Bergische Universität Wuppertal

Fakultät für Mathematik und Naturwissenschaften

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

Preprint BUW-IMACM 21/16

Tatiana Kossacká, Matthias Ehrhardt and Michael Günther

# A Neural Network Enhanced WENO Method for Nonlinear Degenerate Parabolic Equations

June 7, 2021

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

# A NEURAL NETWORK ENHANCED WENO METHOD FOR NONLINEAR DEGENERATE PARABOLIC EQUATIONS

TATIANA KOSSACZKÁ *, MATTHIAS EHRHARDT *, AND MICHAEL GÜNTHER *

**Abstract.** In this paper, we design a new modification of weighted essentially non-oscillatory (WENO) method for solving nonlinear degenerate parabolic equations using deep learning techniques. To this end, we modify the smoothness indicators of an existing WENO algorithm that are responsible for measuring the discontinuity of a numerical solution. We do this in such a way that the consistency and convergence of our new WENO-DS (deep smoothness) method is preserved and can be theoretically proven. We use a convolutional neural network (CNN) and present a novel and effective training procedure. Furthermore, we show that the WENO-DS method can be easily applied in more dimensions without the need to retrain the CNN. We present our results on benchmark examples of nonlinear degenerate parabolic equations, such as the porous medium equation with the Barenblatt solution, the Buckley-Leverett equation and their extensions in two-dimensional space. Here we show that our novel method outperforms the standard WENO method, reliably handles the sharp interfaces and provides good resolution of discontinuities.

**Key words.** Weighted essentially non-oscillatory (WENO) method, Smoothness indicators, Deep Learning, Nonlinear degenerate parabolic equation

**AMS subject classifications.** 65M06, 65M12, 68T05, 35K65

## 1. Introduction.

In this work, we develop a new modification of the weighted essentially non-oscillatory (WENO) scheme for solving nonlinear degenerate parabolic equations of the form

$$(1.1)\qquad \begin{aligned} u_t &= \sum_{i=1}^{d} \frac{\partial b_i(u)}{\partial x_i^2}, \qquad (\mathbf{x},t) \in \Omega \times (0,\infty),\\ u(\mathbf{x},0) &= u_0(\mathbf{x}), \end{aligned}$$

where $\mathbf{x} = (x_1, \ldots, x_d)$ with $d$ being the space dimension. The simplest form of (1.1) with $d = 1$ can be represented by

$$(1.2)\qquad \begin{aligned} u_t &= b(u)_{xx},\\ u(x,0) &= u_0(x), \end{aligned}$$

where $b'(u) \geq 0$ and it is possible that $b(u)$ vanishes for some values of $u$. In this case, the equation (1.2) degenerates on the $u$-level and is not strictly parabolic. We note that such equations are often found in applications. For $b(u) = u^m$, the equation (1.2) is called the *porous medium equation* (PME) [6, 36]:

$$(1.3)\qquad u_t = (u^m)_{xx}, \qquad m > 1,$$

which models the flow of an isentropic gas through a porous medium. At specific points, where $u = 0$, the equation (1.3) degenerates, leading to finite speed of propagation and sharp fronts. In general, a classical solution, i.e. twice continuously differentiable with respect to $x$, might not exist even in the case of a smooth initial condition. Therefore, the weak solution must be considered and is studied e.g. in [3, 27, 35].

*Bergische Universität Wuppertal, Institute of Mathematical Modelling, Analysis and Computational Mathematics (IMACM), Gaußstraße 20, 42119 Wuppertal, Germany ({kossaczka, ehrhardt, guenther}@uni-wuppertal.de).

There are several schemes that solve (1.1) numerically, such as kinetic schemes [5], relaxation schemes [13], local discontinuous Galerkin methods [38] or finite volume schemes [4, 11]. When solving (1.1), we can observe a very similar behaviour to hyperbolic conservation laws. Therefore, the well-known WENO method, which is widely used for solving hyperbolic conservation laws, has also been generalized for solving (1.1).

First, so-called *Essentially non-oscillatory* (ENO) schemes were developed to better capture the collisions involved in solving hyperbolic conservation laws [31, 32]. Later, these schemes were extended by Liu et al. [24] by introducing *Weighted essentially non-oscillatory* (WENO) schemes, which were further investigated in [18]. In this work, a new measure of smoothness was introduced based on the $L^2$-norm of derivatives of the interpolation polynomials over each substencil. Thereafter, it was found that the order of convergence of the WENO scheme introduced in [18] is smaller than the fifth order when the first derivative of the solution vanishes. Therefore, new modifications of the WENO scheme were introduced, for example, [17, 12].

Later, authors Liu et al. developed the WENO scheme for nonlinear degenerate parabolic equations [25]. Two formulations are described in [25]. In the first, the second derivative is directly approximated by a conservative flux difference. In this case, the negative ideal weights appear, so a special treatment of them is required [30]. The desired sixth order of convergence is obtained and numerically demonstrated. The second approach is based on the introduction of an auxiliary variable for the first derivative, then the WENO scheme is applied to two first derivatives instead of the second derivative. However, this case is not discussed further because the error magnitude is larger than in the case of direct application of the WENO method to the second derivative.

Subsequently, new modifications of the sixth-order WENO scheme for nonlinear degenerate parabolic equations were introduced. Christlieb et al. [14] supplied a high order WENO method with a nonlinear filter to avoid spurious oscillations. Hajipour and Malek [15] introduced a new type of nonlinear weights and used a nonstandard Runge-Kutta scheme instead of the *Total Variation Diminishing (TVD)* Runge-Kutta [31] previously used in combination with WENO methods. Abedian et al. [2, 1] aimed to avoid the negative ideal weights and present a new modification of the WENO method. Rathan et al. [29] designed a new smoothness indicator based on the $L^1$-norm. Recently, Jiang [19] developed another WENO method for nonlinear degenerate parabolic equations.

Lately, machine learning methods have been widely used to numerically solve partial differential equations (PDEs). We refer to [23, 33, 10], where machine learning methods are directly used to approximate the solution of a given PDE problem. Bar-Sinai et al. [7] used neural networks to approximate a spatial derivative on a low-resolution grid. Beck et al. [9] used methods from edge detection to better capture shocks and discontinuities.

The idea of improving the WENO method for solving hyperbolic conservation laws using machine learning was presented by Stevens and Colonius [34]. The original smoothness indicators are retained and the finite volume coefficients of the original WENO scheme are perturbed using a neural network algorithm. However, the resulting scheme does not achieve the high order of accuracy, but is reduced to the first order. A further improvement of the WENO method on hyperbolic conservation laws was recently performed by Kossaczká et al. [21]. In this work, the smoothness indicators of the original WENO method are perturbed by training a relatively small neural network so that the high order of convergence is preserved, which was also proved theoretically.

In this work, we aim to generalize the algorithm of [21] also for the nonlinear degenerate parabolic equations. We use a neural network algorithm to modify the smoothness indicators of the original WENO scheme [25, 15], obtaining sixth-order convergence, which we prove theoretically. We emphasize that no post-processing steps need to be performed to maintain the consistency and convergence of the method, which also increases the efficiency of the deep learning algorithm. We extend the method to two-dimensional problems and, in contrast to [21], use a novel effective training procedure and a neural network structure.

The paper is organized as follows. In Section 2, we present the general framework of the WENO method from [25] and [15]. Next, in Section 3, we explain how the smoothness indicators are modified using the Deep Learning algorithm. The convergence of the new method is also proved in this section. In Section 4, we describe the structure of the neural network used in this paper and explain the training procedure. Moreover, in Section 5 we explain how we proceed in two-dimensional problems. We present the numerical results in Section 6, where we demonstrate the improvement with figures and tables. Finally, concluding remarks are made in Section 7.

## 2. The WENO scheme.

We firstly describe the general WENO discretization to solve (1.2) as developed in [25] and later in [15]. We introduce the uniform grid defined by the points $x_i = x_0 + i\Delta x$ with the cell boundaries $x_{i+\frac{1}{2}} = x_i + \frac{\Delta x}{2}$, $i = 0, \dots, N$. The semi-discrete formulation of (1.2) can be written as

$$\frac{du_i(t)}{dt} = \frac{\hat{f}_{i+\frac{1}{2}} - \hat{f}_{i-\frac{1}{2}}}{\Delta x^2}, \tag{2.1}$$

where $u_i(t)$ approximates pointwise $u(x_i, t)$ and the numerical flux $\hat{f}_{i+\frac{1}{2}}$ is chosen such that for all sufficiently smooth $u$

$$\frac{1}{\Delta x^2}\Big(\hat{f}_{i+\frac{1}{2}} - \hat{f}_{i-\frac{1}{2}}\Big) = \big(b(u)\big)_{xx}|_{x=x_i} + O(\Delta x^6), \tag{2.2}$$

with sixth order of accuracy. Following [25] if we implicitly define a function $h$ by

$$b\big(u(x)\big) = \frac{1}{\Delta x^2}\int_{x-\frac{\Delta x}{2}}^{x+\frac{\Delta x}{2}} \left(\int_{\eta-\frac{\Delta x}{2}}^{\eta+\frac{\Delta x}{2}} h(\xi)\, d\xi\right) d\eta, \tag{2.3}$$

then

$$\big(b(u)\big)_{xx} = \frac{1}{\Delta x^2}\big[h(x+\Delta x) - 2h(x) + h(x-\Delta x)\big] \tag{2.4}$$

and with the function

$$g(x) = h\Big(x + \frac{\Delta x}{2}\Big) - h\Big(x - \frac{\Delta x}{2}\Big), \tag{2.5}$$

it holds that

$$\big(b(u)\big)_{xx}|_{x=x_i} = \frac{g(x+\frac{\Delta x}{2}) - g(x-\frac{\Delta x}{2})}{\Delta x^2}. \tag{2.6}$$

Let us now consider a 6-point stencil corresponding to sixth order discretization

$$S(i) = \{x_{i-2}, \dots, x_{i+3}\}. \tag{2.7}$$

This will be divided into three candidate substencils given by

$$S(i)^m = \{x_{i-2+m}, \dots, x_{i+1+m}\}, \qquad m = 0, 1, 2. \tag{2.8}$$

On each of these substencils, the numerical flux $\hat{f}^m_{i\pm\frac{1}{2}}$ needs to be calculated. Let $\hat{f}^m(x)$ be the polynomial approximation of $g(x)$ on each of the substencils (2.8). By an evaluation of these polynomials at $x = x_{i+\frac{1}{2}}$ following formulas from [25] can be obtained:

$$\begin{aligned}
\hat{f}^0_{i+\frac{1}{2}} &= \frac{b(u_{i-2}) - 3b(u_{i-1}) - 9b(u_i) + 11b(u_{i+1})}{12}, \\
\hat{f}^1_{i+\frac{1}{2}} &= \frac{b(u_{i-1}) - 15b(u_i) + 15b(u_{i+1}) - b(u_{i+2})}{12}, \\
\hat{f}^2_{i+\frac{1}{2}} &= \frac{-11b(u_i) + 9b(u_{i+1}) + 3b(u_{i+2}) - b(u_{i+3})}{12},
\end{aligned} \tag{2.9}$$

and by shifting each index by $-1$ we obtain the numerical fluxes $\hat{f}^m_{i-\frac{1}{2}}$. The linear combination of the fluxes (2.9) gives the final approximation on the big stencil (2.7)

$$\hat{f}_{i+\frac{1}{2}} = \sum_{m=0}^{2} d_m \hat{f}^m_{i+\frac{1}{2}}, \tag{2.10}$$

where $d_m$ are the linear weights, which values are

$$d_0 = -\frac{2}{15}, \qquad d_1 = \frac{19}{15}, \qquad d_2 = -\frac{2}{15}. \tag{2.11}$$

They are also called "ideal weights" as they would yields the central sixth order scheme. As it can be seen, the linear weights $d_0$ and $d_2$ are negative. Therefore, the final WENO scheme may be unstable and a special technique treating negative weights has to be used [30]. The weights $d_m$ are then split into positive and negative parts, such that it holds

$$d_m = \sigma^+ \gamma^+_m - \sigma^- \gamma^-_m. \tag{2.12}$$

Following [25] we get the values

$$\begin{aligned}
\gamma^+_0 &= \frac{1}{21}, \quad \gamma^+_1 = \frac{19}{21}, \quad \gamma^+_2 = \frac{1}{21}, \\
\gamma^-_0 &= \frac{4}{27}, \quad \gamma^-_1 = \frac{19}{27}, \quad \gamma^-_2 = \frac{4}{27},
\end{aligned} \tag{2.13}$$

and

$$\sigma^+ = \frac{42}{15}, \quad \sigma^- = \frac{27}{15}. \tag{2.14}$$

Finally, the numerical flux for the WENO scheme can be approximated by

$$\hat{f}_{i+\frac{1}{2}} = \sum_{m=0}^{2} \omega_m \hat{f}^m_{i+\frac{1}{2}}, \tag{2.15}$$

with

$$\omega_m = \sigma^+ \alpha^+_m - \sigma^- \alpha^-_m, \tag{2.16}$$

$$\alpha_m^{\pm} = \frac{\tilde{\alpha}_m^{\pm}}{\sum_{i=0}^{2} \tilde{\alpha}_i^{\pm}}, \qquad \tilde{\alpha}_m^{\pm} = \frac{\gamma_m^{\pm}}{(\epsilon + \beta_m)^2}, \qquad m = 0, 1, 2. \tag{2.17}$$

The parameter $\epsilon$ is used to prevent the denominator from becoming zero, and $\beta_m$ is referred to as the smoothness indicator, which plays the crucial role in deciding which substencils should be chosen for the final flux approximation.

**2.1. Smoothness indicators.** In this section we analyze the smoothness indicators $\beta_m$ as proposed in [18]. They are defined as:

$$\beta_m = \sum_{q=1}^{2} \Delta x^{2q-1} \int_{x_i}^{x_{i+1}} \left( \frac{d^q \hat{f}^m(x)}{dx^q} \right)^2 dx, \tag{2.18}$$

with $\hat{f}^m(x)$ being the polynomial approximation in each of three substencils. There is only one difference from [18], namely that the integration must be over the interval $[x_i, x_{i+1}]$ to satisfy the symmetry property of the parabolic equation. The explicit forms of these indicators corresponding to the flux approximation $\hat{f}_{i+\frac{1}{2}}$ can be obtained as

$$\begin{aligned}
\beta_0 &= \frac{13}{12}\big(b(u_{i-2}) - 3b(u_{i-1}) + 3b(u_i) - b(u_{i+1})\big)^2 \\
&\quad + \frac{1}{4}\big(b(u_{i-2}) - 5b(u_{i-1}) + 7b(u_i) - 3b(u_{i+1})\big)^2, \\
\beta_1 &= \frac{13}{12}\big(b(u_{i-1}) - 3b(u_i) + 3b(u_{i+1}) - b(u_{i+2})\big)^2 \\
&\quad + \frac{1}{4}\big(b(u_{i-1}) - b(u_i) - b(u_{i+1}) + b(u_{i+2})\big)^2, \\
\beta_2 &= \frac{13}{12}\big(b(u_i) - 3b(u_{i+1}) + 3b(u_{i+2}) - b(u_{i+3})\big)^2 \\
&\quad + \frac{1}{4}\big(-3b(u_i) + 7b(u_{i+1}) - 5b(u_{i+2}) + b(u_{i+3})\big)^2
\end{aligned} \tag{2.19}$$

and the Taylor expansion at $x_i$ gives

$$\begin{aligned}
\beta_0 &= b_{xx}^2 \Delta x^4 + b_{xx}^2 f_{xxx} \Delta x^5 + \left( \frac{4}{3} b_{xxx}^2 - \frac{1}{3} b_{xx} b_{xxxx} \right) \Delta x^6 \\
&\quad + \left( \frac{1}{4} b_{xx} b_{xxxxx} - \frac{5}{4} b_{xxx} b_{xxxx} \right) \Delta x^7 + O(\Delta x^8), \\
\beta_1 &= b_{xx}^2 \Delta x^4 + b_{xx}^2 b_{xxx} \Delta x^5 + \left( \frac{4}{3} b_{xxx}^2 + \frac{2}{3} b_{xx} b_{xxxx} \right) \Delta x^6 \\
&\quad + \left( \frac{1}{4} b_{xx} b_{xxxxx} + \frac{17}{12} b_{xxx} b_{xxxx} \right) \Delta x^7 + O(\Delta x^8), \\
\beta_2 &= b_{xx}^2 \Delta x^4 + b_{xx}^2 b_{xxx} \Delta x^5 + \left( \frac{4}{3} b_{xxx}^2 - \frac{1}{3} b_{xx} b_{xxxx} \right) \Delta x^6 \\
&\quad + \left( -\frac{3}{4} b_{xx} b_{xxxxx} + \frac{37}{12} b_{xxx} b_{xxxx} \right) \Delta x^7 + O(\Delta x^8).
\end{aligned} \tag{2.20}$$

For more details and the convergence analysis we refer the reader to [25]. It has been shown, that for the sixth order accuracy the following necessary and sufficient conditions have to be satisfied:

$$\begin{aligned} \sum_{m=0}^{2}(\omega_m - d_m) &= O(\Delta x^8), \\ \omega_m - d_m &= O(\Delta x^3), \\ \omega_0 - \omega_2 &= O(\Delta x^4). \end{aligned} \tag{2.21}$$

As it was shown in [25], the smoothness indicators (2.19) with nonlinear weights (2.16)-(2.17) do not fulfill the conditions (2.21). Therefore, the mapped function as introduced in [17] was used by Liu et al. [25].

**2.2. The MWENO scheme.** Alternatively, Hajipour and Malek [15] defined new nonlinear weights using

$$\alpha_m^{\pm} = \frac{\tilde{\alpha}_m^{\pm}}{\sum_{i=0}^{2}\tilde{\alpha}_i^{\pm}}, \qquad \tilde{\alpha}_m^{\pm} = \gamma_m^{\pm}\left[1 + \left(\frac{\tau_7}{\beta_m + \epsilon}\right)^2\right], \qquad m = 0,1,2, \tag{2.22}$$

and then inserting into (2.16) with

$$\tau_7 = |\beta_0 - \beta_2|. \tag{2.23}$$

From (2.20) it can be seen that

$$\tau_7 = \left|-b_{xx}b_{xxxxx} + \frac{13}{3}b_{xxx}b_{xxxx}\right|\Delta x^7 + O(\Delta x^8). \tag{2.24}$$

It has been shown [15], that using these nonlinear weights the conditions (2.21) are satisfied and the sixth-order accuracy is ensured.

## 3. Application of Deep Learning to the sixth-order WENO Scheme.

Solving nonlinear degenerate parabolic equations is a challenging task in most cases. Not only because of the possible existence of non-smooth solutions or sharp fronts, but also because of the finite propagation speed of the wave fronts. This gives us enough room to improve the existing methods. In [21], new smoothness indicators for the fifth-order WENO-DS scheme were developed using Deep Learning. They were defined as the product of the original smoothness indicators $\beta_m$ and perturbations $\delta_m$, where $\delta_m$ are the outputs of a particular neural network algorithm:

$$\beta_m^{DS} = \beta_m(\delta_m + C), \tag{3.1}$$

where $C$ is a constant that ensures the consistency and convergence of the new method and will be further discussed in subsection 3.1. We apply this idea and modify the smoothness indicators (2.19) in the same way to improve the sixth-order WENO scheme.

We proceed as described in [21] and use the same multiplier $\delta_{m,i}$ for both $\beta_{m,i+\frac{1}{2}}$ and $\beta_{m,i-\frac{1}{2}}$, which are the smoothness indicators used for the flux reconstruction at points $x_{i-\frac{1}{2}}$ and $x_{i+\frac{1}{2}}$ for the approximation of the solution in $x_i$. $\beta_{m,i+\frac{1}{2}}$ is given in (2.19) and $\beta_{m,i-\frac{1}{2}}$ is obtained by shifting each index by $-1$. The new smoothness indicators are:

$$\begin{aligned} \beta_{m,i+\frac{1}{2}}^{DS} &= \beta_{m,i+\frac{1}{2}}(\delta_{m,i} + C), \\ \beta_{m,i-\frac{1}{2}}^{DS} &= \beta_{m,i-\frac{1}{2}}(\delta_{m,i} + C), \end{aligned} \tag{3.2}$$

and the values $\delta_0$, $\delta_1$, $\delta_2$ are obtained, such that it holds

$$\delta_{0,i+1} = \delta_{1,i} = \delta_{2,i-1}, \quad i = 0, \dots, N. \tag{3.3}$$

For more details we refer to [21].

**3.1. Convergence analysis.** In this section we analyze the convergence of the new WENO-DS method formulated by the following properties and theorem.

PROPERTY 3.1. *Let the neural network, represented by a function $F(\cdot)$, have the structure assuring its spatial invariance. Further, let all hidden layers of this neural network be differentiable functions. Then, the multipliers $\delta_{m,i}$ from (3.2) in the node $x_i$ satisfying (3.3) can be expressed as the outputs of the neural network function $F(\cdot)$:*

$$\begin{aligned} \delta_{0,i} &= F\big(\bar{b}(\bar{x}_{i-1})\big) = \Phi(\bar{x}_i - \Delta x) = \Phi(\bar{x}_i) - O(\Delta x), \\ \delta_{1,i} &= F\big(\bar{b}(\bar{x}_i)\big) = \Phi(\bar{x}_i), \\ \delta_{2,i} &= F\big(\bar{b}(\bar{x}_{i+1})\big) = \Phi(\bar{x}_i + \Delta x) = \Phi(\bar{x}_i) + O(\Delta x), \end{aligned} \tag{3.4}$$

*where*

$$\begin{aligned} \bar{x}_i &= (x_{i-k}, x_{i-k+1}, \dots, x_{i+k}), \\ \bar{b}(\bar{x}_i) &= (b(x_{i-k}), b(x_{i-k+1}), \dots, b(x_{i+k})), \end{aligned} \tag{3.5}$$

*with $2k+1$ being the size of the receptive field of the whole neural network and $\Phi$ being the function composition $F \circ \bar{b}$.*

PROPERTY 3.2. *Let the constant $C$ in (3.2) be chosen such that it holds $\Phi(\bar{x}_i) + C > \kappa > 0$ with $\kappa$ fixed and $\Phi$ defined as in Property 3.1.*

THEOREM 3.1. *Let the numerical flux of the WENO-DS scheme be given by (2.9) and (2.15) with the corresponding nonlinear weights given by (2.16) and*

$$\alpha_m^{\pm} = \frac{\tilde{\alpha}_m^{\pm}}{\sum_{i=0}^{2} \tilde{\alpha}_i^{\pm}}, \qquad \tilde{\alpha}_m^{\pm} = \gamma_m^{\pm} \left[ 1 + \left( \frac{\tau_{7,i+\frac{1}{2}}}{\beta_{m,i+\frac{1}{2}}^{DS} + \epsilon} \right)^2 \right], \qquad m = 0, 1, 2, \tag{3.6}$$

*with $\gamma_m^{\pm}$ given by (2.13), $\beta_{m,i+\frac{1}{2}}^{DS}$ defined in (3.2) and $\tau_7$ defined by*

$$\tau_7 = \left| \beta_{0,i+\frac{1}{2}} - \beta_{2,i+\frac{1}{2}} \right|. \tag{3.7}$$

*To define a negative flux $\hat{f}_{i-\frac{1}{2}}$, (3.6) is used with $\beta_{m,i+\frac{1}{2}}^{DS}$ being replaced by $\beta_{m,i-\frac{1}{2}}^{DS}$ from (3.2) Next, (3.7) is used with $\beta_{m,i+\frac{1}{2}}$ from (2.19) being replaced by $\beta_{m,i-\frac{1}{2}}$ obtained by shifting each index in (2.19) by $-1$. Let the multipliers $\delta_{m,i}$ in (3.2) be the output of a neural network algorithm satisfying the Property 3.1 and Property 3.2. Then, the resulting WENO-DS method (2.1) for smooth solutions of the nonlinear degenerate parabolic equation (1.1) exhibits a sixth-order accuracy.*

*Proof.* From (2.20) we see that

$$\beta_{m,i\pm\frac{1}{2}} = b_{xx}^2 \Delta x^4 + O(\Delta x^5), \tag{3.8}$$

and from (2.24)

$$\tau_{7,i\pm\frac{1}{2}} = O(\Delta x^7). \tag{3.9}$$

Then using Property 3.1 it holds

$$\begin{aligned}\beta^{DS}_{m,i\pm\frac{1}{2}} &= \beta_{m,i\pm\frac{1}{2}}(\delta_{m,i}+C) = \big(b_{xx}^2\Delta x^4 + O(\Delta x^5)\big)\big(\Phi(\bar{x}_i)+O(\Delta x)+C\big)\\ &= b_{xx}^2 P(\bar{x}_i)\Delta x^4 + O(\Delta x^5),\end{aligned} \tag{3.10}$$

with $P(\bar{x}_i) = \Phi(\bar{x}_i) + C$ and $C$ satisfying Property 3.2. Then $P(\bar{x}_i) = O(1)$ is ensured. Assuming $b_{xx} \neq 0$, it holds

$$\frac{\tau_{7,i\pm\frac{1}{2}}}{\beta^{DS}_{m,i\pm\frac{1}{2}}} = \hat{D}\Delta x^3 + O(\Delta x^4), \qquad \hat{D} = \frac{|-b_{xx}b_{xxxxxx} + \frac{13}{3}b_{xxx}b_{xxxx}|}{b_{xx}^2 P(\bar{x}_i)}. \tag{3.11}$$

We take $\epsilon = 0$, substitute now this into (3.6) (for simplicity we drop the index $i \pm \frac{1}{2}$) and obtain

$$\tilde{\alpha}^{\pm}_m = \gamma^{\pm}_m \left[1 + \left(\frac{\tau_7}{\beta^{DS}_m + \epsilon}\right)^2\right] = \gamma^{\pm}_m\big(1 + O(\Delta x^6)\big), \tag{3.12}$$

and

$$\alpha^{\pm}_m = \frac{\gamma^{\pm}_m \left[1 + \left(\frac{\tau_7}{\beta^{DS}_m + \epsilon}\right)^2\right]}{\sum_{i=0}^2 \gamma^{\pm}_m\big(1 + O(\Delta x^6)\big)}, \tag{3.13}$$

which implies

$$\gamma^{\pm}_m = \alpha^{\pm}_m \frac{1}{1 + \left(\frac{\tau_7}{\beta^{DS}_m + \epsilon}\right)^2} \sum_{i=0}^{2} \gamma^{\pm}_m\big(1 + O(\Delta x^6)\big) = \alpha^{\pm}_m + O(\Delta x^6), \tag{3.14}$$

where we used $\sum_{i=0}^2 \gamma^{\pm}_m = 1$.

We investigate now the conditions (2.21). Due to the normalization we see that $\sum_{i=0}^2 \alpha^{\pm}_m = 1$. Inserting this into (2.16) and using (2.14) we have $\sum_{i=0}^2 \omega_m = 1$. From (2.11) we conclude that $\sum_{i=0}^2 d_m = 1$ and the first condition is always fulfilled. Then using (3.14), inserting into (2.12) and using (2.16) we fulfill also the second condition:

$$d_m = \sigma^+\big(\alpha^+_m + O(\Delta x^6)\big) - \sigma^-\big(\alpha^-_m + O(\Delta x^6)\big) = \omega_m + O(\Delta x^6). \tag{3.15}$$

Finally, realizing that $\gamma^{\pm}_0 = \gamma^{\pm}_2$, the third condition is also fulfilled:

$$\begin{aligned}\omega_0 - \omega_2 &= \sigma^+\alpha^+_0 - \sigma^-\alpha^-_0 - \sigma^+\alpha^+_2 + \sigma^-\alpha^-_2 = \sigma^+\big(\gamma^+_0 + O(\Delta x^6)\big)\\ &\quad - \sigma^-\big(\gamma^-_0 + O(\Delta x^6)\big) - \sigma^+\big(\gamma^+_2 + O(\Delta x^6)\big) + \sigma^-\big(\gamma^-_2 + O(\Delta x^6)\big)\\ &= O(\Delta x^6)\end{aligned} \tag{3.16}$$

and the sixth-order convergence of the WENO-DS method for smooth solutions of nonlinear degenerate parabolic equation (1.1) is ensured. ∎

**4. The structure of a neural network and training procedure.** In our application, we use the *convolutional neural network (CNN)*. Here, we ensure the spatial invariance of the resulting numerical scheme and make the multipliers $\delta_m$ independent of their position in the spatial grid. Then we use the differentiable activation function *exponential linear unit (ELU)* for all hidden layers. In the output layer, we use either a sigmoid activation function or no activation function. The number of its hidden layers, kernel size, and number of channels are chosen separately for each of the equation classes. Our goal is to keep the CNN as small as possible, while still achieving the best possible results. In all our experiments, we set $C = 0.1$ in (3.2) and the value of $\epsilon$ to $10^{-13}$.

Since we want to improve the smoothness indicators, we first calculate the first and second central finite differences of $b(x_i)$, $i = 0, \dots, N$. From these parameters we obtain the information about the smoothness of the solution and they represent an effective preprocessing of the given data. The input values for the first learned hidden layer are:

$$b_{\text{diff1}} = b(x_{i+1}) - b(x_{i-1}), \quad b_{\text{diff2}} = b(x_{i+1}) - 2b(x_i) + b(x_{i-1}). \tag{4.1}$$

Now we explain how the training procedure is performed. First, the weights of the CNN are randomly initialized and a problem is selected from a data set. The computational domain is divided into $N \times M$ steps, where $N$ is a number of space steps and $M$ is a number of time steps. One possibility would then be to continue as described in [21], where we successively computed the entire solution up to the final time $T$. We used the solution at time step $n$ and calculated the solution at time step $n+1$ and during this calculation the CNN was used to predict the multipliers of the smoothness indicators. After each of these time steps, we calculated the loss and its gradient with respect to the weights of the CNN using the backpropagation algorithm. We repeated these steps until the final time $T$ and in this time step we tested our model on a validation set.

We introduce a novel training procedure in this paper. At the beginning of the training, we select a problem from a dataset. Then we perform one time step and use the CNN to predict the multipliers of the smoothness indicators. Then we compute the loss and its gradient with respect to the weights of the CNN. After this step, however, we do not automatically proceed to the next time step; instead, we randomly decide whether to proceed to the next time step of a current problem or to choose another problem from our data set and run one time step of that problem. The probability of choosing the new problem is determined at the beginning of the training session. We use the probability $\varphi = 0.1$ in our trainings. This means that we select a new problem from a dataset with probability $\varphi = 0.1$. We remember all opened problems and if no new problem is opened (with probability $1 - \varphi$), we proceed to execute the next time step of a problem uniformly sampled from the set of already opened problems. After each of these time steps, the loss and its gradient with respect to the weights of the CNN are calculated. The gradient is then used to update the weights.

To improve the gradient propagation into the lower layers, we use the residual learning framework [16]. It may happen that when using a deeper neural network, its effectiveness is compromised, which is not caused by overfitting, as reported in [16]. The idea is to introduce a so-called *identity mapping* that only adds the output of the previous layer to the output of the next layer. It is important that neither additional parameters nor computational complexity are added.

To update the weights of the CNN we use the Adam optimizer [20]. The optimizer parameters will be specified for each of the equation classes separately. As the default loss function we use the mean square error

$$LOSS_{\text{MSE}}(u) = \frac{1}{N} \sum_{i=0}^{N} (u_i - u_i^{\text{ref}})^2, \tag{4.2}$$

where $u_i$ is a numerical approximation of $u(x_i)$ and $u_i^{\text{ref}}$ denotes the corresponding reference solution. For the implementation we use Python with the library Pytorch [28].

**5. Two-dimensional implementation.** Here we consider the two-dimensional form of (1.1):

$$u_t = b_1(u)_{xx} + b_2(u)_{yy}. \tag{5.1}$$

The procedure described in Section 2 can be easily applied dimension-by-dimension to obtain the approximations of numerical fluxes $\hat{f}_{i+\frac{1}{2}}$ and $\hat{k}_{i+\frac{1}{2}}$, such that it holds

$$\begin{aligned} \frac{1}{\Delta x^2}\left(\hat{f}_{i+\frac{1}{2}} - \hat{f}_{i-\frac{1}{2}}\right) &= \big(b_1(u)\big)_{xx}|_{(x_i,y_j)} + O(\Delta x^6), \\ \frac{1}{\Delta y^2}\left(\hat{k}_{i+\frac{1}{2}} - \hat{k}_{i-\frac{1}{2}}\right) &= \big(b_2(u)\big)_{yy}|_{(x_i,y_j)} + O(\Delta x^6), \end{aligned} \tag{5.2}$$

using the uniform grid with nodes $(x_i, y_j)$, $\Delta x = x_{i+1} - x_i$, $\Delta y = y_{j+1} - y_j$. The corresponding semi-discrete form of (5.1) takes the form

$$\frac{du_i(t)}{dt} = \frac{\hat{f}_{i+\frac{1}{2}} - \hat{f}_{i-\frac{1}{2}}}{\Delta x^2} + \frac{\hat{k}_{i+\frac{1}{2}} - \hat{k}_{i-\frac{1}{2}}}{\Delta y^2}. \tag{5.3}$$

We could use two-dimensional CNN for training in this case to see if the information from the second dimension can improve the smoothness indicators in the first dimension. However, experimentally we got better results with one-dimensional CNNs in each direction.

**6. Numerical Results.** In this section, we present the numerical results to show the efficiency of the proposed numerical scheme WENO-DS based on the neural network algorithm. We use the nonlinear weights (2.22), replacing $\beta_m$ with $\beta_m^{DS}$ (3.2). This is done to discretize the diffusion term and for the discretization of the advection term, which later appears in the examples, we use an analogous procedure as described in [21]. Then the following system of ordinary differential equations (ODEs) has to be solved

$$\frac{du(t)}{dt} = L(u). \tag{6.1}$$

For this purpose we use a third-order total variation diminishing (TVD) Runge-Kutta method [31] given by

$$\begin{aligned} u^{(1)} &= u^n + \Delta t L(u^n), \\ u^{(2)} &= \frac{3}{4}u^n + \frac{1}{4}u^{(1)} + \frac{1}{4}\Delta t L(u^{(1)}), \\ u^{n+1} &= \frac{1}{3}u^n + \frac{2}{3}u^{(2)} + \frac{2}{3}\Delta t L(u^{(2)}), \end{aligned} \tag{6.2}$$

where $u^n$ is the numerical solution at the time step $n$.

For solving, we use the time step of the one-dimensional problems

$$u_t + f(u)_x = b(u)_{xx}, \tag{6.3}$$

such that

$$\frac{0.4}{\Delta t} = \frac{\max_u |f'(u)|}{\Delta x} + \frac{\max_u |b'(u)|}{\Delta x^2}. \tag{6.4}$$

For two-dimensional problems

$$u_t + f_1(u)_x + f_2(u)_y = b_1(u)_{xx} + b_2(u)_{yy}, \tag{6.5}$$

the time step is set as

$$\frac{0.4}{\Delta t} = \frac{\max_u |f_1'(u)|}{\Delta x} + \frac{\max_u |f_2'(u)|}{\Delta y} + \frac{\max_u |b_1'(u)|}{\Delta x^2} + \frac{\max_u |b_2'(u)|}{\Delta y^2}. \tag{6.6}$$

**6.1. The porous medium equation.** As the first example we apply the CNN algorithm to enhance the numerical solution of the porous medium equation (1.2) with (1.3).

The *Barenblatt solution* [8, 37] is a weak solution of the PME with the explicit form

$$B_m(\mathbf{x}, t) = t^{-\alpha} \left[ \left( 1 - k|\mathbf{x}|^2 t^{-\frac{2\alpha}{d}} \right)^+ \right]^{\frac{1}{m-1}}, \quad t > 0, \quad \mathbf{x} \in \Omega \subseteq \mathbb{R}^d, \quad m > 1, \tag{6.7}$$

where $v^+ = \max(v, 0)$ and $k = \frac{\alpha(m-1)}{2md}$ with $\alpha = \frac{d}{(m-1)d+2}$. For $d = 1$, the compact support of this Barenblatt solution is the interval $[-a_m(t), a_m(t)]$, where

$$a_m(t) = \sqrt{\frac{2m}{\alpha(m-1)}}\, t^{\alpha}, \tag{6.8}$$

with $\alpha = \frac{1}{m+1}$. The solution is not differentiable at the interface points $x = \pm a_m(t)$ [26].

In our numerical experiments, we take as initial condition the Barenblatt solution (6.7) at time $t = 1$. We use zero boundary conditions $u(\pm 6, t) = 0$ for $t > 1$ and divide the computational domain into 64 uniform cells.

For the training, we proceed as described in Section 4. When a new problem is to be selected from a data set, an exponent $m$ in PME (1.3) is chosen such that $m \in \mathcal{U}(2, 8)$. In this way, we cover a wide range of different problems and the final numerical scheme can be reliably used for different values of $m$. For training, we fix $T = 1.4$. We use a rather small CNN with only three hidden layers. The structure is described in Figure 1, where also the number of channels and the kernel size can be found.

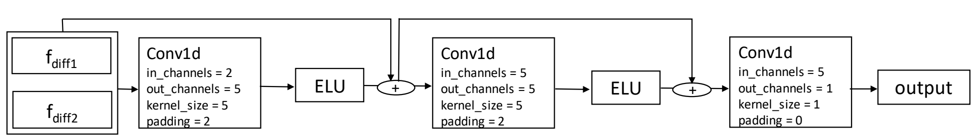



Fig. 1: A structure of the convolutional neural network used for the porous medium equation.

We use the loss function (4.2), where a reference solution is computed from (6.7). To match the training contribution from very small loss problems to large loss problems, we use the following loss scaling:

$$LOSS_{\text{MSE}}(u) = \begin{cases} 10^2 LOSS_{\text{MSE}}(u), & \text{if} \quad LOSS_{\text{MSE}}(u) < 10^4, \\ 10\sqrt{LOSS_{\text{MSE}}(u)}, & \text{otherwise.} \end{cases} \tag{6.9}$$

To update the weights we use the Adam optimizer with learning rate 0.1.

Due to the rather large variance of the training, we performed 20 trainings and selected the one that gave the best results on a validation set. We present the history of the value of the loss function for the problems from the validation set on Figure 2. We tested our model every 5 training steps and the loss was evaluated at time $T = 2$. The validation set contains PME problems with different exponents $m$ generated randomly. We rescale the loss values for each validation problem to be in the interval $[0, 1]$. It can be seen that the low values are obtained after a fairly small number of training steps.

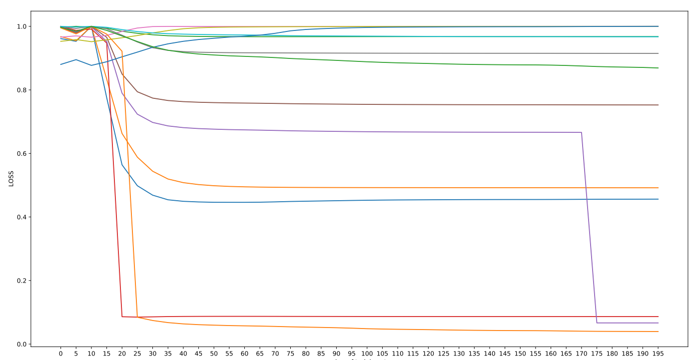
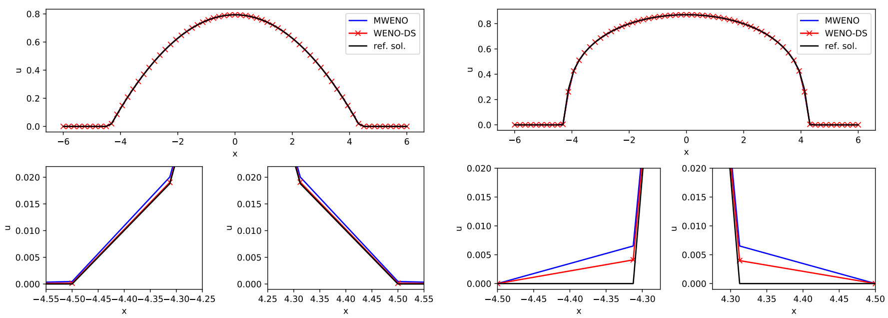



Fig. 2: Loss values for different validation problems evaluated each 5 training steps.

In some cases, we see that the loss value increases slightly as the number of iterations increases. This is because we want to optimize the method for a wide range of parameters $m$ and also over the entire time domain. However, we conclude that in most cases, which we demonstrate later in the tables and figures, the improvement outweighs a slight increase in the error that occurs in a rather small number of cases. We take the model obtained after the 195th training step as our final WENO-DS scheme. Here the loss values are stable and we found experimentally that further training would lead to overfitting, so the suboptimal results would be obtained.

We show the results on problems from the test set. These were not in the training or validation set. In Figure 3 we present the solution of the PME for $m = 2, 4$. We observe that WENO-DS yields a better solution in the regions where discontinuity occurs. This also affects the $L_\infty$ and $L_2$ errors, whose values we compare in the Table 1. We compare the errors for different parameters $m$ and $T$ and highlight the best performing WENO method in bold. We divide the error of the MWENO method by the error of WENO-DS in the column labeled 'ratio' to show how well our method performs compared to the original method. We realize that our new method outperforms the MWENO method in most cases.

Finally, we demonstrate the theoretically proven sixth-order of convergence for a




(a) $m = 2$ (b) $m = 4$

Fig. 3: Comparison of the MWENO and WENO-DS methods for the numerical solution of the porous medium equation with various parameter values $m$, $N = 64$.

heat equation with smooth initial condition given by

$$u_t = u_{xx}, \quad u(x, 0) = \sin(x), \quad -\pi \leq x \leq \pi, \quad 0 \leq t \leq 1. \tag{6.10}$$

The exact solution for this example is

$$u(x, t) = e^{-t} \sin x \tag{6.11}$$

and we take the boundary conditions from the exact solution for this case. The results can be found in Table 2 and we observe that the sixth-order convergence is ensured. Let us note, that we used for these results the same WENO-DS scheme which was an output of the learning procedure for the porous medium equation with the Barenblatt solution. We also did not retrain the CNN for different values of $N$.

**6.2. The convection-diffusion Buckley-Leverett equation.** In the next example we solve the convection-diffusion Buckley–Leverett equation of the form

$$u_t + f(u)_x = \epsilon \left(\nu(u) u_x\right)_x, \qquad \epsilon\, \nu(u) \geq 0. \tag{6.12}$$

This is a prototype model for oil reservoir simulations (two-phase flow). In our test we choose $\epsilon = 0.01$ and the flux function

$$f(u) = \frac{u^2}{u^2 + a(1-u)^2} \left(1 - g(1-u)^2\right), \tag{6.13}$$

where $a < 1$ is a constant representing the ratio of the viscosities of the two fluids and $g$ is a gravitational effect. Usually, $\nu(u)$ vanishes at some points so the equation (6.12) is a degenerate parabolic equation. Moreover, the sign of $f'(u)$ changes its sign, so the handling of the flux is more complicated. We choose

$$\nu(u) = \begin{cases} 4u(1-u), & 0 \leq u \leq 1, \\ 0, & \text{otherwise}, \end{cases} \tag{6.14}$$

so we obtain the parabolic term in a form

$$b(u) = \begin{cases} 0, & u < 0, \\ \epsilon\,(2u^2 - \frac{4}{3}u^3), & 0 \leq u \leq 1, \\ \frac{2}{3}\epsilon, & u > 1. \end{cases} \tag{6.15}$$

| | $L_\infty$ | | | $L_2$ | | |
|---|---|---|---|---|---|---|
| $m$ | MWENO | WENO-DS | ratio | MWENO | WENO-DS | ratio |
| 2 | 0.003257 | **0.001221** | 2.67 | 0.002689 | **0.001075** | 2.50 |
| 3 | 0.017395 | **0.014781** | 1.18 | 0.011163 | **0.009243** | 1.21 |
| 4 | 0.045135 | **0.040757** | 1.11 | 0.028080 | **0.025137** | 1.12 |
| 5 | 0.112800 | **0.105249** | 1.07 | 0.069098 | **0.064075** | 1.08 |
| 6 | 0.177022 | **0.173597** | 1.02 | 0.108670 | **0.104464** | 1.04 |
| 7 | **0.088695** | 0.090100 | 0.98 | **0.057645** | 0.058483 | 0.99 |
| 8 | **0.175060** | 0.179969 | 0.97 | **0.109320** | 0.111824 | 0.98 |

(a) $T = 1.2$

| | $L_\infty$ | | | $L_2$ | | |
|---|---|---|---|---|---|---|
| $m$ | MWENO | WENO-DS | ratio | MWENO | WENO-DS | ratio |
| 2 | 0.004877 | **0.003501** | 1.39 | 0.003013 | **0.002290** | 1.32 |
| 3 | 0.010907 | **0.008025** | 1.36 | 0.008057 | **0.005505** | 1.46 |
| 4 | 0.032591 | **0.029200** | 1.12 | 0.020487 | **0.018229** | 1.12 |
| 5 | 0.104031 | **0.097510** | 1.07 | 0.063717 | **0.059600** | 1.07 |
| 6 | 0.219481 | **0.214394** | 1.02 | 0.134668 | **0.130684** | 1.03 |
| 7 | 0.028863 | **0.023676** | 1.22 | 0.018625 | **0.012921** | 1.44 |
| 8 | **0.013782** | 0.014577 | 0.95 | **0.010280** | 0.011453 | 0.90 |

(b) $T = 1.5$

| | $L_\infty$ | | | $L_2$ | | |
|---|---|---|---|---|---|---|
| $m$ | MWENO | WENO-DS | ratio | MWENO | WENO-DS | ratio |
| 2 | 0.001235 | **0.001040** | 1.19 | 0.000952 | **0.000766** | 1.24 |
| 3 | 0.058471 | **0.056758** | 1.03 | 0.036008 | **0.034991** | 1.03 |
| 4 | 0.026741 | **0.018162** | 1.47 | 0.016951 | **0.011387** | 1.49 |
| 5 | 0.101398 | **0.092241** | 1.10 | 0.062115 | **0.056459** | 1.10 |
| 6 | 0.201053 | **0.194967** | 1.03 | 0.123476 | **0.119017** | 1.04 |
| 7 | 0.052631 | **0.047613** | 1.11 | 0.033208 | **0.027910** | 1.19 |
| 8 | 0.043306 | **0.039945** | 1.08 | 0.027796 | **0.024824** | 1.12 |

(c) $T = 2$

Table 1: Comparison of $L_\infty$ and $L_2$ error of MWENO and WENO-DS methods for the solution of the porous medium equation with various parameter $m$ and $T$. As 'ratio' we denote the error of the MWENO method divided by the error of WENO-DS (rounded to 2 decimal points).

The initial condition reads

$$u(x,0) = \begin{cases} 0, & 0 \le x \le 1 - \frac{1}{\sqrt{2}}, \\ 1, & 1 - \frac{1}{\sqrt{2}} < x \le 1, \end{cases} \tag{6.16}$$

and we divide the computational domain into 128 uniform cells.

In the training we proceed as in the previous example. As there exists no analytical solution in this case, we firstly create our data set, where we compute the

| | WENO-DS | | | |
|---|---|---|---|---|
| N | $L_\infty$ | Order | $L_2$ | Order |
| 20 | 6.148104e-06 | - | 4.858320e-06 | - |
| 40 | 5.641584e-08 | 6.767898 | 4.406364e-08 | 6.784725 |
| 80 | 8.366046e-10 | 6.075410 | 8.927014e-10 | 5.625267 |
| 160 | 1.300835e-11 | 6.007036 | 1.714365e-11 | 5.702432 |
| 320 | 1.962874e-13 | 6.050326 | 2.656299e-13 | 6.012112 |

Table 2: $L_\infty$ and $L_2$-norm error with convergence order of WENO-DS on (6.10)

reference solutions on fine grid for the equation (6.12). In this data set we consider the constants $a \in \mathcal{U}[0.1, 0.95]$ and $d \in \mathcal{U}[0, 6]$, divide the computational domain $[0, 1]$ into 1024 uniform cells and compute the solution up to time $T = 0.1$. We use the MWENO method [15] combined with the WENO-Z method [12] for the computation of these reference solutions.

The structure of the chosen CNN can be found in Figure 4. In the training we optimize not only the WENO-DS method to approximate the parabolic term, but also the WENO-DS [21] to approximate the hyperbolic term. The structure of the CNN remains the same for both cases. We use the loss function (4.2) and the Adam optimizer with the learning rate $10^{-5}$.

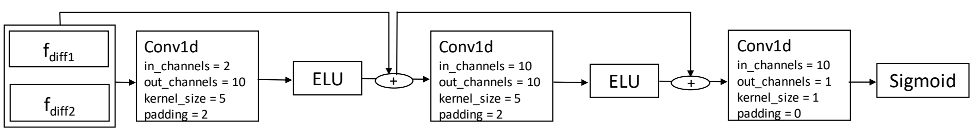



Fig. 4: A structure of the convolutional neural network used for the Buckley-Leverett equation.

We created a validation set with 12 different combinations of $a$ and $g$ generated randomly. On this set, we tested our model every 100 training steps. The Figure 5 shows how the value of the loss function changes as the number of training steps increases. We scaled the loss values again so that they are in $[0, 1]$. We see similar behavior to the Buckley-Leverett example of [21]. However, more than 2 optima can apparently be distinguished. There may be a small set of problems for which the optimum exists after only a few initial training steps (after zooming in to the bottom row region, we would see a slight decrease at the beginning, which is then replaced by an increase), the next optimum would be reached after about 5200 training steps, for the next set of problems we might see the optimum after 7600 training steps, and for the other set of problems the optimum would be reached after more than 8000 training steps. However, further training would not make sense because the error would become too large for the other set of problems. Finally, we choose the model obtained after the 4800th training step and present the results computed with this model.

We present the numerical solution of the Buckley-Leverett equation in Figure 6. We observe that our scheme provides a high quality of numerical solutions for both of these problems. Further, we compare the $L_\infty$ and $L_2$ errors of the problems from

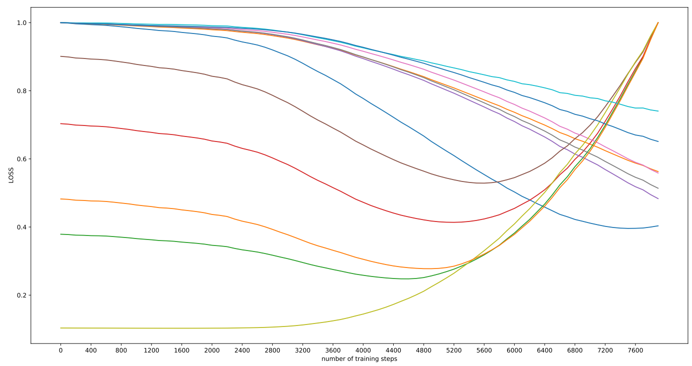
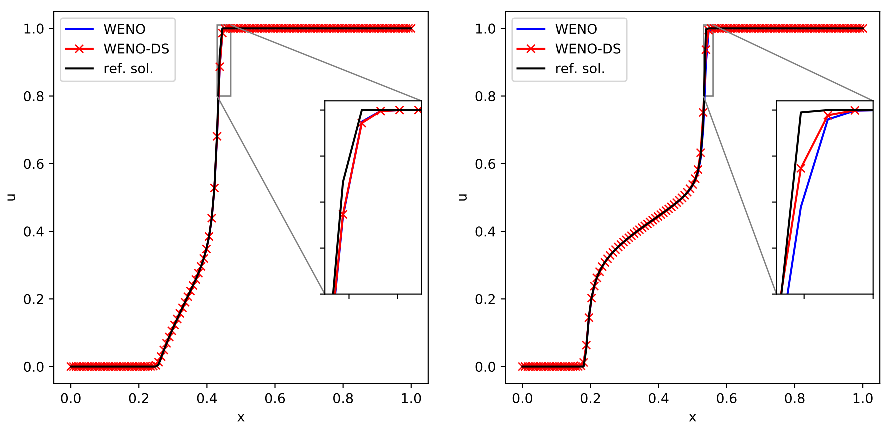



Fig. 5: Loss values for different validation problems evaluated each 100 training steps.

the test set with various parameters $a$ and $g$ in Table 3. We see, that in almost all cases our method provides smaller errors.




(a) $a = 1$ and $g = 0$

(b) $a = 1$ and $g = 5$

Fig. 6: Comparison of the original WENO (WENO-Z combined with MWENO) and WENO-DS methods for the numerical solution of the Buckley-Leverett equation with various parameters $a$ and $g$, $T = 0.1$, $N = 128$.

**6.3. The strongly degenerate parabolic convection-diffusion equation.** In this example we test the method trained on the Buckley-Leverett data from the previous example. We do not retrain the method and apply it to the strongly degenerate parabolic convection-diffusion equation of a form

$$u_t + f(u)_x = \epsilon \left(\nu(u) u_x\right)_x, \qquad \epsilon\, \nu(u) \geq 0. \tag{6.17}$$

| | | $L_\infty$ | | | $L_2$ | | |
|---|---|---|---|---|---|---|---|
| $a$ | $g$ | WENO | WENO-DS | ratio | WENO | WENO-DS | ratio |
| 1 | 5 | 0.102771 | **0.060673** | 1.69 | 0.009442 | **0.006242** | 1.51 |
| 1 | 0 | 0.037256 | **0.035068** | 1.06 | 0.003709 | **0.003493** | 1.06 |
| 1 | 3 | 0.081126 | **0.065823** | 1.23 | 0.007497 | **0.005990** | 1.25 |
| 0.75 | 5 | 0.065215 | **0.033212** | 1.96 | 0.006346 | **0.003856** | 1.65 |
| 0.75 | 4 | 0.077982 | **0.052171** | 1.49 | 0.007096 | **0.005972** | 1.19 |
| 0.5 | 5 | 0.086089 | **0.076892** | 1.12 | **0.008228** | 0.008637 | 0.95 |
| 0.5 | 2 | 0.045176 | **0.039770** | 1.14 | 0.004495 | **0.003955** | 1.14 |
| 0.5 | 1 | 0.030054 | **0.028264** | 1.06 | 0.003729 | **0.003603** | 1.03 |
| 0.3 | 3 | 0.035122 | **0.030277** | 1.16 | 0.004715 | **0.003233** | 1.46 |
| 0.25 | 4 | 0.083372 | **0.041290** | 2.02 | 0.007815 | **0.005713** | 1.37 |

Table 3: Comparison of $L_\infty$ and $L_2$ error of original WENO (WENO-Z combined with MWENO) and WENO-DS methods for the solution of the Buckley-Leverett equation with various parameters $a$ and $g$, $T = 0.1$. As 'ratio' we denote the error of the original WENO method divided by the error of WENO-DS (rounded to 2 decimal points).

This is a benchmark example presented e.g. in [19, 22, 25]. We take $\epsilon = 0.1$, $f(u) = u^2$ and

$$\nu(u) = \begin{cases} 0, & |u| \le 0.25, \\ 1, & |u| > 0.25. \end{cases} \tag{6.18}$$

This leads to a fact, that the equation is hyperbolic if $u \in [-0.25, 0.25]$ and parabolic elsewhere. The parabolic term takes a form

$$b(u) = \begin{cases} \epsilon\,(u + 0,25), & u < -0.25, \\ \epsilon\,(u - 0,25), & u > 0.25, \\ 0, & u \le |0.25|. \end{cases} \tag{6.19}$$

The initial condition is taken as

$$u(x,0) = \begin{cases} 1, & -\frac{1}{\sqrt{2}} - 0.4 < x \le -\frac{1}{\sqrt{2}} + 0.4, \\ -1, & \frac{1}{\sqrt{2}} - 0.4 < x \le \frac{1}{\sqrt{2}} + 0.4, \\ 0, & \text{otherwise}. \end{cases} \tag{6.20}$$

We use the zero boundary conditions and compute the solution to the final time $T = 0.7$ with $N = 128$ and $N = 256$. We present the numerical results in Figure 7 and see that our method is able to capture the discontinuities and sharp interfaces very well. The reference solution is obtained using MWENO and WENO-Z method with $N = 1024$.

**6.4. Two-dimensional porous medium equation.** In the next example we solve the two-dimensional PME in the form

$$u_t = (u^m)_{xx} + (u^m)_{yy}, \qquad m > 1. \tag{6.21}$$

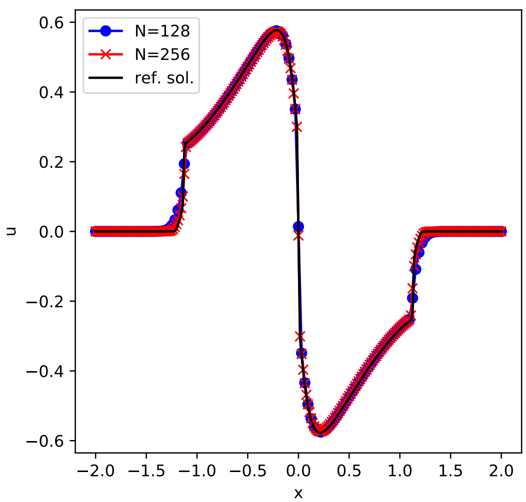



Fig. 7: Numerical solution of the strongly degenerate parabolic equation, $T = 0.7$.

As an initial condition we use a Barenblatt solution (6.7) at time $t = 1$ with $d = 2$. In this case, the Barenblatt solution has no derivative at the points of the circle $x^2+y^2 = \sqrt{\frac{4m}{\alpha(m-1)}}\, t^{\alpha}$, with $\alpha = \frac{1}{m}$. We choose the computational domain $\Omega = [-10, 10]$ and zero boundary condition $u = 0$ on the boundary $\partial\Omega$. We divide the computational domain into $64 \times 64$ space grid points.

In our training we proceed analogously to the one-dimensional PME example and again simulate the equation (6.21) for $m \in \mathcal{U}(2, 8)$ to make the final numerical scheme more robust. We use the same CNN structure as described in Figure 1, the same loss function (6.9) and Adam optimizer with the learning rate 0.1 to update the weights. We show the progress of the loss function on the Figure 8 and see that the stable values of the loss function are obtained after a few first training steps. We could take the model obtained after the 30th training step, where small values of loss for some problems are obtained, or the model obtained after the 40th training step. Here we obtain minimal value of loss for another class of problems. Both of them would give us sufficient results and we decided to compare the results of the model obtained after the 40th training step.

Alternatively, we can use the method which was an output of the training procedure for the one-dimensional porous medium equation from the subsection 6.1. We compare the errors of the both methods in the Table 4. We see that the results are very similar and also the method trained on a one-dimensional example can be reliably used in more-dimensional space. This observation can be very useful when the computation of a reference solution in more dimensions becomes too demanding. Figure 9 illustrates the solution for $m = 2$.

**6.5. Two-dimensional Buckley-Leverett equation.** As a last example we solve the two-dimensional Buckley-Leverett equation of the form

$$u_t + f_1(u)_x + f_2(u)_y = \epsilon\,(u_{xx} + u_{yy}), \tag{6.22}$$

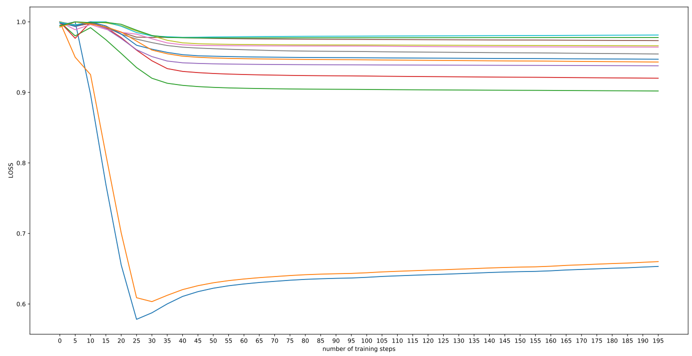


Fig. 8: Loss values for different validation problems evaluated each 5 training steps.

| | $L_\infty$ | | | | | $L_2$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $m$ | MWENO | WENO-DS (2d model) | ratio | WENO-DS (1d model) | ratio | MWENO | WENO-DS (2d model) | ratio | WENO-DS (1d model) | ratio |
| 2 | 0.009582 | 0.008436 | 1.14 | **0.008118** | 1.18 | 0.000836 | 0.000671 | 1.25 | **0.000660** | 1.27 |
| 3 | 0.055924 | **0.053288** | 1.05 | 0.053661 | 1.04 | 0.004178 | **0.003810** | 1.10 | 0.003938 | 1.06 |
| 4 | **0.102970** | 0.104485 | 0.99 | 0.105505 | 0.98 | 0.009584 | **0.009156** | 1.05 | 0.009359 | 1.02 |
| 5 | 0.191146 | **0.185335** | 1.03 | 0.189306 | 1.01 | 0.015311 | **0.014818** | 1.03 | 0.015023 | 1.02 |
| 6 | 0.154870 | **0.141532** | 1.09 | 0.142142 | 1.09 | 0.012903 | **0.012314** | 1.05 | 0.012444 | 1.04 |
| 7 | 0.268363 | **0.270085** | 0.99 | 0.271441 | 0.99 | 0.019981 | **0.019297** | 1.04 | 0.019738 | 1.01 |
| 8 | **0.298711** | 0.299791 | 1.00 | 0.301236 | 0.99 | 0.021872 | **0.021427** | 1.02 | 0.021806 | 1.00 |

Table 4: Comparison of $L_\infty$ and $L_2$ error of MWENO and WENO-DS methods for the solution of the porous medium equation with various parameter $m$, $d = 2$, $T = 2$. As 'ratio' we denote the error of the MWENO method divided by the error of WENO-DS (rounded to 2 decimal points).

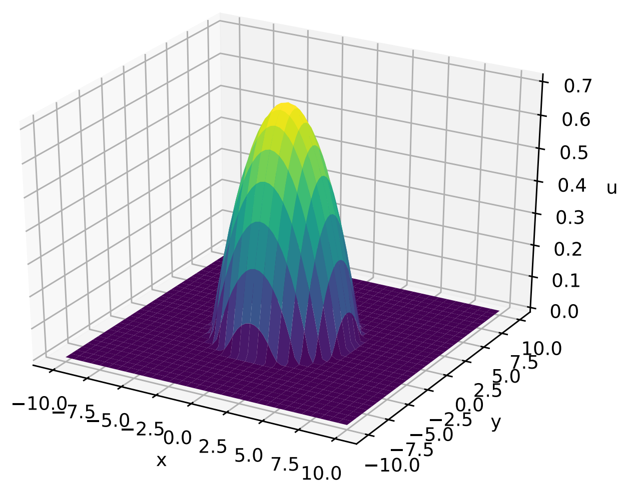



Fig. 9: Numerical solution of the porous medium equation with $d = 2$, $m = 2$ and $T = 2$. $64 \times 64$ cells.

with $\epsilon = 0.01$ and the flux functions

$$f_1(u) = \frac{u^2}{u^2 + (1-u)^2}, \qquad f_2(u) = f_1(u)\left(1 - 5(1-u)^2\right). \tag{6.23}$$

We solve equation (6.22) with the WENO-DS method trained on the one-dimensional Buckley-Leverett equation from subsection 6.2. We divide the computational domain $[-1.5, 1.5] \times [-1.5, 1.5]$ into $120 \times 120$ uniform cells and solve the equation with the initial condition

$$u(x, y, 0) = \begin{cases} 1, & x^2 + y^2 < 0.5, \\ 0, & \text{otherwise}. \end{cases} \tag{6.24}$$

The results at time $T = 0.5$ are presented in Figure 10 and agree with the results shown in [22]. With this example we demonstrate, that the method trained on one-dimensional data can be easily applied also in more dimensions and provides a high quality numerical solution to the equation with a nonlinear, degenerate diffusion.

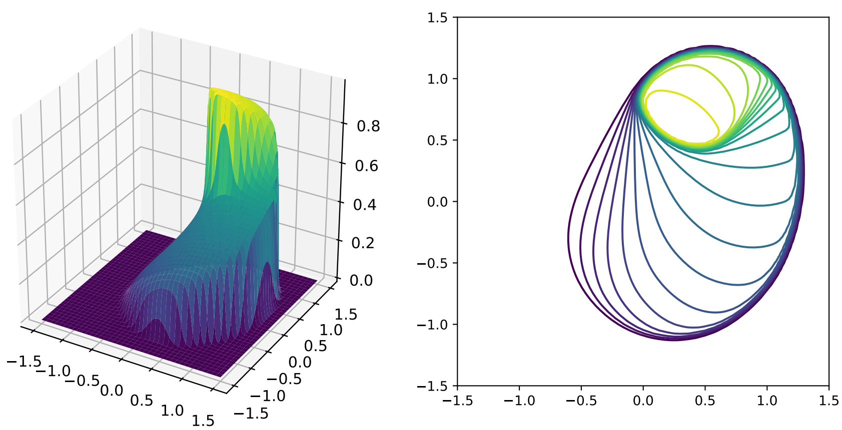



Fig. 10: Numerical solution of the two-dimensional Buckley-Leverett equation at $T = 0.5$. $120 \times 120$ cells.

**7. Conclusions.** In this paper, we developed a new modification of WENO scheme for nonlinear degenerate parabolic equations. Using deep learning techniques we improved the smoothness indicators of the original WENO method and applied our enhancement to the MWENO scheme. We preserved the sixth-order convergence and proved it theoretically. We presented an effective training procedure and extended it also to higher-dimensional space. In the one-dimensional and two-dimensional benchmark examples from the literature we demonstrate, that the WENO-DS method outperforms the standard WENO scheme in the challenging examples of nonlinear degenerate parabolic equations and remains sixth-order convergent in smooth regions.

## REFERENCES

[1] R. Abedian, *A new high-order weighted essentially non-oscillatory scheme for non-linear degenerate parabolic equations*, Numerical Methods for Partial Differential Equations, 37 (2021), pp. 1317–1343.

[2] R. Abedian, H. Adibi, and M. Dehghan, *A high-order weighted essentially non-oscillatory (WENO) finite difference scheme for nonlinear degenerate parabolic equations*, Comput. Phys. Commun., 184 (2013), pp. 1874–1888.

[3] H. W. Alt and S. Luckhaus, *Quasilinear elliptic-parabolic differential equations*, Math. Zeitschr., 183 (1983), pp. 311–341.

[4] T. Arbogast, C.-S. Huang, and X. Zhao, *Finite volume WENO schemes for nonlinear parabolic problems with degenerate diffusion on non-uniform meshes*, J. Comput. Phys., 399 (2019), p. 108921.

[5] D. Aregba-Driollet, R. Natalini, and S. Tang, *Explicit diffusive kinetic schemes for nonlinear degenerate parabolic systems*, Math. Comp., 73 (2004), pp. 63–94.

[6] D. G. Aronson, *The porous medium equation*, in Nonlinear diffusion problems, Springer, 1986, pp. 1–46.

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

[8] G. I. Barenblatt, *On self-similar motions of a compressible fluid in a porous medium*, Akad. Nauk SSSR. Prikl. Mat. Meh, 16 (1952), pp. 79–6.

[9] A. D. Beck, J. Zeifang, A. Schwarz, and D. Flad, *A neural network based shock detection and localization approach for discontinuous Galerkin methods*, J. Comput. Phys., 423 (2020).

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

[11] M. Bessemoulin-Chatard and F. Filbet, *A finite volume scheme for nonlinear degenerate parabolic equations*, SIAM J. Sci. Comput., 34 (2012), pp. B559–B583.

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

[13] F. Cavalli, G. Naldi, G. Puppo, and M. Semplice, *High-order relaxation schemes for nonlinear degenerate diffusion problems*, SIAM J. Numer. Anal., 45 (2007), pp. 2098–2119.

[14] A. Christlieb, W. Guo, Y. Jiang, et al., *Kernel based high order "explicit" unconditionally stable scheme for nonlinear degenerate advection-diffusion equations*, J. Sci. Comput., 82, 52 (2020).

[15] M. Hajipour and A. Malek, *High accurate NRK and MWENO scheme for nonlinear degenerate parabolic PDEs*, Appl. Math. Model., 36 (2012), pp. 4439–4451.

[16] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

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

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

[19] Y. Jiang, *High order finite difference multi-resolution WENO method for nonlinear degenerate parabolic equations*, J. Sci. Comput., 86, 16 (2021).

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

[21] T. Kossaczká, M. Ehrhardt, and M. Günther, *Enhanced fifth order WENO shock-capturing schemes with deep learning*, tech. report, IMACM Preprint 21/02, 2021.

[22] A. Kurganov and E. Tadmor, *New high-resolution central schemes for nonlinear conservation laws and convection–diffusion equations*, J. Comput. Phys., 160 (2000), pp. 241–282.

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

[24] X.-D. Liu, S. Osher, and T. Chan, *Weighted essentially non-oscillatory schemes*, J. Comput. Phys., 115 (1994), pp. 200–212.

[25] Y. Liu, C.-W. Shu, and M. Zhang, *High order finite difference WENO schemes for nonlinear degenerate parabolic equations*, SIAM J. Sci. Comput., 33 (2011), pp. 939–965.

[26] Y. Lu and W. Jäger, *On solutions to nonlinear reaction–diffusion–convection equations with degenerate diffusion*, J. Diff. Eqs., 170 (2001), pp. 1–21.

[27] F. Otto, *$L^1$-contraction and uniqueness for quasilinear elliptic–parabolic equations*, J. Diff. Eqs., 131 (1996), pp. 20–38.

[28] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., *Pytorch: An imperative style, high-performance deep learning library*, arXiv preprint arXiv:1912.01703, (2019).

[29] S. Rathan, R. Kumar, and A. D. Jagtap, *$L^1$-type smoothness indicators based WENO scheme for nonlinear degenerate parabolic equations*, Appl. Math. Comput., 375 (2020), p. 125112.

[30] J. Shi, C. Hu, and C.-W. Shu, *A technique of treating negative weights in WENO schemes*, J. Comput. Phys., 175 (2002), pp. 108–127.

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

[32] C.-W. Shu and S. Osher, *Efficient implementation of essentially non-oscillatory shock-capturing schemes, II*, in Upwind and High-Resolution Schemes, Springer, 1989, pp. 328–374.

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

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

[35] C. Van Duyn and L. Peletier, *Nonstationary filtration in partially saturated porous media*, Arch. Rat. Mech. Anal., 78 (1982), pp. 173–198.

[36] J. L. Vázquez, *The porous medium equation: mathematical theory*, Oxford University Press on Demand, 2007.

[37] Y. B. Zel'dovich and A. S. Kompaneets, *Towards a theory of heat conduction with thermal conductivity depending on the temperature*, Collection of papers dedicated to 70th birthday of Academician A. F. Ioffe, Izd. Akad. Nauk SSSR, Moscow, (1950), pp. 61–71.

[38] Q. Zhang and Z.-L. Wu, *Numerical simulation for porous medium equation by local discontinuous Galerkin finite element method*, J. Sci. Comput., 38 (2009), pp. 127–148.