Bergische Universität Wuppertal

Fakultät für Mathematik und Naturwissenschaften

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

Preprint BUW-IMACM 21/25
(updated version of Preprint 21/02)

Tatiana Kossacká, Matthias Ehrhardt and Michael Günther

Enhanced fifth order WENO Shock-Capturing Schemes with Deep Learning

August 2, 2021

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

Enhanced fifth order WENO Shock-Capturing Schemes with Deep Learning

Tatiana Kossaczká*, Matthias Ehrhardt, Michael Günther

Institute of Mathematical Modelling, Analysis and Computational Mathematics (IMACM), Chair of Applied Mathematics and Numerical Analysis, Bergische Universität Wuppertal, Gaußstraße 20, 42119 Wuppertal, Germany

Abstract

In this paper we enhance the well-known fifth order WENO shock-capturing scheme by using deep learning techniques. This fine-tuning of an existing algorithm is implemented by training a rather small neural network to modify the smoothness indicators of the WENO scheme in order to improve the numerical results especially at discontinuities. In our approach no further post-processing is needed to ensure the consistency of the method. Moreover, the formal order of accuracy of the resulting scheme can be proven.

We demonstrate our findings with the inviscid Burgers' equation, the Buckley-Leverett equation and the 1-D Euler equations of gas dynamics. Hereby we investigate the classical Sod problem and the Lax problem and show that our novel method outperforms the classical fifth order WENO schemes in simulations where the numerical solution is too diffusive or tends to overshoot at shocks. Finally, the straight-forward extension of the method to two-dimensional problems is included and illustrated using the 2D Burgers' equation.

Keywords: Weighted essentially non-oscillatory method, Hyperbolic conservation laws, Smoothness indicators, Deep Learning, Neural Networks
2000 MSC: 65M06, 68T07, 76L05

*corresponding author
Email addresses: kossaczka@uni-wuppertal.de (Tatiana Kossaczká), ehrhardt@uni-wuppertal.de (Matthias Ehrhardt), guenther@uni-wuppertal.de (Michael Günther)

1. Introduction

Typically, numerical fluid mechanics deals with nonlinear hyperbolic partial differential equations (PDEs). In its simplest one-dimensional form, these equations can be represented as

$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} = 0, \qquad t > 0, \tag{1}$$

where x represents space, t denotes time, $u(x,t)$ is conserved quantity and $f(u(x,t))$ is its flux. It is well-known that discontinuities may develop after a finite time regardless of the smoothness of the initial or boundary data. Hence, suitable numerical methods must be designed for these problems, especially to approximate discontinuous solutions. First in 1980, Crandall and Majda [1] proposed the class of *monotone schemes* that are nonlinearly stable in the L_1 norm and satisfy certain entropy conditions. It can be proven that the corresponding solutions converge to bounded variation entropy solutions including error estimates. However, these schemes are only first order accurate and by the fundamental Godunov theorem [2] it is known that one has to consider nonlinear non-oscillatory schemes to overcome this accuracy barrier.

In this direction so-called *shock-capturing schemes* were developed that were able to resolve sharply a shock or a steep gradient region without introducing too much diffusion or overshoot behaviour [3]. Additionally, to remedy the above mentioned drawback, at regions with smooth flow these schemes exhibit a rather high order of convergence. The well-known representative of this class of methods are the *essentially non-oscillatory (ENO) schemes* [4] with high order accuracy in smooth regions and sharply resolving shocks in an essentially non-oscillatory way using a *smoothness indicator function*, see e.g. [5]. Later on Jiang and Shu [6] further improved these schemes and proposed a weighted ENO (in the sequel abbreviated with WENO-JS) scheme, that is still regarded as a state-of-the-art solution approach.

In order to achieve higher order accuracy for WENO schemes, the stencil for the spatial reconstruction needs to be enlarged and to keep it in a compact size Qiu and Shu [7, 8] developed the *Hermite WENO (HWENO)* schemes. In order to further increase the efficiency, Pirozzoli [9] developed a *hybrid compact-WENO scheme* using up-wind schemes in the smooth regions. Alternatively, Hill and Pullin [10] designed another hybrid WENO scheme, combining special centered difference schemes with WENO methods, cf. [11, 12] for most recent approaches.

Subsequently, different new strategies were developed by modifying the WENO-JS schemes, i.e. by altering by smoothness indicators [13, 14, 15, 16, 17, 18] or by modifying the nonlinear weights [19]. Besides, another goal in optimizing these schemes was to minimize the dispersion error (dispersion-relation-preserving (DRP) schemes) [20, 21], also combined with the WENO approach leading to optimized WENO (OWENO) schemes [22]. Since classical WENO methods are too dissipative for direct numerical simulations (DNS) of turbulence, a goal was to reduce the dissipation by including an automatic dissipation adjustment [23]. In 2016 Fu, Hu and Adams [24] proposed a family of high-order *targeted ENO (TENO) schemes* that are particularly suitable for DNS. These methods reduce the numerical dissipation by an ENO-like stencil selection and increase the numerical robustness by assembling a set of low-order candidate stencils with increasing width, see also the extensions in [25, 26]. For a detailed review on WENO schemes we refer to [27].

Recently, machine learning was widely used to compute the solution of PDEs. We refer to [28, 29, 30], where the neural network algorithm is used to approximate a solution of a particular PDE problem. Following that approach, the solution of a particular PDE is a result of a neural network training procedure. Another idea is to improve a specific numerical scheme using neural networks. The training of a neural network is made offline and results in a new numerical scheme applicable to a wider class of problems. This idea was recently used by Beck et al. [31] for discontinuous Galerkin methods or in [32] for learning iterative PDE solvers and we also follow this approach. Bar-Sinai et al. [33] use neural networks and learn from high resolution solutions to approximate a spatial derivative on a coarse grid. We refer the reader also to [34, 35, 36] for other work in this direction.

In [37] deep reinforcement learning is applied to design a new numerical scheme to solve hyperbolic conservation laws. Authors apply their method for solving of the Burgers' equation and compare their results with the standard WENO scheme. The recent work of Stevens and Colonius [38] introduces new WENO-NN scheme based on neural network algorithm. In their work, the finite-volume coefficients of the WENO-JS scheme are perturbed, while maintaining the original smoothness indicators and nonlinear weights. However, the resulting scheme presented in their paper has only first order of accuracy. Another neural network based WENO scheme was developed by Liu and Wen [39], where the new smoothness indicators are an output of the neural network algorithm. However, in this case the formal order of accu-

racy of the reconstruction of the resulting method can neither be analytically proven, nor guaranteed. For a detailed discussion on the formal order of accuracy (in contrast to the term "convergence") for WENO methods, see e.g. [40, 41].

We implement in our work another WENO extension based on deep learning. This approach is used to improve the classic WENO-JS and WENO-Z [13] scheme in this paper, but could be efficiently applied also to other WENO methods. For this purpose we will train a rather small neural network to perturb the smoothness indicator functions of the WENO-JS scheme. As we do not develop any new smoothness indicators as in [39], but only their multiplicative perturbations, we are able to prove the formal order of accuracy of the resulting scheme. We call this new scheme WENO-DS (Deep Smoothness), as we modify the smoothness indicators using deep neural networks. This scheme has less diffusion and less overshoot in shocks than the WENO-JS and the WENO-Z scheme, while maintaining high order accuracy in smooth regions.

Finally, let us note that the use of WENO methods is not limited to hyperbolic PDEs, see e.g. [42] for an application to parabolic PDEs in finance.

The paper is structured as follows. In Section 2 we introduce the WENO-JS and WENO-Z schemes under consideration in detail. Next, in Section 3 we introduce our deep learning approach, where neural networks are used to further improve WENO methods without any post-processing. The corresponding proofs of the formal order of accuracy for two WENO schemes are given in Section 4. Then we present our numerical results in Section 5, which illustrate the improvements of our proposed method. Finally, we conclude our work in Section 6.

2. The WENO Scheme

Let $\{I_i\}$ be the partition of a spatial domain with the i-th cell $I_i = [x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}]$. We consider a uniform grid defined by the points $x_i = x_0 + i\Delta x$, $i = 0, \dots, N$, which are the centers of the cells with cell boundaries defined by $x_{i+\frac{1}{2}} = x_i + \frac{\Delta x}{2}$. The value of a function f at x_i is indicated by $f_i = f(x_i)$.

The spatial discretization of one-dimensional conservation laws (1) yields a system of ordinary differential equations (ODEs) and the resulting semi-discrete scheme is

$$\frac{du_i}{dt} = -\frac{1}{\Delta x}\big(\hat{f}_{i+\frac{1}{2}} - \hat{f}_{i-\frac{1}{2}}\big), \tag{2}$$

where $\hat{f}$ is a numerical approximation of the flux function f. Following [6], if we define a function h implicitly by

$$f\big(u(x)\big) = \frac{1}{\Delta x}\int_{x-\frac{\Delta x}{2}}^{x+\frac{\Delta x}{2}} h(\xi)\,d\xi, \tag{3}$$

then (2) is approximated by

$$f'\big(u(x_i)\big) = \frac{1}{\Delta x}\big(h_{i+\frac{1}{2}} - h_{i-\frac{1}{2}}\big), \qquad h_{i\pm\frac{1}{2}} = h(x_{i\pm\frac{1}{2}}), \tag{4}$$

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

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

Further, the *flux splitting method* is applied, thus we write

$$f(u) = f^+(u) + f^-(u), \quad \text{where} \quad \frac{df^+(u)}{du} \geq 0 \quad \text{and} \quad \frac{df^-(u)}{du} \leq 0. \tag{6}$$

The numerical flux $\hat{f}_{i\pm\frac{1}{2}}$ is then represented by $\hat{f}_{i\pm\frac{1}{2}} = \hat{f}^+_{i\pm\frac{1}{2}} + \hat{f}^-_{i\pm\frac{1}{2}}$ and the final scheme is formed as

$$\frac{du_i}{dt} = -\frac{1}{\Delta x}\left[\left(\hat{f}^+_{i+\frac{1}{2}} - \hat{f}^+_{i-\frac{1}{2}}\right) + \left(\hat{f}^-_{i+\frac{1}{2}} - \hat{f}^-_{i-\frac{1}{2}}\right)\right]. \tag{7}$$

Next, only the construction of $\hat{f}^+_{i\pm\frac{1}{2}}$ is considered. The negative part can be then obtained using symmetry (see e.g. [43]).

2.1. Fifth order WENO scheme

For a construction of $\hat{f}_{i+\frac{1}{2}}$, the fifth order WENO method uses a 5-point stencil

$$S(i) = \{x_{i-2}, \ldots, x_{i+2}\} \tag{8}$$

divided into three candidate substencils, which are given by

$$S^m(i) = \{x_{i+m-2}, x_{i+m-1}, x_{i+m}\}, \quad m = 0, 1, 2. \tag{9}$$

To form the numerical flux over the entire 5-point stencil, the numerical flux for each of these substencils $\hat{f}^m_{i+\frac{1}{2}} = h_{i+\frac{1}{2}} + O(\Delta x^3)$ is calculated. These fluxes are then averaged in such a way, that fifth order accuracy is ensured in the

smooth regions. In regions with discontinuities, the weights should partly remove the contribution of these stencils so that the solution near the shock can be approximated in more stable manner.

Let $\hat{f}^m(x)$ be the polynomial approximation of $h(x)$ on each of the substencils (9). Then, evaluated at $i+\frac{1}{2}$ we obtain

$$\hat{f}^m(x_{i+\frac{1}{2}}) = \hat{f}^m_{i+\frac{1}{2}} = \sum_{j=0}^{2} c_{m,j}\, f_{i+m-2+j}, \quad i = 0, \ldots, N, \tag{10}$$

where $c_{m,j}$ are the Lagrangian interpolation coefficients, dependent on m (see [6]). They take an explicit form

$$\begin{aligned} \hat{f}^0_{i+\frac{1}{2}} &= \frac{2f_{i-2} - 7f_{i-1} + 11f_i}{6}, \\ \hat{f}^1_{i+\frac{1}{2}} &= \frac{-f_{i-1} + 5f_i + 2f_{i+1}}{6}, \\ \hat{f}^2_{i+\frac{1}{2}} &= \frac{2f_i + 5f_{i+1} - f_{i+2}}{6}, \end{aligned} \tag{11}$$

and the numerical fluxes $\hat{f}^m_{i-\frac{1}{2}}$ can be obtained by shifting each index by -1. Using the Taylor series expansion it can be shown that:

$$\begin{aligned} \hat{f}^0_{i\pm\frac{1}{2}} &= h_{i\pm\frac{1}{2}} - \frac{1}{4} f_{xxx}(0)\Delta x^3 + O(\Delta x^4), \\ \hat{f}^1_{i\pm\frac{1}{2}} &= h_{i\pm\frac{1}{2}} + \frac{1}{12} f_{xxx}(0)\Delta x^3 + O(\Delta x^4), \\ \hat{f}^2_{i\pm\frac{1}{2}} &= h_{i\pm\frac{1}{2}} - \frac{1}{12} f_{xxx}(0)\Delta x^3 + O(\Delta x^4), \end{aligned} \tag{12}$$

where we used the short notation for the derivatives $f_x = f'(u)u_x$, etc.. Doing so, we obtain the general form of these expressions

$$\hat{f}^m_{i\pm\frac{1}{2}} = h_{i\pm\frac{1}{2}} + A_m \Delta x^3 + O(\Delta x^4), \tag{13}$$

with A_m being independent of Δx.

Then, the convex combination of the interpolated values $\hat{f}^m_{i\pm\frac{1}{2}}$ given by

$$\hat{f}_{i\pm\frac{1}{2}} = \sum_{j=0}^{2} \omega_m\, \hat{f}^m_{i\pm\frac{1}{2}} \tag{14}$$

yields the WENO approximation of the value $h_{i\pm\frac{1}{2}}$, where ω_m are the nonlinear weights defined as, cf. [6]

$$\omega_m^{JS} = \frac{\alpha_m^{JS}}{\sum_{i=0}^{2} \alpha_i^{JS}}, \qquad \text{where} \qquad \alpha_m^{JS} = \frac{d_m}{(\epsilon + \beta_m)^2}. \tag{15}$$

The scheme using these nonlinear weights is denoted as the WENO-JS scheme. The parameter ϵ guarantees that the denominator does not become zero, and the coefficients d_0, d_1 and d_2 are called ideal weights, which would form the upstream fifth order central scheme for the 5-point stencil and satisfy (5). Their values are:

$$d_0 = \frac{1}{10}, \quad d_1 = \frac{6}{10}, \quad d_2 = \frac{3}{10}. \tag{16}$$

The parameter β_m is called the *smoothness indicator* and is analyzed in the next section.

2.2. Smoothness Indicators

The role of smoothness indicators is to measure the regularity of the polynomial approximation $\hat{f}^m(x)$ in each of three substencils. As developed in [6], they are defined as:

$$\beta_m = \sum_{q=1}^{2} \Delta x^{2q-1} \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \Big(\frac{d^q \hat{f}^m(x)}{dx^q} \Big)^2 dx. \tag{17}$$

Corresponding to the flux approximation $\hat{f}_{i+\frac{1}{2}}$ they take an explicit form

$$\begin{aligned}
\beta_0 &= \frac{13}{12}\big(f_{i-2} - 2f_{i-1} + f_i\big)^2 + \frac{1}{4}\big(f_{i-2} - 4f_{i-1} + 3f_i\big)^2, \\
\beta_1 &= \frac{13}{12}\big(f_{i-1} - 2f_i + f_{i+1}\big)^2 + \frac{1}{4}\big(-f_{i-1} + f_{i+1}\big)^2, \\
\beta_2 &= \frac{13}{12}\big(f_i - 2f_{i+1} + f_{i+2}\big)^2 + \frac{1}{4}\big(3f_i - 4f_{i+1} + f_{i+2}\big)^2,
\end{aligned} \tag{18}$$

and their Taylor expansions at x_i are:

$$\begin{aligned}
\beta_0 &= f_x^2\Delta x^2 + \Big(\frac{13}{12}f_{xx}^2 - \frac{2}{3}f_x f_{xxx}\Big)\Delta x^4 \\
&\quad + \Big(-\frac{13}{6}f_{xx}f_{xxx} + \frac{1}{2}f_x f_{xxxx}\Big)\Delta x^5 + O(\Delta x^6), \\
\beta_1 &= f_x^2\Delta x^2 + \Big(\frac{13}{12}f_{xx}^2 + \frac{1}{3}f_x f_{xxx}\Big)\Delta x^4 + O(\Delta x^6), \\
\beta_2 &= f_x^2\Delta x^2 + \Big(\frac{13}{12}f_{xx}^2 - \frac{2}{3}f_x f_{xxx}\Big)\Delta x^4 \\
&\quad + \Big(\frac{13}{6}f_{xx}f_{xxx} - \frac{1}{2}f_x f_{xxxx}\Big)\Delta x^5 + O(\Delta x^6).
\end{aligned} \tag{19}$$

These indicators are designed to come closer to zero for smooth parts of the solution so that the nonlinear weights ω_m come closer to the ideal weights d_m. In the case that the stencil S^m contains a discontinuity, β_m is $O(1)$ and the corresponding weight ω_m becomes smaller, therefore the contribution of the substencil S^m is reduced.

Following the work of Henrick, Aslam and Powers [19], it can be shown that demanding (5) we obtain the sufficient conditions for the fifth order accuracy:

$$\sum_{m=0}^{2}(\omega_m^{\pm} - d_m) = O(\Delta x^6), \tag{20}$$

$$\omega_m^{\pm} - d_m = O(\Delta x^3). \tag{21}$$

Considering the overall finite difference formula

$$\hat{f}_{j+\frac{1}{2}} - \hat{f}_{j-\frac{1}{2}} = f'(x)\Delta x + O(\Delta x^6),$$

it can be shown, that (21) may be relaxed and we obtain the following sufficient and necessary conditions:

$$\sum_{m=0}^{2}(\omega_m^{\pm} - d_m) = O(\Delta x^6), \tag{22}$$

$$\sum_{m=0}^{2} A_m(\omega_m^{+} - \omega_m^{-}) = O(\Delta x^3), \tag{23}$$

$$\omega_m^{\pm} - d_m = O(\Delta x^2). \tag{24}$$

Note that due to the normalization (15), the first condition (22) (resp. (20)) is always fulfilled. The superscripts $\pm$ on ω_m specify their use in $\hat{f}_{i+\frac{1}{2}}$ or $\hat{f}_{i-\frac{1}{2}}$.

The analysis of the formal order of accuracy was performed in [6] and it was shown that if

$$\beta_m = D\big(1 + O(\Delta x^2)\big), \tag{25}$$

with D being a non-zero constant independent of m, the condition (24) is satisfied and the scheme has the expected fifth order accuracy. However, it was shown in [19] that at the critical points where the first derivative of f vanishes, the order of accuracy of the scheme from [6] decreases to the third order. Moreover, if the second derivative also vanishes, the accuracy order is further reduced to the second order. For a further explanation of this problem we refer the interested reader to [19].

2.3. The WENO-Z scheme

In this paper we consider the modified WENO scheme of Borges et al. [13] with a new global smoothness indicator, which is characterized by

$$\tau_5 = |\beta_0 - \beta_2|. \tag{26}$$

It is easy to see from the equations (19) that

$$\tau_5 = \left|-\frac{13}{3} f_{xx} f_{xxx} + f_x f_{xxxx}\right| \Delta x^5 + O(\Delta x^6). \tag{27}$$

The new WENO-Z weights are then defined by

$$\omega_m^Z = \frac{\alpha_m^Z}{\sum_{i=0}^{2} \alpha_i^Z}, \quad \text{where} \quad \alpha_m^Z = d_m \left[1 + \left(\frac{\tau_5}{\beta_m + \epsilon}\right)^2\right]. \tag{28}$$

Borges et al. [13] have shown that when using these nonlinear weights, fifth order accuracy is preserved, even at the critical points where $f'(u) = 0$.

3. The Deep Learning Approach for WENO Schemes

To better capture discontinuities and avoid oscillations, we propose to apply deep learning to develop new smoothness indicators. We construct them as products of the original smoothness indicators β_m and multipliers δ_m which are outputs of a neural network algorithm. We refer to these

new smoothness indicators as β_m^{DS}, where index DS corresponds to the new WENO-DS scheme:

$$\beta_m^{DS} = \beta_m(\delta_m + C), \tag{29}$$

where C is a constant, whose role is crucial for the proof of consistency and the formal order of accuracy and we will explain how to choose it in the Section 4. We emphasize that this formulation as a multiplication is very advantageous in a sense that the consistency and accuracy properties can be analytically shown. In the case that the solution is smooth and the original smoothness indicator β_m converges to zero, the smoothness indicator β_m^{DS} behaves in the same way. If the smoothness indicator β_m is $O(1)$, the multiplier δ_m can change it so that the final scheme performs better. We emphasize, that there was an attempt by Liu and Wen [39] to learn the smoothness indicators directly. However, in this case the consistency and accuracy analysis could not be performed.

In the original WENO method, the stencil (8) is used to approximate the solution in x_i, and the fluxes are being reconstructed in the points $x_{i-\frac{1}{2}}$ and $x_{i+\frac{1}{2}}$. To define $\hat{f}^m_{i-\frac{1}{2}}$ we use (10) and shift each index by -1. In our approach we proceed as in the classical WENO method [5] and compute the smoothness indicators as described in (17) in the Section 2.2. We use them for a flux reconstruction $\hat{f}_{i+\frac{1}{2}}$ and then by shifting each of the index by -1 we compute the smoothness indicators corresponding to the flux approximation $\hat{f}_{i-\frac{1}{2}}$. We denote them as $\beta_{m,i+\frac{1}{2}}$ and $\beta_{m,i-\frac{1}{2}}$, respectively. For a fixed m we could make the multiplier δ_m for $\beta_{m,i+\frac{1}{2}}$ and $\beta_{m,i-\frac{1}{2}}$ dependent on the location of the substencils corresponding to $\beta_{m,i+\frac{1}{2}}$ and $\beta_{m,i-\frac{1}{2}}$. This would result in two different multipliers for $\beta_{m,i+\frac{1}{2}}$ and $\beta_{m,i-\frac{1}{2}}$. However, experimentally we got the superior results by using the same multiplier $\delta_{m,i}$ for both $\beta_{m,i+\frac{1}{2}}$ and $\beta_{m,i-\frac{1}{2}}$, dependent only on the position i of the global stencil. The new smoothness indicators are then computed as

$$\begin{aligned} \beta^{DS}_{m,i+\frac{1}{2}} &= \beta_{m,i+\frac{1}{2}}(\delta_{m,i} + C), \\ \beta^{DS}_{m,i-\frac{1}{2}} &= \beta_{m,i-\frac{1}{2}}(\delta_{m,i} + C), \end{aligned} \tag{30}$$

and the values δ_0, δ_1, δ_2 are obtained by simple index shift so that it holds

$$\delta_{0,i+1} = \delta_{1,i} = \delta_{2,i-1}, \quad i = 0, \ldots, N. \tag{31}$$

Finally, we obtain the flux approximations in the same way as in classical

WENO schemes, but using the new smoothness indicators (30):

$$\hat{f}^p_{i-\frac{1}{2}} \quad \text{and} \quad \hat{f}^n_{i+\frac{1}{2}}, \tag{32}$$

which are used for approximating of a solution in a point x_i. Superscripts p and n indicate the difference between the values at the same location $x_{i+\frac{1}{2}}$ when we approximate the solution in points x_i and x_{i+1} (resp. at the same location $x_{i-\frac{1}{2}}$ when we approximate the solution in points x_{i-1} and x_i). We present the whole algorithm of the method in the Figure 1.

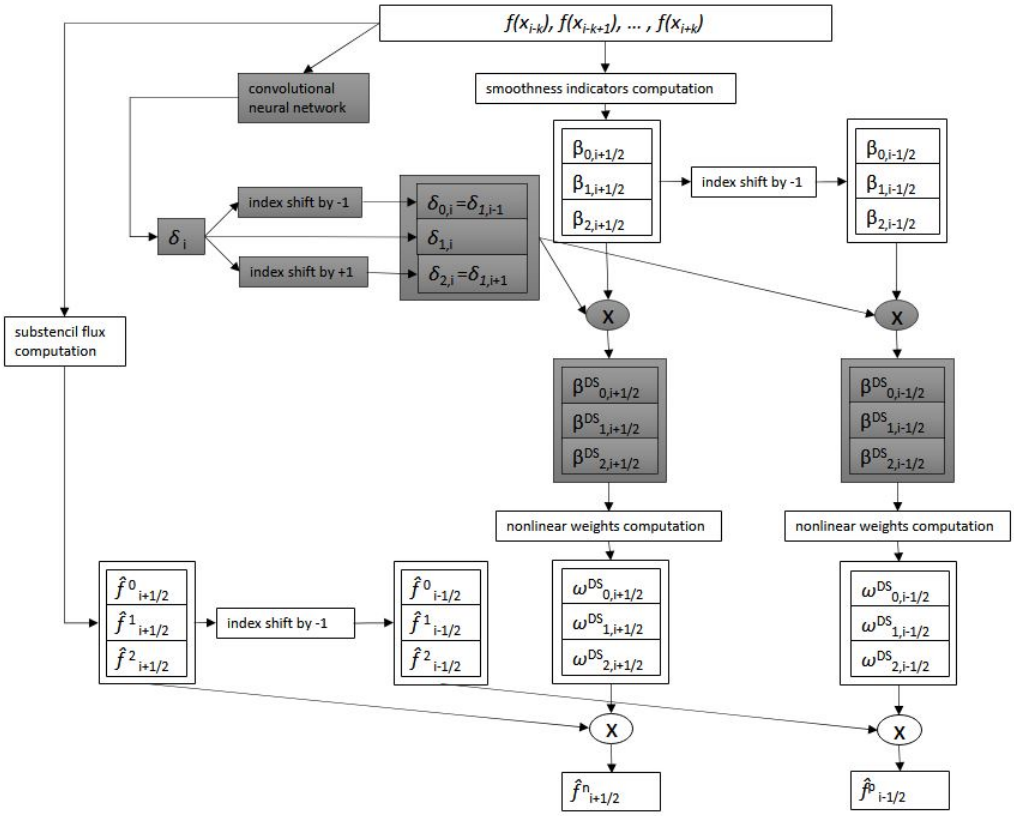


Figure 1: The structure of the WENO method combined with the neural network algorithm. The white parts of the graph correspond to the original WENO method. The grey parts are added to this method so that the whole graph corresponds to the new method WENO-DS. $2k+1$ is the size of the receptive field of the whole CNN, $\times$ denotes the element-wise multiplication.

As we mentioned before, the flux splitting technique (6) is used. Each

part of a flux, f^+ and f^- represents different type of input data to the neural network. Therefore we use two neural networks, for the positive and negative part of a flux with the input values $f^+(x_i)$ to the first neural network and $f^-(x_i)$ to the second neural network, $i = 0, \dots, N$. Each of the neural networks produces different outputs, multipliers corresponding to the positive and negative part of a flux. For simplicity we further drop the superscripts $\pm$ and when we talk about the inputs to the neural network we always mean both $f^+(x_i)$ and $f^-(x_i)$. We denote by $\bar{f}(x)$ the vector $\big(f(x_0), f(x_1), \dots, f(x_N)\big)$ and formulate the neural network as a function $F\big(\bar{f}(x)\big)$.

To ensure consistency we propose the use of a *convolutional neural network (CNN)*. Firstly, this type of neural network is computationally efficient and secondly, it makes the multipliers independent of their position in the spatial grid so that the final numerical scheme is spatially invariant. To ensure the accuracy of the method, we use the differentiable activation functions like the exponential linear unit (ELU) and the sigmoid function. If the layers of the neural network are differentiable functions, then their composition, the neural network function $F(\cdot)$, is also a differentiable function. The ELU activation function has the advantage that it does not cause the dying gradient problem, the sigmoid activation function ensures that the output of the neural network is between 0 and 1.

Let us note that we use for the implementation Python with the deep learning library PyTorch [44] (`https://pytorch.org/`), which is capable of GPU acceleration.

4. Accuracy analysis of the new WENO scheme (WENO-DS)

4.1. Accuracy analysis of WENO-JS scheme with new smoothness indicators $\beta^{DS}_{m,i\pm\frac{1}{2}}$

Let us express the multipliers $\delta_{m,i}$ for the smoothness indicators $\beta_{m,i\pm\frac{1}{2}}$, $m = 0, 1, 2$ used in the node x_i as the outputs of a neural network function. Following (31) and using the fact, that the layers of the CNN are spatially invariant differentiable functions, we can write

$$\begin{aligned}
\delta_{0,i} &= F\big(\bar{f}(\bar{x}_{i-1})\big) = \Phi(\bar{x}_i - \Delta x) = \Phi(\bar{x}_i) - O(\Delta x),\\
\delta_{1,i} &= F\big(\bar{f}(\bar{x}_i)\big) = \Phi(\bar{x}_i),\\
\delta_{2,i} &= F\big(\bar{f}(\bar{x}_{i+1})\big) = \Phi(\bar{x}_i + \Delta x) = \Phi(\bar{x}_i) + O(\Delta x),
\end{aligned} \tag{33}$$

where

$$\begin{aligned} \bar{x}_i &= (x_{i-k}, x_{i-k+1}, \dots, x_{i+k}), \\ \bar{f}(\bar{x}_i) &= \big(f(x_{i-k}), f(x_{i-k+1}), \dots, f(x_{i+k})\big), \end{aligned} \tag{34}$$

where $2k+1$ is the size of the receptive field of the whole CNN and Φ is the function composition $F \circ \bar{f}$. Then using (25) it holds

$$\beta^{DS}_{m,i\pm\frac{1}{2}} = \beta_{m,i\pm\frac{1}{2}}(\delta_{m,i} + C) = D\big(1 + O(\Delta x^2)\big)\big(\Phi(\bar{x}_i) + O(\Delta x) + C\big), \tag{35}$$

with D being some non-zero constant independent of m. We denote $P(\bar{x}_i) = \Phi(\bar{x}_i) + C$ and we set C such that $\Phi(\bar{x}_i) + C > \kappa > 0$ with κ fixed. Then we ensure that $P(\bar{x}_i) = O(1)$. Performing the multiplication in (35) we obtain

$$\begin{aligned} \beta^{DS}_{m,i\pm\frac{1}{2}} &= D\big(P(\bar{x}_i) + P(\bar{x}_i)O(\Delta x^2) + O(\Delta x) + O(\Delta x^3)\big) \\ &= DP(\bar{x}_i)\big(1 + O(\Delta x)\big) = \tilde{D}\big(1 + O(\Delta x)\big). \end{aligned} \tag{36}$$

Here we can proceed as in [19], but for the reader's convenience we repeat the steps of the proof: insert (36) into (15) and take $\epsilon = 0$

$$\alpha^{DS}_{m,i\pm\frac{1}{2}} = \frac{d_m}{\big(\tilde{D}(1 + O(\Delta x)\big)^2} = \frac{d_m}{\tilde{D}^2}\big(1 + O(\Delta x)\big). \tag{37}$$

This implies that

$$\sum_{m=0}^{2} \alpha^{DS}_{m,i\pm\frac{1}{2}} = \frac{1}{\tilde{D}^2}\big(1 + O(\Delta x)\big), \tag{38}$$

where we used the fact that $\sum_{m=0}^{2} d_m = 1$. Finally, substituting into (15) we obtain

$$\omega^{DS}_{m,i\pm\frac{1}{2}} = d_m + O(\Delta x), \tag{39}$$

where the superscript DS denotes the enhancement of the nonlinear weights (15) using our novel method. We see, that neither the condition (21), nor (24) is satisfied. However, as (39) holds, we can still guarantee that for the WENO-JS scheme with the smoothness indicators (29) we have a formal order of accuracy degraded to the third order, cf. Borges et al. [13].

4.2. Accuracy analysis of WENO-Z scheme with new smoothness indicators $\beta^{DS}_{m,i\pm\frac{1}{2}}$

Let us now analyse the formal order of accuracy (of the reconstruction) of the scheme (28) with the new smoothness indicators (29). From (19) we see that the smoothness indicators $\beta_{m,i\pm\frac{1}{2}}$ are of the form

$$\beta_{m,i\pm\frac{1}{2}} = f_x^2\Delta x^2 + O(\Delta x^4), \tag{40}$$

and the global smoothness indicator (26)

$$\tau_5 = O(\Delta x^5). \tag{41}$$

Then it holds

$$\begin{aligned}\beta^{DS}_{m,i\pm\frac{1}{2}} &= \beta_{m,i\pm\frac{1}{2}}(\delta_{m,i} + C) = \big(f_x^2\Delta x^2 + O(\Delta x^4)\big)\big(P(\bar{x}_i) + O(\Delta x)\big)\\ &= f_x^2 P(\bar{x}_i)\Delta x^2 + O(\Delta x^3).\end{aligned} \tag{42}$$

We take $\epsilon = 0$ and choose C such that $\Phi(\bar{x}_i) + C > \kappa > 0$, with κ fixed. Then we see that in the non-critical points where $f_x \neq 0$

$$\frac{\tau_5}{\beta^{DS}_{m,i\pm\frac{1}{2}}} = \hat{D}\Delta x^3 + O(\Delta x^4), \tag{43}$$

where $\hat{D} = \frac{\left|-\frac{13}{3}f_{xx}f_{xxx} + f_x f_{xxxx}\right|}{f_x^2 P(\bar{x}_i)}$. Substituting this into (28) we obtain

$$\alpha^{DS}_{m,i\pm\frac{1}{2}} = d_m\big(1 + O(\Delta x^6)\big) \quad \text{and} \quad \sum_{m=0}^{2}\alpha^{DS}_{m,i\pm\frac{1}{2}} = \big(1 + O(\Delta x^6)\big), \tag{44}$$

so it follows directly

$$\omega^{DS}_{m,i\pm\frac{1}{2}} = d_m + O(\Delta x^6) \tag{45}$$

and the condition (21) is satisfied. Here we use the superscript DS denoting the enhancement of the nonlinear weights (28) using our novel method. Since we ensure $P(\bar{x}_i) > C > \kappa > 0$, the multipliers $P(\bar{x}_i)$ do not introduce any further critical points. Therefore the analysis of the critical points with $f_x = 0$ remains the same as in [13]. Thus we can guarantee the fifth order accuracy of the scheme (28) with the smoothness indicators (29) also in the critical points.

5. Numerical Results

For the system of ODEs resulting from (2) we use a third-order total variation diminishing (TVD) Runge-Kutta method [45] given by

$$\begin{aligned} u^{(1)} &= u^n + \Delta t\, L(u^n), \\ u^{(2)} &= \frac{3}{4}u^n + \frac{1}{4}u^{(1)} + \frac{1}{4}\Delta t\, L(u^{(1)}), \\ u^{n+1} &= \frac{1}{3}u^n + \frac{2}{3}u^{(2)} + \frac{2}{3}\Delta t\, L(u^{(2)}), \end{aligned} \tag{46}$$

where $L = -\frac{1}{\Delta x}(\hat{f}_{i+\frac{1}{2}} - \hat{f}_{i-\frac{1}{2}})$ and u^n is the solution at the time step n.

For (6) we consider in our examples the Lax-Friedrichs flux splitting

$$f^{\pm}(u) = \frac{1}{2}\big(f(u) \pm \alpha u\big), \tag{47}$$

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

The Neural Network Structure

The proposed neural network algorithm can be generally applied to any type of conservation laws. For the equations where discontinuities or shocks are present, we propose to train a neural network separately for each equation class. Then we can better adjust the size of a neural network and its structure as well as the loss function, which leads to better results.

As we mentioned earlier, the inputs to the CNN are the values $f^+(x_i)$ and $f^-(x_i)$, $i = 0, \ldots, N$ and we train two neural networks for a positive and negative part of a flux. (The superscripts $\pm$ will be further dropped.)

The first layer of the neural network is not learned, but represents a preprocessing of the solution from the last time step into a set of features that we assume to be suitable inputs for the following learned layers. Since our goal is to improve the smoothness indicators, we first compute the first and second central finite differences of $\bar{f}(\bar{x}_i)$ as defined in (34), $i = 0, \ldots, N$. These parameters give us information about the smoothness of the solution and can facilitate and speed up the training of the CNN, and we can use a rather small CNN that still remains powerful. So we have the following values as input for the first learned hidden layer:

$$f_{\text{diff1},i} = \bar{f}(\bar{x}_{i+1}) - \bar{f}(\bar{x}_{i-1}), \quad f_{\text{diff2},i} = \bar{f}(\bar{x}_{i+1}) - 2\bar{f}(\bar{x}_i) + \bar{f}(\bar{x}_{i-1}). \tag{48}$$

The values $f_{\text{diff1,i}}$, $f_{\text{diff2,i}}$ computed from f^+ from (47) represent the input values for the first neural network and the values $f_{\text{diff1,i}}$, $f_{\text{diff2,i}}$ computed from f^- represent the input values for the second neural network.

Next, we use a fixed number of hidden layers, each with a specific kernel size and number of channels. We set these CNN parameters separately for each of the equation classes and experimentally find the best setting for each equation, keeping the size of proposed CNN small. We move the kernel by one space step so the stride is set to 1, and we use an ELU activation function in all hidden layers except the last one where we use sigmoid. In all our experiments, we set $C = 0.1$ in (29), which we experimentally found to be efficient. Let us note, that due to subsequent normalization of β_m^{DS} during the computation of nonlinear weights, using large value of C would decrease the effect of the trained multipliers. On the other hand, for C close to zero the experimental order of convergence could be smaller on coarse grids (but still achieved for $\Delta x \to 0$). We use the nonlinear weights as defined in (28), replacing β_m with β_m^{DS}. The value of ϵ is set to 10^{-13}.

As the first choice of the loss function we use the mean square error

$$LOSS_{\text{MSE}}(u) = \frac{1}{N}\sum_{i=0}^{N}(u_i - u_i^{\text{ref}})^2, \tag{49}$$

where u_i is a numerical approximation of $u(x_i)$ and u_i^{ref} is the corresponding reference solution. An advantage of this L_2-norm based loss function in contrast to the L_1-norm based loss function is stronger gradients with respect to u_i resulting in faster training.

5.1. The Buckley-Leverett equation

In the first example, we apply our neural network algorithm to the Buckley-Leverett equation, which was also considered, for example, in [45, 46, 6]. It is a typical example with a non-convex flux function modeling a two-phase fluid flow in a porous medium [47]. The flux in (1) is given by

$$f(u) = \frac{u^2}{u^2 + a(1-u)^2}, \quad -1 \le x \le 1, \quad 0 \le t \le 0.4, \tag{50}$$

where $a < 1$ is a constant indicating the ratio of the viscosities of the two fluids. The initial condition is set as

$$u(x,0) = \begin{cases} 1, & \text{if} \quad -0.5 \le x \le 0, \\ 0, & \text{elsewhere} \end{cases} \tag{51}$$

and we use periodic boundary condition.

In our implementation, we use the CNN with 3 hidden layers with the structure described in the Figure 2. The inputs to the learned hidden layers are the features (48). First, we create the dataset for which we compute the reference solution for the equation (1) with the flux (50) and the initial condition (51). We randomly generate the parameter a from a uniformly distributed range $[0.05, 0.95]$. We divide the computational domain $[-1, 1]$ into 1024 spatial steps and the solution is computed up to time $T = 0.4$, where the time domain is divided into 8960 time steps. We use the WENO-Z method to compute this reference solution.

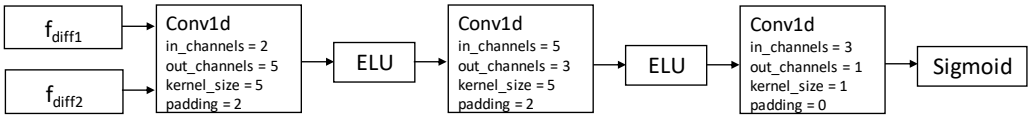


Figure 2: A structure of the convolutional neural network used for the Buckley Leverett equation (the structure is same for both inputs $f^+(x_i)$ and $f^-(x_i)$, f_{diff1} and f_{diff2} are defined in (48) and are computed from both $f^+(x_i)$ and $f^-(x_i)$).

For the training, we proceed as follows. At the beginning, we randomly choose a problem and its reference solution from our dataset. The weights of the CNN are randomly initialized and we train our model on a solution where our computational domain is divided into 128×140 steps and successively compute the entire solution until the final time T. Using the solution at the time step n, we compute the solution at the time step $n + 1$ and during this computation the CNN is used to predict the multipliers of the smoothness indicators. After each of these time steps, we compute the loss and its gradient with respect to the weights of the CNN using backpropagation algorithm. Then we use this gradient to update the weights, using the well-known Adam optimizer [48] with learning rate 0.0001. After the last time step at time T, we test a model on a validation set and repeat the above steps. Then we select the model with the best performance on the validation set as our final model. For both training and comparing the performance of the models, we use the loss function defined as

$$LOSS(u) = LOSS_{\text{MSE}}(u) + LOSS_{OF}(u), \tag{52}$$

where $LOSS_{\mathrm{MSE}}(u)$ is defined in (49) and

$$LOSS_{OF}(u) = \sum_{i=0}^{N} |\min(u_i, u_{\min}) - u_{\min}| + |\max(u_i, u_{\max}) - u_{\max}| \qquad (53)$$

represents the sum of the overflows of the solution above the maximum and below the minimum value of u, in our case $u_{\max} = 1$ and $u_{\min} = 0$. By adding this term to our loss function, we want to avoid the undesirable oscillations that occur especially in the first time steps of the solution.

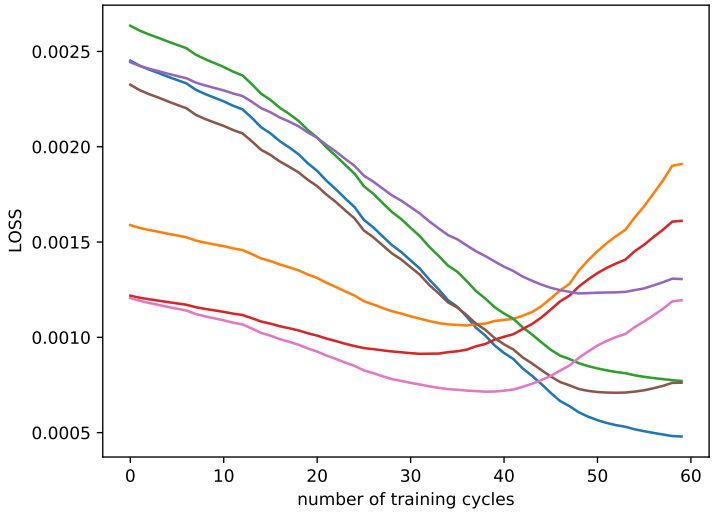


Figure 3: Loss values for different validation problems at different training cycles (x-axis).

The Figure 3 shows how the value of the loss function for the problems from the validation set (which are not present in the training set) changes with increasing number of training cycles. As training cycle we denote a sequence of training steps performed on a solution for a single randomly chosen parameter a until the final time T. The loss is then evaluated at this final time T. We apparently see two optima for different values of a. If there are more than 50 training cycles, the loss begins to increase significantly for some problems, indicating that further training is not efficient. This is caused by a fact, that the neural network structure is not complex enough to represent multipliers suitable for the whole range of problems. Instead, the optimization finds a compromise, which is suboptimal for some of the problems (leading to loss increase). Since we want only one final numerical

scheme as output, we choose the model obtained after the 46th training cycle as the final model and present the result computed with it.

We compare the L_∞- and L_2-error in the Table 1 for the solution of the conservation law (1) with (50), $a \in \{0.25, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$. Let us note that these parameters were neither in the training, nor in the validation set. We highlight the best performing WENO method in bold. In the 'ratio'

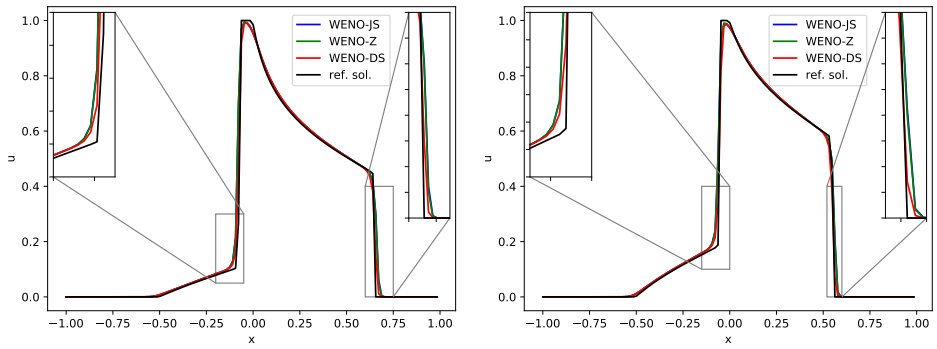
	L_∞				L_2			
a	WENO-JS	WENO-Z	WENO-DS	ratio	WENO-JS	WENO-Z	WENO-DS	ratio
0.25	0.429654	0.435090	**0.183302**	2.34	0.068405	0.067912	**0.034065**	1.99
0.4	0.408252	0.405047	**0.340068**	1.19	0.059344	0.058160	**0.056051**	1.04
0.5	0.317824	0.320094	**0.179696**	1.77	0.049913	0.049026	**0.033758**	1.45
0.6	0.459994	0.456687	**0.297523**	1.53	0.062155	0.061275	**0.048766**	1 26
0.7	0.476089	0.475015	**0.310196**	1.53	0.073021	0.072581	**0.049836**	1.46
0.8	0.207676	**0.197021**	0.250032	0.79	0.032560	**0.030994**	0.038974	0.80
0.9	0.375720	0.367802	**0.181120**	2.03	0.062257	0.061834	**0.038510**	1.61

Table 1: Comparison of L_∞ and L_2 error of WENO-JS, WENO-Z and WENO-DS methods for the solution of the Buckley-Leverett equation with the initial condition (51). As 'ratio' we denote the minimum error of the methods WENO-JS and WENO-Z divided by the error of WENO-DS (rounded to 2 decimal points).

column, we divide the minimum error of WENO-JS and WENO-Z by the error of WENO-DS to show how well our new method performs compared to the better of the standard methods mentioned. WENO-DS outperforms the standard WENO methods in most cases. Only for $a = 0.8$ is the error of WENO-DS larger than the error of the other two methods. However, this may be due to the fact that the standard WENO methods perform disproportionately well for $a = 0.8$. $a = 0.8$ perform disproportionately well compared to other values of the parameter a.

In the Figure 4, we show the solution of the Buckley-Leverett equation for the test problems with $a = 0.25$ and $a = 0.5$. It can be seen that the WENO-DS gives a better solution quality than the WENO-JS or WENO-Z.

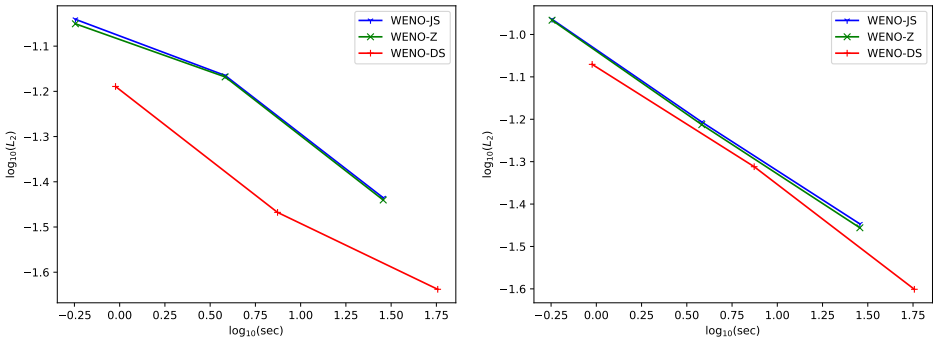
We also analyze the computational cost of our method and present the comparison in Figure 5. For this purpose, we also retrained the neural network on 64 and 256 grid points in space. In the Figure 5a, we compare the computational cost of all discussed methods for solving the Buckley-Leverett equation with $a = 0.25$. We see that WENO-DS performs significantly better than WENO-JS and WENO-Z. In the second Figure 5b we compare the methods for $a = 0.6$. From the Table 1, we see that WENO-DS provides only



(a) Solution for $a = 0.25$ (b) Solution for $a = 0.5$

Figure 4: Comparison of the WENO-JS, WENO-Z and WENO-DS methods on the solution of the Buckley-Leverett equation with the initial condition (51), $N = 128$.

a moderate improvement in error rate, but even in this case it outperforms the other two methods in efficiency. Furthermore, it should be noted that the neural network implementation is not yet optimized. For example, the speed could be further increased by GPU acceleration.



(a) $a = 0.25$ (b) $a = 0.6$

Figure 5: Comparison of computational cost against L_2-error on the solution of the Buckley-Leverett equation with the initial condition (51).

A further improvement in the speed of our method could be achieved if the multipliers were updated only once per time step. More precisely, we can compute the multipliers only in the first Runge-Kutta stage and then, assuming that the smoothness of the solution does not change significantly within one time step, the same multipliers could be used in the following two Runge-Kutta stages. This means that the costly evaluation of the CNN is

performed only once per time step instead of three times per time step, and the corresponding additional time cost is reduced to only about 1/3. We used this approach to test our method and examine how the error values change. Corresponding results can be found in Table 2. We see that very similar error values are obtained, which justifies this approach and makes it interesting for future research.

	L_∞				L_2			
a	WENO-JS	WENO-Z	WENO-DS	ratio	WENO-JS	WENO-Z	WENO-DS	ratio
0.25	0.429654	0.435090	**0.209345**	2.05	0.068405	0.067912	**0.036956**	1.84
0.4	0.408252	0.405047	**0.314087**	1.29	0.059344	0.058160	**0.054628**	1.06
0.5	0.317824	0.320094	**0.166484**	1.91	0.049913	0.049026	**0.033010**	1.49
0.6	0.459994	0.456687	**0.306510**	1.49	0.062155	0.061275	**0.047933**	1.28
0.7	0.476089	0.475015	**0.318173**	1.49	0.073021	0.072581	**0.050179**	1.45
0.8	0.207676	**0.197021**	0.226666	0.87	0.032560	**0.030994**	0.036531	0.85
0.9	0.375720	0.367802	**0.184700**	1.99	0.062257	0.061834	**0.038899**	1.59

Table 2: Comparison of L_∞ and L_2 error of WENO-JS, WENO-Z and WENO-DS methods for the solution of the Buckley-Leverett equation with the initial condition (51), when the evaluation of the CNN was done only in the first Runge-Kutta stage.

Finally, we verify the analytically proven fifth-order accuracy of the WENO-DS scheme for a transport equation with a smooth solution given as

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = 0, \quad u(x,0) = \sin(\pi x), \quad 0 \le x \le 2, \quad 0 \le t \le 0.5, \tag{54}$$

with periodic boundary conditions. Let us note, that we use the same WENO-DS method, which is an output of the training procedure for the Buckley-Leverett equation and we also do not retrain the neural network for different N. Here we demonstrate numerically, that our method can be reliably used also for different class of equation with different initial condition and remains convergent. The results can be found in Table 3. There is a great improvement when we compare our scheme with the WENO-NN scheme of Stevens and Colonius [38], where the resulting scheme exhibits only first-order accuracy.

Although our scheme remains convergent for any type of equation and for arbitrary discretization, in examples with strong discontinuities we recommend to retrain the neural network for solving a new class of PDE. Doing so, we can improve the performance of the neural network and achieve the enhancement when compared to the existing methods. We refer to [38], where authors aim to use the neural network trained only once for any class

	WENO-Z		WENO-DS	
N	L_∞	Order	L_∞	Order
20	9.369742e-03	-	9.402549e-03	-
40	2.558719e-04	5.194516	2.558830e-04	5.199496
80	9.466151e-06	4.756500	9.466165e-06	4.756560
160	3.177833e-07	4.896663	3.177834e-07	4.896665
320	9.957350e-09	4.996137	9.957351e-09	4.996138
640	3.117835e-10	4.997145	3.117834e-10	4.997146

Table 3: L_∞-norm error and convergence order of WENO-Z and WENO-DS on (54).

of PDE. However, no improvement e.g. in Burgers' equation is achieved and as demonstrated on example with 1-D Euler equations, also no improvement can be guaranteed when the discretization changes.

5.2. The inviscid Burgers' equation

In the next example we consider the inviscid Burgers' equation, where the flux function in (1) is given by

$$f(u) = \frac{u^2}{2}, \quad 0 \le x \le 2, \quad 0 \le t \le 0.3. \tag{55}$$

We consider following initial conditions

$$u(x,0) = \begin{cases} z_1, & \text{if} \quad 1 \le x \le 2, \\ 0, & \text{elsewhere}, \end{cases} \tag{56}$$

$$u(x,0) = \exp\big(-z_2(x-1)^2\big), \tag{57}$$

$$u(x,0) = z_3 \sin(\pi x), \tag{58}$$

where

$$z_1 \in \mathcal{U}[1,2], \quad z_2 \in \mathcal{U}[10,30], \quad z_3 \in \mathcal{U}[1,2]. \tag{59}$$

Using these initial conditions, we cover problems with both continuous and discontinuous initial conditions, and we simulate the shocks and discontinuities very well. We train a single CNN on all mentioned classes of initial conditions and use periodic boundary condition.

We first create the data set for training, in which we compute the reference solution of the Burgers' equation with the initial conditions (56)–(58). The

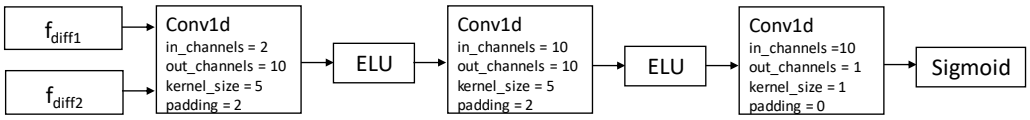


Figure 6: A structure of the convolutional neural network used for the Burgers' equation (the structure is the same for both inputs $f^+(x_i)$ and $f^-(x_i)$, f_{diff1} and f_{diff2} are defined in (48) and are computed from both $f^+(x_i)$ and $f^-(x_i)$).

computational domain is divided into 1024 space steps and 6400 time steps and the solution is computed up to time $T = 0.3$ using the WENO-Z scheme. We use the CNN with 3 hidden layers with the structure described in the Figure 6. For the training, we proceed in the same way as in the previous example. The only differences are that the computational domain is divided into 128×100 steps and the learning rate used by the Adam optimizer is 0.001. In this example, we use the mean square error loss function (49) for training and validation. As the training on Burgers' equation exhibits much higher variance than in the Buckley-Leverett case, we performed 3 trainings each with 90 training cycles and finally selected the model showing the best performance on the validation set.

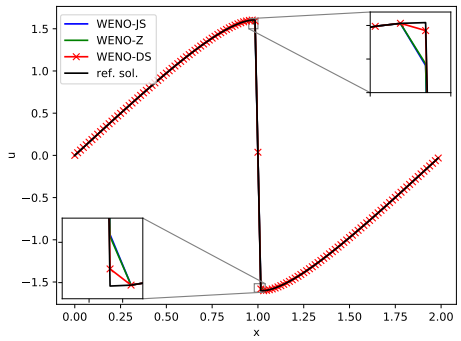
		L_∞				L_2			
initial condition	z_j	WENO-JS	WENO-Z	WENO-DS	ratio	WENO-JS	WENO-Z	WENO-DS	ratio
(56)	1.19	0.632213	0.632788	**0.303060**	2.09	0.082373	0.082141	**0.046932**	1.75
	1.53	0.594943	0.58678	**0.485714**	1.21	0.080341	0.07877	**0.067175**	1.17
	1.84	0.704680	0.694358	**0.542599**	1.28	0.094967	0.093019	**0.076102**	1.22
(57)	14.94	0.113498	0.104374	**0.100061**	1.04	0.016926	**0.015137**	0.015164	1.00
	21.65	0.236125	0.229290	**0.196110**	1.17	0.032979	0.031680	**0.029141**	1.09
	29.08	0.312595	**0.310937**	0.388739	0.80	0.040632	**0.040199**	0.049385	0.81
(58)	1.46	0.059072	0.056751	**0.051553**	1.10	0.010443	0.010032	**0.007307**	1.37
	1.6	0.063780	0.061391	**0.037165**	1.65	0.011275	0.010853	**0.005552**	1.95
	1.9	0.072586	0.069995	**0.023841**	2.94	0.012831	0.012373	**0.003396**	3.64

Table 4: Comparison of L_∞ and L_2 error of WENO-JS, WENO-Z and WENO-DS methods for the solution of the Burgers' equation with the initial condition parameters inside of training set intervals (59). As 'ratio' we denote the minimum error of the methods WENO-JS and WENO-Z divided by the error of WENO-DS (rounded to 2 decimal points).

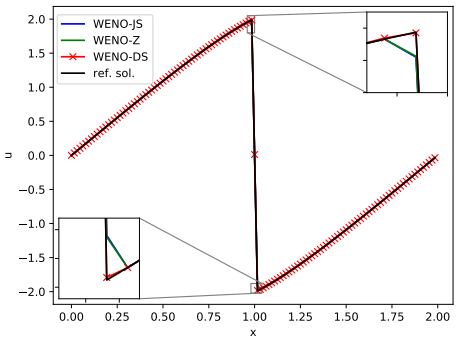
We compare the errors on the problems from the test set in Table 4 and 5. These were not in the training or validation set and the parameters were randomly generated. We observe rather small or no improvement for problems with the initial condition (57), but the improvement is significant for the solution with the discontinuous initial condition (56) as well as with

		L_∞				L_2			
initial condition	z_j	WENO-JS	WENO-Z	WENO-DS	ratio	WENO-JS	WENO-Z	WENO-DS	ratio
(56)	0.71	0.204162	0.199246	**0.150400**	1.32	0.031740	0.030740	**0.028178**	1.09
	2.41	1.541714	1.554201	**1.004904**	1.53	0.199285	0.200194	**0.129223**	1.54
	2.57	1.063823	1.055948	**0.755908**	1.40	0.140068	0.138411	**0.102963**	1.34
	3.13	0.622858	0.600619	**0.287540**	2.09	0.087067	0.084495	**0.054477**	1.55
(57)	33.9	0.351086	0.345278	**0.266422**	1.30	0.045665	0.044691	**0.036426**	1.23
	34.67	0.285791	0.283237	**0.194424**	1.46	0.037350	0.036815	**0.027150**	1.36
(58)	0.94	0.009524	**0.007189**	0.007898	0.91	0.001737	0.001509	**0.001491**	1.01
	2.12	0.077503	0.074744	**0.012296**	6.08	0.013701	0.013213	**0.001712**	7.72
	2.44	0.083010	0.080022	**0.003978**	20.12	0.014675	0.014147	**0.000537**	26.35

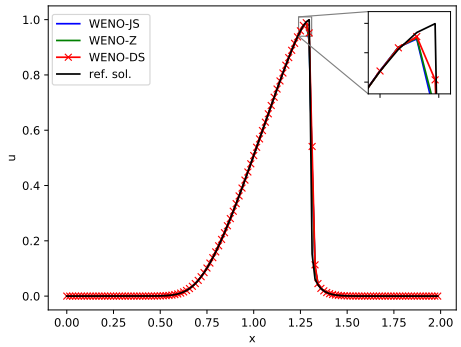
Table 5: Comparison of L_∞ and L_2 error of WENO-JS, WENO-Z and WENO-DS methods for the solution of the Burgers' equation with the initial condition parameters outside of training set intervals (59). As 'ratio' we denote the minimum error of the methods WENO-JS and WENO-Z divided by the error of WENO-DS (rounded to 2 decimal points).



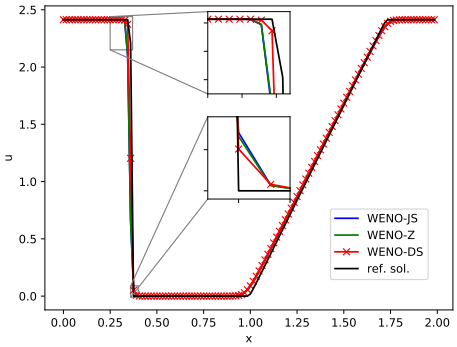
(a) Initial condition (58) with $z_3 = 1.6$.



(b) Initial condition (58) with $z_3 = 2.12$.



(c) Initial condition (57) with $z_2 = 29.08$.



(d) Initial condition (56) with $z_1 = 2.41$.

Figure 7: Comparison of the WENO-JS, WENO-Z and WENO-DS methods for the solution of the Burgers' equation with various initial conditions, $N = 128$.

the initial condition (58).

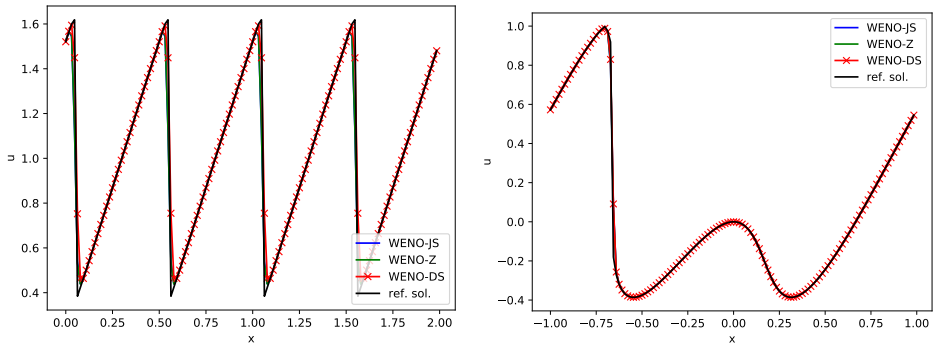
We conclude that the WENO-DS significantly outperforms the classical WENO methods. It should be noted that although our training set was created with the parameters sampled from uniform distribution as specified in (59), the method can also generalise for parameter values outside of these intervals, as can be seen in Table 5. Especially, we highlight the last two problems from Table 5, where we see a great improvement.

In the Figure 7 we show the solution of the Burgers' equation with the initial condition (56) for $z_1 = 2.41$, (57) for $z_2 = 29.08$, (58) for $z_3 = 1.6$ and (58) for $z_3 = 2.12$. We observe that WENO-DS captures shocks and discontinuities very well and gives us a better solution compared to WENO-JS and WENO-Z.

Next, we test our method on two examples, where the Burgers' equation with the flux function (55) and following initial conditions will be solved:

$$u(x,0) = 1 + \sin(4\pi x), \quad 0 \leq x \leq 2, \tag{60}$$

$$u(x,0) = -x \sin\left(\frac{3}{2}\pi x\right), \quad -1 \leq x \leq 1. \tag{61}$$



(a) Initial condition (60). (b) Initial condition (61).

Figure 8: Comparison of the WENO-JS, WENO-Z and WENO-DS methods for the solution of the Burgers' equation with various initial conditions, $N = 128$.

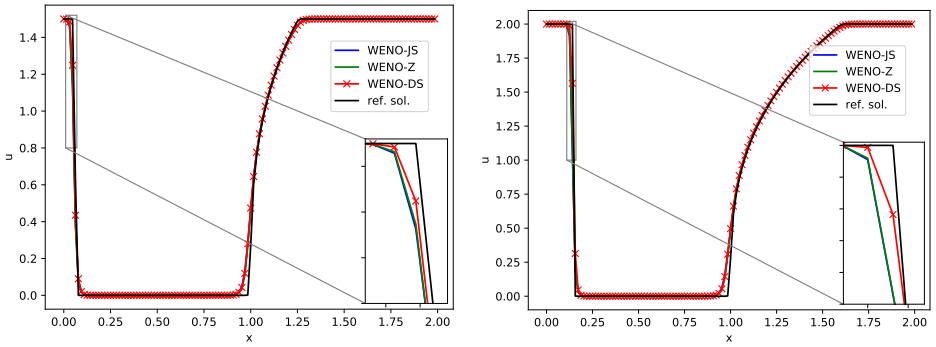
We compute the solution up to time $T = 0.3$. Let us note, that the method was not retrained on these initial conditions and still performs well. Figure 8 illustrates the solution.

Finally, we test our method on the equation (1) with another flux func-

tion, according to [37]

$$f(u) = \frac{u^4}{16}, \quad 0 \le x \le 2, \quad 0 \le t \le 0.3. \tag{62}$$

For this purpose, we use the initial condition (56) with $z_1 = 1.5$ and $z_1 = 2$. We illustrate the solution in Figure 9 computed up to the final time $T = 0.3$ and see that WENO-DS performs very well in both cases.



(a) Initial condition (56) with $z_1 = 1.5$.

(b) Initial condition (56) with $z_1 = 2$.

Figure 9: Comparison of the WENO-JS, WENO-Z and WENO-DS methods for the solution of the equation (1) with the flux function (62), $N = 128$.

5.3. The two-dimensional Burgers' equation

To demonstrate the performance of WENO-DS in two dimensional space we apply the method trained on one-dimensional data for the 1D Burgers' equation (55) to the two-dimensional Burgers' equation of the form

$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} + \frac{\partial f(u)}{\partial y} = 0, \qquad f(u) = \frac{u^2}{2} \tag{63}$$

on the spatial domain $[-1, 1] \times [-1, 1]$ divided into 128×128 uniform cells. As considered by Cao, Xu and Zheng [49] we use the initial condition

$$u(x, y, 0) = (x^2 - 1)^2 (y^2 - 1)^2, \qquad -1 \le x \le 1, \quad -1 \le y \le 1. \tag{64}$$

We present the solution at time $T = 0.8$ in the Figure 10. We compare the error values in Table 6, where the reference solution was computed on the spatial discretization with 256×256 cells. Let us note, that no additional retraining was needed, as we apply the method using dimension-by-dimension principle.

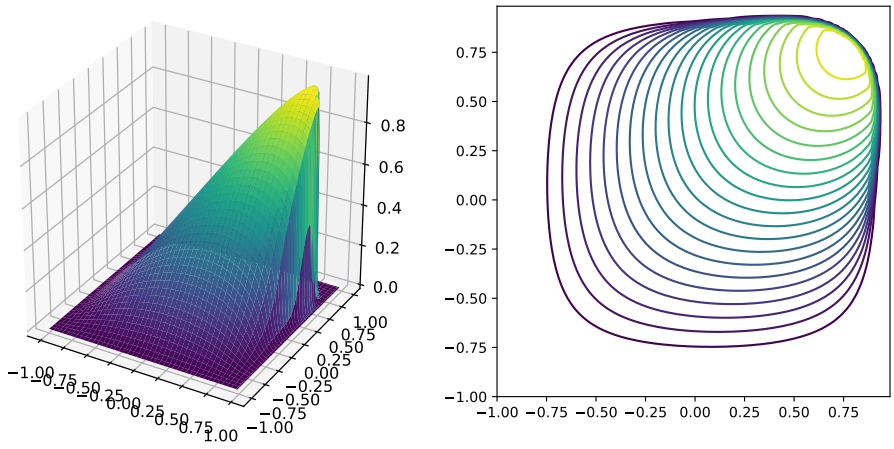


Figure 10: Numerical solution of the two-dimensional Burgers' equation using WENO-DS at $T = 0.8$. 128×128 cells.

	L_∞				L_2			
$N \times N$	WENO-JS	WENO-Z	WENO-DS	ratio	WENO-JS	WENO-Z	WENO-DS	ratio
64×64	0.411625	0.409869	**0.323205**	1.27	0.023487	0.022837	**0.020863**	1.09

Table 6: Comparison of L_∞ and L_2 error of WENO-JS, WENO-Z and WENO-DS methods for the solution of the two-dimensional Burgers equation with the initial condition (64). As 'ratio' we denote the minimum error of the methods WENO-JS and WENO-Z divided by the error of WENO-DS (rounded to 2 decimal points).

5.4. The one-dimensional Euler equations

We now investigate how WENO-DS behaves when applied to the one-dimensional Euler system, which is considered a classical benchmark problem for methods for conservation laws. It has the form

$$\begin{aligned} \frac{\partial \rho}{\partial t} + \frac{\partial (\rho u)}{\partial x} &= 0, \\ \frac{\partial \rho u}{\partial t} + \frac{\partial (\rho u^2 + p)}{\partial x} &= 0, \\ \frac{\partial E}{\partial t} + \frac{\partial (uE + up)}{\partial x} &= 0, \end{aligned} \tag{65}$$

where ρ is the density, u is the velocity, p is the pressure and E is a total energy given by

$$E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho u^2. \tag{66}$$

We take $\gamma = 1.4$, which is the ratio of the specific heats. To compute the fluxes, we use the characteristic decomposition of the system according to the steps in [5]. We use the Roe scheme to obtain the eigenvectors and eigenvalues [50] and the Lax-Friedrichs flux splitting to obtain the corresponding component of the flux. We take the solution based on [51] as the reference solution.

The most common benchmark problems are the Sod problem [52], where the initial condition is specified as

$$(\rho, u, p) = \begin{cases} (1, 0.75, 1) & 0 \leq x < 0.5, \\ (0.125, 0, 0.1) & 0.5 \leq x \leq 1 \end{cases} \tag{67}$$

and the Lax problem [53] with an initial condition

$$(\rho, u, p) = \begin{cases} (0.445, 0.698, 3.528) & 0 \leq x < 0.5, \\ (0.5, 0, 0.571) & 0.5 \leq x \leq 1. \end{cases} \tag{68}$$

We use an adaptive step size

$$\Delta t = \frac{0.9\Delta x}{\max(c_i + |u_i|)}, \quad c^2 = \frac{\gamma p}{\rho}, \tag{69}$$

where u_i is the local velocity and c_i the local speed of sound.

The solution consists of the left rarefaction wave, the right travelling contact wave and the right shock wave. We want to imitate this behavior of the solution, so we construct our data set as described in Appendix A.

We use the CNN with 3 hidden layers with the structure described in Figure 11. After projecting the flux and the solution on the characteristic fields using the left eigenvectors, we use Lax-Friedrichs flux splitting for each component of characteristic variables. From these values we compute the features (48), which are the inputs to the learned hidden layers.

In this example we repeat the training procedure of the previous examples with some small modifications described below. To begin, we randomly generate the initial state from the dataset described earlier. We divide the spatial domain into 100 steps and compute the solution for the given initial state up to the time $T = 0.05$. After each time step we compute loss using the reference solution from [51]. We use the gradient to update the weights, using Adam optimizer with learning rate 0.001. At the last time step we test the model on the validation problems, which are the problems with randomly

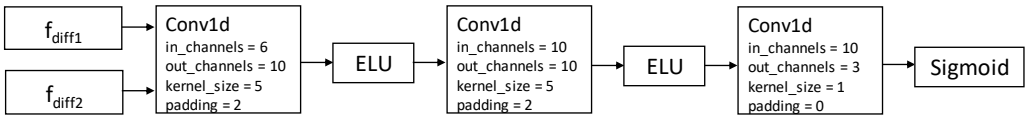


Figure 11: A structure of the convolutional neural network used for the Euler system (the structure is same for both inputs $f^+(x_i)$ and $f^-(x_i)$, f_{diff1} and f_{diff2} are defined in (48) and are computed from both $f^+(x_i)$ and $f^-(x_i)$).

generated initial conditions according to algorithm described in Appendix A and repeat the procedure with the new initial parameters (ρ, u, p). We use the loss function

$$LOSS(\rho, u, p) = LOSS_{\text{MSE}}(\rho) + LOSS_{\text{MSE}}(u) + LOSS_{\text{MSE}}(p) \tag{70}$$

for training and validation.

Based on the validation problems, we choose the final model and present the solution of the Sod problem (67) for ρ, u and p using 100, 200 and 300 space points. We compute the solution up to time $T = 0.2$. We illustrate the solution in Figure 12 and compare the corresponding error values in Table 7.

Next, we also present the solution of the Lax problem (68) with the final time $T = 0.13$ in the Figure 13.

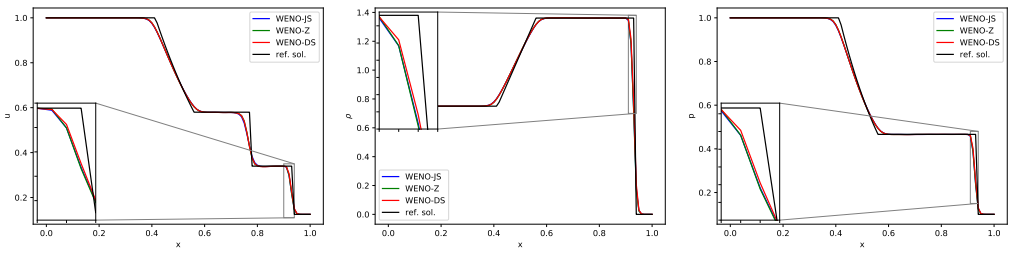


Figure 12: Solution of Sod problem (67), using the WENO-JS, WENO-Z and WENO-DS methods, $T = 0.2$, $N = 100$.

Finally, we also applied the trained method to the shock entropy wave interaction problem [46] with an initial condition

$$(\rho, u, p) = \begin{cases} (3.857143, 2.629369, 10.33333) & -5 \leq x < -4, \\ (1 + 0.2\sin(5x), 0, 1) & -4 \leq x \leq 5. \end{cases} \tag{71}$$

	L_∞				L_2			
$N = 100$	WENO-JS	WENO-Z	WENO-DS	ratio	WENO-JS	WENO-Z	WENO-DS	ratio
ρ	**0.150172**	0.152604	0.151802	0.99	0.024930	0.024381	**0.023882**	1.02
p	0.225413	0.228468	**0.211217**	1.07	0.026963	0.026894	**0.025194**	1.07
u	0.628438	0.643351	**0.577795**	1.09	0.069043	0.069791	**0.064271**	1.07
$N = 200$	WENO-JS	WENO-Z	WENO-DS	ratio	WENO-JS	WENO-Z	WENO-DS	ratio
ρ	**0.136956**	0.138225	0.136976	1.00	0.016462	0.015903	**0.015272**	1.04
p	0.215353	0.217666	**0.186693**	1.15	0.017538	0.017530	**0.015451**	1.13
u	0.591525	0.603275	**0.495718**	1.19	0.046245	0.046542	**0.042207**	1.10
$N = 300$	WENO-JS	WENO-Z	WENO-DS	ratio	WENO-JS	WENO-Z	WENO-DS	ratio
ρ	0.126180	0.125691	**0.124829**	1.01	0.012834	0.012254	**0.011701**	1.05
p	0.204582	0.208307	**0.165333**	1.24	0.013423	0.013479	**0.011468**	1.17
u	0.553209	0.570251	**0.430247**	1.29	0.035888	0.036231	**0.035287**	1.02

Table 7: Comparison of L_∞ and L_2 error of WENO-JS, WENO-Z and WENO-DS methods for the solution of the Euler equations of gas dynamics for the Sod problem (67) with $N = 100$, $N = 200$ and $N = 300$. As 'ratio' we denote the minimum error of the methods WENO-JS and WENO-Z divided by the error of WENO-DS (rounded to 2 decimal points).

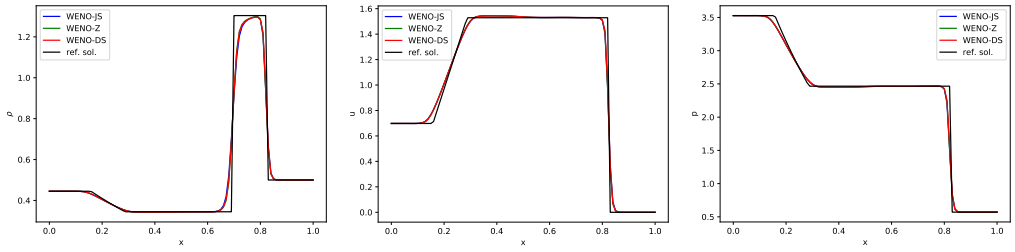


Figure 13: Solution of Lax problem (68), using the WENO-JS, WENO-Z and WENO-DS methods, $T = 0.13$, $N = 100$.

In the Figure 14 we show the solution using WENO-Z and WENO-DS methods, when the computational domain is divided into 512 uniform cells up to final time $T = 1.8$. As a reference solution we used the solution computed using WENO-Z method with 2048 space points. This is an example, which has a moving Mach = 3 shock interacting with sine waves in density, so the numerical method needs to deal with the physical oscillations contained in a flow. Finally, let us note, that although we have not trained the presented model on the parameters that would lead to such a solution, the method is robust enough and can detect the shocks present in the solution.

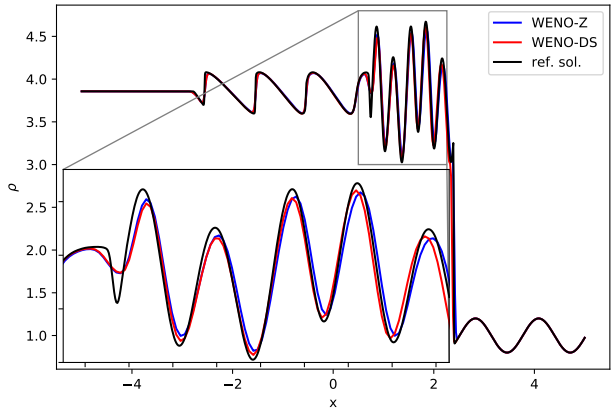


Figure 14: Solution of shock entropy wave interaction problem (71), $N = 512$.

6. Conclusion

In this work, we have improved the fifth-order WENO shock-capturing scheme by using deep learning techniques. To do this, we trained a relatively small neural network to obtain modified smoothness indicators of the WENO scheme. This was done in a way that avoided post-processing of the coefficients to ensure consistency. We applied our enhancement to the WENO-Z scheme, where the (formal) fifth-order accuracy on the smooth solutions can be proven analytically.

Our new method, the WENO-DS scheme, is quite easy to use and significantly improves the numerical results, especially in the presence of discontinuities, even for cases that have not been trained before. We have demonstrated our results with the inviscid Burgers' equation, the Buckley-Leverett equation, and the 1-D Euler equations of gas dynamics. We showed, that the method can efficiently solve the problems in more dimensional space without additional retraining.

Finally, let us note, that this paper can be seen as a proof of concept, that neural networks can be efficiently combined with an existing numerical scheme, preserving its formal accuracy order. As part of our future work we will extend our numerical scheme to more applications in 2D and 3D and also study more efficient variants of our approach using GPU computing.

Appendix A. Parameters used for generating the data set for 1-D Euler equations of gas dynamics

The problem samples representing different versions of the Euler equations of gas dynamics (67) were defined using parameters generated by the following algorithm.

Choose randomly $s \in \{0, 2\}$
if $s = 0$ **then**
  $p_l = a + b, \quad a \in \mathcal{U}[0.5, 10], \quad b \in \mathcal{U}[-0.05, 0.05],$
  $p_r = 1/c, \quad c \in \mathcal{U}[5, 10],$
  $\rho_l = p_l,$
  $\rho_r = p_r + d, \quad d \in \mathcal{U}[-0.05, 0.05],$
  $u_l = e, \quad e \in \mathcal{U}[0, 1],$
  $u_r = 0,$
else if $s = 1$ **then**
  $p_l = 1,$
  $p_r = 0.1,$
  $\rho_l = k, \quad k \in \mathcal{U}[1, 3],$
  $\rho_r = \frac{1}{10}\rho_l + l, \quad l \in \mathcal{U}[-0.05, 0.05],$
  $u_l = m, \quad m \in \mathcal{U}[0, 1],$
  $u_r = 0,$
else
  $p_l = n, \quad n \in \mathcal{U}[3, 4],$
  $p_r = \frac{1}{7}p_l + q, \quad q \in \mathcal{U}[-0.05, 0.05]$
  $\rho_l = r, \quad r \in \mathcal{U}[0.3, 0.6],$
  $\rho_r = \rho_l + s, \quad s \in \mathcal{U}[-0.05, 0.05],$
  $u_l = t, \quad t \in \mathcal{U}[0, 1],$
  $u_r = 0,$
end if

where

$$(\rho, u, p) = \begin{cases} (\rho_l, u_l, p_l) & 0 \le x < 0.5, \\ (\rho_r, u_r, p_r) & 0.5 \le x \le 1. \end{cases} \tag{A.1}$$

References

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

[2] S. Godunov, Different Methods For Shock Waves, Ph.D. thesis, Moscow State University, 1954.

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

[4] 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.

[5] 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. URL: https://doi.org/10.1007/BFb0096355. doi:10.1007/BFb0096355.

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

[7] 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 (2004) 115–135.

[8] 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 (2005) 642–663.

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

[10] 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 (2004) 435–450.

[11] Z. Zhao, Y.-T. Zhang, Y. Chen, J. Qiu, A Hermite WENO method with modified ghost fluid method for compressible two-medium flow problems, Comm. Comput. Phys. 30 (2021) 851–873.

URL: http://global-sci.org/intro/article_detail/cicp/19319.html. doi:https://doi.org/10.4208/cicp.OA-2020-0184.

[12] Z. Zhao, Y. Chen, J. Qiu, A hybrid WENO method with modified ghost fluid method for compressible two-medium flow problems, Numer. Math.: Theor. Meth. Appl. (2021) to appear. URL: http://arxiv.org/abs/2009.00461.

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

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

[15] C. H. Kim, Y. Ha, J. Yoon, Modified non-linear weights for fifth-order weighted essentially non-oscillatory schemes, J. Sci. Comput. 67 (2016) 299–323.

[16] S. Rathan, G. N. Raju, A modified fifth-order WENO scheme for hyperbolic conservation laws, Comput. Math. Appl. 75 (2018) 1531–1549.

[17] 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 (2013) 68–86.

[18] L. Li, H. B. Wang, G. Y. Zhao, et al., Efficient WENOCU4 scheme with three different adaptive switches, J. Zhejiang Univ. Sci. A 21 (2020) 695–720. URL: https://doi.org/10.1631/jzus.A2000006.

[19] 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 (2005) 542–567.

[20] Y. Liu, Globally optimal finite-difference schemes based on least squares, Geophysics 78 (2013) T113–T132.

[21] C. K. Tam, J. C. Webb, Dispersion-relation-preserving finite difference schemes for computational acoustics, J. Comput. Phys. 107 (1993) 262–281.

[22] Z. Wang, R. Chen, Optimized weighted essentially nonoscillatory schemes for linear waves with discontinuity, J. Comput. Phys. 174 (2001) 381–404.

[23] J. Fernández-Fidalgo, L. Ramírez, P. Tsoutsanis, I. Colominas, X. Nogueira, A reduced-dissipation WENO scheme with automatic dissipation adjustment, J. Comput. Phys. 425 (2021) 109749.

[24] L. Fu, X. Y. Hu, N. A. Adams, A family of high-order targeted ENO schemes for compressible-fluid simulations, J. Comput. Phys. 305 (2016) 333–359.

[25] L. Fu, X. Y. Hu, N. A. Adams, A new class of adaptive high-order targeted ENO schemes for hyperbolic conservation laws, J. Comput. Phys. 374 (2018) 724–751.

[26] L. Fu, A hybrid method with TENO based discontinuity indicator for hyperbolic conservation laws, Commun. Comput. Phys. 26 (2019) 973–1007.

[27] C.-W. Shu, High order weighted essentially nonoscillatory schemes for convection dominated problems, SIAM Review 51 (2009) 82–126.

[28] I. E. Lagaris, A. Likas, D. I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE Trans. Neural Netw. 9 (1998) 987–1000.

[29] J. Sirignano, K. Spiliopoulos, DGM: A deep learning algorithm for solving partial differential equations, J. Comput. Phys. 375 (2018) 1339–1364.

[30] J. Berg, K. Nyström, A unified deep artificial neural network approach to partial differential equations in complex geometries, Neurocomputing 317 (2018) 28–41.

[31] A. D. Beck, J. Zeifang, A. Schwarz, D. Flad, A neural network based shock detection and localization approach for discontinuous Galerkin methods, J. Comput. Phys. 423 (2020). doi:10.1016/j.jcp.2020.109824.

[32] J.-T. Hsieh, S. Zhao, S. Eismann, L. Mirabella, S. Ermon, Learning neural PDE solvers with convergence guarantees, 2019. URL: https://arxiv.org/abs/1906.01200. arXiv:1906.01200.

[33] Y. Bar-Sinai, S. Hoyer, J. Hickey, M. P. Brenner, Learning data-driven discretizations for partial differential equations, Proc. Nat. Acad. Sci. 116 (2019) 15344–15349.

[34] N. Discacciati, J. S. Hesthaven, D. Ray, Controlling oscillations in high-order discontinuous Galerkin schemes using artificial viscosity tuned by neural networks, J. Comput. Phys. 409 (2020) 109304.

[35] D. Ray, J. S. Hesthaven, Detecting troubled-cells on two-dimensional unstructured grids using a neural network, J. Comput. Phys. 397 (2019) 108845.

[36] Y. Feng, T. Liu, K. Wang, A characteristic-featured shock wave indicator for conservation laws based on training an artificial neuron, J. Sci. Comput. 83 (2020) 1–34.

[37] Y. Wang, Z. Shen, Z. Long, B. Dong, Learning to discretize: Solving 1D scalar conservation laws via deep reinforcement learning, Commun. Comput. Phys. 28 (2020) 2158–2179. URL: http://global-sci.org/intro/article_detail/cicp/18408.html. doi:https://doi.org/10.4208/cicp.OA-2020-0194.

[38] B. Stevens, T. Colonius, Enhancement of shock-capturing methods via machine learning, Theor. Comput. Fluid Dyn. 34 (2020) 483–496. URL: https://doi.org/10.1007/s00162-020-00531-1.

[39] Q. Liu, X. Wen, The WENO reconstruction based on the artificial neural network, Adv. Appl. Math. 9 (2020) 574–583. URL: https://doi.org/10.12677/aam.2020.94069.

[40] W.-S. Don, R. Borges, Accuracy of the weighted essentially non-oscillatory conservative finite difference schemes, J. Comput. Phys. 250 (2013) 347–372.

[41] F. Aràndiga, A. Baeza, A. Belda, P. Mulet, Analysis of WENO schemes for full and global accuracy, SIAM J. Numer. Anal. 49 (2011) 893–915.

[42] T. Kossaczká, The Weighted Essentially Non-Oscillatory Method for Problems in Finance, Master's thesis, Bergische Universität Wuppertal, Germany, 2019.

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

[44] A. Paszke, et al., PyTorch: An imperative style, high-performance deep learning library, in: H. Wallach, et al. (Eds.), Advances in Neural Information Processing Systems 32, Curran Associates, Inc., 2019, pp. 8024–8035.

[45] C.-W. Shu, S. Osher, Efficient implementation of essentially non-oscillatory shock-capturing schemes, J. Comput. Phys. 77 (1988) 439–471.

[46] C.-W. Shu, S. Osher, Efficient implementation of essentially non-oscillatory shock-capturing schemes, II, in: M. Hussaini, B. van Leer, J. Van Rosendale (Eds.), Upwind and High-Resolution Schemes, Springer, 1989, pp. 328–374.

[47] R. J. LeVeque, Finite volume methods for hyperbolic problems, volume 31, Cambridge University Press, 2002.

[48] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014). Published as a conference paper at ICLR 2015.

[49] W. Cao, Q. Xu, Z. Zheng, Solution of two-dimensional time-fractional Burgers equation with high and low Reynolds numbers, Adv. Difference Eqs. 2017 (2017) 1–14.

[50] P. L. Roe, Approximate Riemann solvers, parameter vectors, and difference schemes, J. Comput. Phys. 43 (1981) 357–372.

[51] P. Wesseling, Principles of Computational Fluid Dynamics, volume 29, Springer Science & Business Media, 2009.

[52] G. A. Sod, A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws, J. Comput. Phys. 27 (1978) 1–31.

[53] P. D. Lax, Weak solutions of nonlinear hyperbolic equations and their numerical computation, Comm. Pure Appl. Math. 7 (1954) 159–193.