. Powers, Mapped weighted essentially non-oscillatory schemes: achieving optimal order near critical points, J. Comput. Phys. 207 (2) (2005) 542–567.

[25] R. Borges, M. Carmona, B. Costa, W. S. Don, An improved weighted essentially non-oscillatory scheme for hyperbolic conservation laws, J. Comput. Phys. 227 (6) (2008) 3191–3211.

[26] M. Castro, B. Costa, W. S. Don, High order weighted essentially non-oscillatory WENO-Z schemes for hyperbolic conservation laws, J. Comput. Phys. 230 (5) (2011) 1766–1792.

[27] Y. Ha, C. H. Kim, Y. J. Lee, J. Yoon, An improved weighted essentially non-oscillatory scheme with a new smoothness indicator, J. Comput. Phys. 232 (1) (2013) 68–86.

[28] J. Zhu, J. Qiu, A new fifth order finite difference WENO scheme for solving hyperbolic conservation laws, J. Comput. Phys. 318 (2016) 110–121.

[29] S. Rathan, R. Kumar, A. D. Jagtap, L1-type smoothness indicators based WENO scheme for nonlinear degenerate parabolic equations, Appl. Math. Comput. 375 (2020) 125112.

[30] B. Stevens, T. Colonius, Enhancement of shock-capturing methods via machine learning, Theor. Comput. Fluid Dyn. 34 (3) (2020) 483–496.

[31] R. Wang, R. J. Spiteri, Linear instability of the fifth-order WENO method, SIAM J. Numer. Anal. 45 (5) (2007) 1871–1901.

[32] T. Kossaczká, M. Ehrhardt, M. Günther, Enhanced fifth order WENO shock-capturing schemes with deep learning, Res. Appl. Math. 12 (2021) 100201.

[33] T. Kossaczká, M. Ehrhardt, M. Günther, A neural network enhanced weighted essentially non-oscillatory method for nonlinear degenerate parabolic equations, Phys. Fluids 34 (2) (2022) 026604.

[34] T. Kossaczká, M. Ehrhardt, M. Günther, A deep smoothness WENO method with applications in option pricing, in: M. Ehrhardt, M. Günther (Eds.), Progress in Industrial Mathematics at ECMI 2021, Springer, 2022, pp. 417–423.

[35] A. D. Jagtap, K. Kawaguchi, G. E. Karniadakis, Adaptive activation functions accelerate convergence in deep and physics-informed neural networks, J. Comput. Phys. 404 (2020) 109136.

[36] A. D. Jagtap, K. Kawaguchi, G. Em Karniadakis, Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks, Proc. Royal Soc. A 476 (2239) (2020) 20200334.

[37] A. D. Jagtap, Y. Shin, K. Kawaguchi, G. E. Karniadakis, Deep Kronecker neural networks: A general framework for neural networks with adaptive activation functions, Neurocomputing 468 (2022) 165–180.

[38] A. D. Jagtap, G. E. Karniadakis, How important are activation functions in regression and classification? a survey, performance comparison, and future directions, J. Machine Learn. Model. Comput. 4 (1) (2023).

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

[40] C. W. Schulz-Rinne, Classification of the Riemann problem for two-dimensional gas dynamics, SIAM J. Math. Anal. 24 (1) (1993) 76–88.

[41] C. W. Schulz-Rinne, J. P. Collins, H. M. Glaz, Numerical solution of the Riemann problem for two-dimensional gas dynamics, SIAM J. Sci. Comput. 14 (6) (1993) 1394–1414.

[42] T. Chang, G.-Q. Chen, S. Yang, On the 2-D Riemann problem for the compressible Euler equations. I. Interaction of shocks and rarefaction waves, Discr. Contin. Dynam. Syst. 1 (4) (1995) 555–584.

[43] T. Chang, G.-Q. Chen, S. Yang, On the 2-D Riemann problem for the compressible Euler equations II. Interaction of contact discontinuities, Discr. Contin. Dynam. Syst. 6 (2) (1999) 419–430.

[44] A. Kurganov, E. Tadmor, Solution of two-dimensional Riemann problems for gas dynamics without Riemann problem solvers, Numer. Meth. Part. Diff. Eqs. 18 (5) (2002) 584–608.

[45] T. Zhang, Y. X. Zheng, Conjecture on the structure of solutions of the riemann problem for two-dimensional gas dynamics systems, SIAM J. Math. Anal. 21 (3) (1990) 593–630.

[46] C.-W. Shu, T. A. Zang, G. Erlebacher, D. Whitaker, S. Osher, High-order ENO schemes applied to two-and three-dimensional compressible flow, Appl. Numer. Math. 9 (1) (1992) 45–71.