Neural Network-Based Limiter with Transfer Learning

Rémi Abgrall[1] · Maria Han Veiga[2]

Abstract

Recent works have shown that neural networks are promising parameter-free limiters for a variety of numerical schemes (Morgan et al. in A machine learning approach for detecting shocks with high-order hydrodynamic methods. https://doi.org/10.2514/6.2020-2024; Ray et al. in J Comput Phys 367: 166–191. https://doi.org/10.1016/j.jcp.2018.04.029, 2018; Veiga et al. in European Conference on Computational Mechanics and VII European Conference on Computational Fluid Dynamics, vol. 1, pp. 2525–2550. ECCM. https://doi.org/10.5167/uzh-168538, 2018). Following this trend, we train a neural network to serve as a shock-indicator function using simulation data from a Runge-Kutta discontinuous Galerkin (RKDG) method and a modal high-order limiter (Krivodonova in J Comput Phys 226: 879–896. https://doi.org/10.1016/j.jcp.2007.05.011, 2007). With this methodology, we obtain one- and two-dimensional black-box shock-indicators which are then coupled to a standard limiter. Furthermore, we describe a strategy to transfer the shock-indicator to a residual distribution (RD) scheme without the need for a full training cycle and large dataset, by finding a mapping between the solution feature spaces from an RD scheme to an RKDG scheme, both in one- and two-dimensional problems, and on Cartesian and unstructured meshes. We report on the quality of the numerical solutions when using the neural network shock-indicator coupled to a limiter, comparing its performance to traditional limiters, for both RKDG and RD schemes.

✉ Maria Han Veiga
   mhanveig@umich.edu

   Rémi Abgrall
   remi.abgrall@math.uzh.ch

1   University of Zurich, Zurich, Switzerland

2   University of Michigan, Ann Arbor, USA

1 Introduction

When dealing with nonlinear conservation laws, it is well known that discontinuous solutions can emerge, even for smooth initial data [14]. The numerical approximation of the discontinuous solution will develop non-physical oscillations around the discontinuity, which in turn will negatively impact the accuracy of the numerical scheme. There exist many different stabilisation methods to control these oscillations, for example, through the addition of a viscous term (as denoted by the right-hand side of (1)) or use of limiters:

$$\frac{\partial}{\partial t}u + \nabla \cdot f(u) = \nabla \cdot (v(u)\nabla u). \tag{1}$$

Neural networks regained popularity in the past decade due to the computational tractability of the back-propagation algorithm, used for the learning of weights and biases in a deep neural network. Deep neural networks have been shown to generate robust models for classification in many areas of applications [23, 39] and theoretically, to generate universal classifiers and function approximators [34, 35, 38]. In the context of computational fluid dynamics, the idea of using artificial neural networks as troubled-cell indicators has been explored first in [31], motivated by the objective to find a universal troubled-cell indicator that can be used for general conservation laws. The authors show that this type of approach is promising, performing better than traditional, parameter-dependent limiters. Further studies following a similar idea have been conducted for other numerical schemes [4, 28].

Using limiters is a common way to perform shock capturing in a numerical scheme. In the context of the discontinuous Galerkin (DG) method, we can point out, among many, the high-order (HIO) limiter [6, 22], which does the limiting in a hierarchical manner. In particular, [22] relies on the modal representation of the numerical solution and it is formulated specifically for the modal DG method using Legendre polynomials as basis functions. For high-enough resolution, it does not clip the solution extrema.

Transfer learning is concerned with using a model built using a particular source data distribution on a different (but related) target data distribution [42]. One simple example is the task of spam filtering, where a model is used to discriminate between spam and non-spam emails—a model can be trained on the data of a particular user and adapted to be used on the data of a new user, who might receive significantly different emails. Following the same reasoning, we are interested in training shock-indicators that work in different numerical schemes, while using only (or the majority of the) training data from a particular numerical scheme. This can be useful, for example, in the case where there is access to a solver for which labeled data are easy to obtain.

In this work, we are interested in studying two main questions.

(i)   Is it possible to learn a data-driven shock-indicator function which requires minimal user input once trained?
(ii) Can this shock-indicator be used in different numerical schemes, leading to stabilisation methods which are agnostic to the underlying numerical scheme?

The paper is structured as follows: we start some preliminaries in Sect. 2, the methodology of training a data-driven shock-indicator and its integration with existing computational fluid dynamics (CFD) codes are presented in Sect. 3; in Sect. 4, the construction of the dataset is described (as well as the extension to two-dimensional problems) and Sect. 5 describes the transfer learning strategy. In Sects. 6 and 7, numerical results for

one-dimensional and two-dimensional problems, respectively, are shown. We conclude the paper with a discussion and outlook in Sect. 8 and finally, our conclusions are drawn in Sect. 9.

In the spirit of open and reproducible science, all the datasets, trained models and some solvers[1] are made available in a public repository [15].

2 Preliminaries: the Runge-Kutta Discontinuous Galerkin (RKDG) Scheme

Let us consider the classical modal RKDG method [10] using a simple scalar problem in one space dimension:

$$\begin{cases} \partial_t u + \partial_x f(u) = 0, & (x,t) \in \Omega \times [0, \infty], \\ u(t=0) = u_0, \\ u_{\partial\Omega} = g. \end{cases} \tag{2}$$

Let $\Omega \in \mathbb{R}$ be a regular domain which is discretised by N elements $K_p = [x_{p-1/2}, x_{p+1/2}]$ for $p = 1, \cdots, N$. Consider the local space $\mathcal{V}$ given by the set $\{\phi_i\}_{i=0}^n$ of one-dimensional Legendre polynomials with degree of at most n in x. For any element K, the numerical solution is written as

$$u^K(x,t) = \sum_{i=0}^n \hat{u}_i^K(t)\phi_i(x),$$

where the modal coefficient $\hat{u}_i^K(t)$ is obtained by the L^2 projection of the solution $u(x)$ in element K on the ith Legendre basis polynomial. The DG method is based on a weak form of (2), projecting it on the polynomial basis, followed by an integration by parts. We obtain the following semi-discrete formulation of the DG method as

$$\frac{\mathrm{d}\hat{u}_i^K}{\mathrm{d}t} + \left[\hat{f}(u^K(x,t))\phi_i(x)\right]_{x_{p-1/2}}^{x_{p+1/2}} - \int_{K_p} f(u^K(x,t))\partial_x\phi_i(x)\mathrm{d}x = 0, \quad i = 0, \cdots, n,$$

where we exploited the fact that Legendre polynomials form an orthonormal basis. Note that the surface term in the previous equation needs a Riemann solver to compute a continuous numerical flux at element boundaries, noted here $\hat{f}$. Once the spatial component has been discretised, we are left with an ordinary differential equation of the form

$$\frac{\mathrm{d}}{\mathrm{d}t}u = \mathcal{L}(u),$$

where $\mathcal{L}$ denotes the DG discretisation operator. Integration in time is performed using a strong stability preserving (SSP) RK method [17]. The time step has to fulfill a Courant–Friedrich–Lewy (CFL) condition to achieve the numerical stability, which for the RKDG scheme reads [11]

---

[1] Some of the solvers used are still under development and not publicly available.

$$\Delta t = \frac{C}{2n+1} \frac{\Delta x}{|v_{\max}|},$$

where n is the polynomial degree and C is a constant usually set to $C = 0.8$.

The method described above can compute solutions to (2) which are smooth (or contain weak shocks or other discontinuities). However, if the discontinuities are strong, the numerical solution will become significantly oscillatory. To surpass these difficulties, a slope limiter is used after each RK inner stage.

Now we describe the limiting procedure for a modal RKDG scheme as in [36], considering the minmod (MIN) limiter [18].

The coefficient u_1 for the linear term of the DG basis is modified as

$$\tilde{u}_1^K = \frac{1}{\sqrt{3}} \mathrm{minmod}\left(\sqrt{3}\hat{u}_1^K, \frac{1}{2}\left(\hat{u}_0 - \hat{u}_0^{K_1}\right), \frac{1}{2}\left(\hat{u}_0^{K_r} - \hat{u}_0^K\right)\right),$$

here $\tilde{u}_1^K$ is the new weight, K_1 denotes the left neighbouring cell and K_r the right neighbouring cell, and the MIN function is defined as

$$\mathrm{minmod}(a,b,c) = \begin{cases} s\min(|a|,|b|,|c|), & s = \mathrm{sign}(a) = \mathrm{sign}(b) = \mathrm{sign}(c), \\ 0, & \text{otherwise}. \end{cases}$$

In a system, each component of the conserved variables is limited separately. The $\sqrt{3}$-factors account for the scaling of the Legendre polynomial. If the limited weights are the same as the old weights (i.e., $\tilde{u}_1^K = \hat{u}_1^K$), the solution u^K is unlimited. Otherwise, the limited numerical solution becomes

$$\tilde{u}^K = \hat{u}_0^K + \tilde{u}_1^K \phi_1^K,$$

discarding further HIO information. Although this limiter effectively reduces over-shootings and oscillations, it can also be triggered at smooth extrema and lead to an unnecessary loss of HIO information. To avoid clipping of the solution at smooth extrema, the limiter presented above can be improved by a bounded version [11],

$$\mathrm{minmodB}(a,b,c) = \begin{cases} a, & \text{if } |a| \leqslant M(\Delta x^K)^2, \\ \mathrm{minmodB}(a,b,c), & \text{otherwise}. \end{cases}$$

Here M is a user-defined parameter that gives an estimate of the smoothness of the solution $u(x)$. It is to note that M can take a value in a large range of positive numbers, and that it is usually a global quantity fixed in the beginning of the numerical experiment. Thus, this can be a drawback if the solution has different smoothness properties across the domain (in space and time).

Conceptually, a limiter can be thought of having two sub-steps:

(i)   a shock-indicator procedure $\mathcal{C}$ that identifies a "troubled cell" (a cell which needs limiting);

(ii)   a reconstruction procedure Π that modifies the solution polynomial in those troubled cells into a reconstructed polynomial which is less oscillatory.

Then we can write a limiter as a function s that takes some local properties of the solution (let us denote this with the map $X(u(x))$), and returns a modified solution $\tilde{u}(x)$, which has some desired properties (e.g., non-oscillatory, maximum principle preserving, etc.):

$$s(u(x), X(u(x))) = \begin{cases} u(x), & \text{if } \mathcal{C}(X(u)) \text{ does not identify a troubled cell,} \\ \Pi(u(x)), & \text{if } \mathcal{C}(X(u)) \text{ identifies a troubled cell.} \end{cases}$$

For example, as detailed above, the MIN-based limiting procedure can be understood in these two sub-steps: $\mathcal{C}(X(u))$ checks if $\tilde{u}_1 = \hat{u}_1$, there is an equality, no troubled cell is detected, otherwise, the cell is troubled. Then if the cell is troubled, the reconstruction $\Pi(X(u(x))) = \hat{u}_0^K + \tilde{u}_1^K \phi_1^K$ is used. The map $X(u(x))$ encodes local properties of the solution around the interval K. In this case,

$$X(u(x)) = \left(\hat{u}_1^K, \frac{1}{2}\sqrt{3}(\hat{u}_0 - \hat{u}_0^{K_l}), \frac{1}{2}\sqrt{3}(\hat{u}_0^{K_r} - \hat{u}_0^K) \right).$$

In the following section, we describe how to construct a data-driven shock-indicator function $\mathcal{C}$ which is parameter-free once it has been trained.

3 Data-Driven Method

In the following section, we focus on the three main aspects of the proposed method.

– The setup of the learning algorithm to generate a shock-indicator function (Sect. 3.1).
– The integration of a neural network that serves as a shock-indicator with an existing CFD code (Sect. 3.2).
– A description of the performance measures used to validate the proposed data-driven method (Sect. 3.3).

3.1 Setup of the Learning Algorithm

In this section, we describe the details of the learning algorithm to generate a data-driven shock-indicator.

We wish to learn a map $\mathcal{C} : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X}$ denotes an arbitrary set containing examples that we wish to label with possible outcomes $\mathcal{Y}$. The task at hand is a binary classification (i.e., is there a troubled cell/is there a shock in this cell?), thus $\mathcal{C}$ will be a binary classifier and $\mathcal{Y} = \{0, 1\}$.

We choose $\mathcal{C}$ to be defined by the composition of a sequence of functions $g_1, g_2, \cdots, g_n$, yielding the function form

$$\mathcal{C}(x) = g_n(\cdots g_2(g_1(x))).$$

This is known as the multilayer perceptron (MLP) neural network (a type of deep neural network [16]). There are many different classifiers which can be used, but it has been shown that deep neural networks perform well on a variety of classification tasks, in particular when the classification plane is nonlinear.

Each function $g_i(w_i, b_i, h_i(\cdot))$ is parameterised by a matrix w_i, called the weights matrix, a vector b_i called the bias vector and an activation function $h_i(\cdot)$ which introduces the nonlinearity on the neural network.

These parameters are tuned through the minimisation of a loss function $\mathcal{L}(x)$, which measures how well the mapping $\mathcal{C}$ performs on a given dataset $\mathcal{D}$ of inputs and outputs, using backpropagation and gradient descent algorithms.

The gradient descent [37] is a first-order iterative optimization algorithm for finding a local minimum of a function (in this case, the loss function $\mathcal{L}(x)$), relying on the fact that for a small enough update η,

$$a_{n+1} = a_n - \eta \nabla \mathcal{L}(a_n),$$

then

$$\mathcal{L}(a_{n+1}) \leqslant \mathcal{L}(a_n)$$

for a differentiable and convex loss function $\mathcal{L}$. Because we cannot guarantee the convexity of our loss function, a stochastic gradient descent [7] like procedure is used. It is not the standard stochastic gradient descent because instead of using a global learning rate η, the Adam algorithm [21] is used, which chooses the learning rate adaptively for each parameter (in this case, for the weights and biases).

Furthermore, two different loss functions are considered:

– the standard cross-entropy

$$\mathcal{L}(\mathcal{D}) = -\frac{1}{N} \sum_i^N y_i \log(\hat{p}_i) + (1 - y_i) \log(1 - \hat{p}_i); \tag{3}$$

– the weighted cross-entropy

$$\mathcal{L}(\mathcal{D}) = -\frac{1}{N} \sum_i^N y_i \log(\hat{p}_i)\omega + (1 - y_i) \log(1 - \hat{p}_i)(1 - \omega). \tag{4}$$

We consider the weighted cross-entropy loss function as we expect that there will be a class imbalance in the dataset (both during the training phase and prediction phase). In particular, it is more likely to find cells which are in no need for stabilisation than ones which are in need for stabilisation. Furthermore, it is more desirable to overlimit than to miss a cell that needs limiting, as it might lead to unphysical results and potentially crash the code. To account for this class imbalance, it is common practice to use a weighted cost function, which increases the penalty of mislabeling a positive label [20]. The asymmetry in the loss function is added through the coefficient ω.

Finally, we specify the activation functions used, for the initial $n - 1$ hidden layers, rectified linear units (ReLU) are used:

$$h(x) = \max(0, x).$$

Although there exist more sophisticated activation functions, typically modifications to ReLU, e.g., leaky ReLU, parametric ReLU or randomised leaky ReLU, these require further parameter estimation, adding at least one more dependence to the saved model, and the empirical improvement on the performance is not extremely significant [43].

For the last layer (output layer), a sigmoid function is used:

$$h(x) = \frac{1}{1 + e^{-x}},$$

to attain a value that can be interpreted as a probability.

Lastly, the datasets (described in Sect. 4) are split into disjoint sets of training, validation and test sets.

The training phase is detailed in Algorithm 1. The hyper-parameters, such as batch-size (required the stochastic gradient descent type of minimization procedure), number of training epochs and general architecture of the neural network are fixed at run-time. We also add a early stopping criteria to minimize the risk of overfitting to the training data. This is triggered when the empirical generalisation error (measured through the loss on the validation set) increases for several training cycles [30].

Algorithm 1 Sketch of training algorithm
Data Training set $\mathcal{T}$, validation set $\mathcal{Q}$, hyper-parameters: number of epochs, architecture
Result $\mathcal{C}$ (trained classifier)
for number of epochs **do**
　　Compute training loss: $\mathcal{L}(\mathcal{T})$;
　　Use Adam algorithm to compute parameter update;
　　Compute validation loss: $\mathcal{L}(\mathcal{Q})$;
　　if early stopping criteria is met **then**
　　　　finish training;
　　end
end

3.1.1　Architecture

The architecture of the neural network is a hyperparameter as well. There is the general empirical wisdom that deep networks (more hidden layers) generalise better than shallow networks, even with the same amount of degrees of freedom [26].

There are, in addition, works which aim to establish lower bounds for the shape and size of deep neural networks, for example, [29], which establish lower bounds for the number of required non-zero weights and layers necessary for an MLP (using ReLU activation functions).

However, to use those estimates, assumptions about the classification function's regularity and desired approximation accuracy (in the L^2 norm) must be made, and those change the shape and size of the network dramatically, leading to very different lower bounds for the number of layers and number of neurons. In practice, we choose networks which are deeper and have small width, mainly with the intent to reduce the computational complexity of the model evaluation in mind. In Appendix A, the explicit architectures are detailed.

3.1.2　Directional Invariance

To introduce the feature invariance (see Fig. 1), the models tested use an aggregate way to estimate the label. For a given feature vector $\mathbf{x}$, several copies of this vector are generated $\mathbf{x}_1, \cdots, \mathbf{x}_n$ where permutations between the features are performed and their prediction is evaluated. The final label estimation is given by the majority label produced when evaluating the classifier on samples $\mathbf{x}, \mathbf{x}_1, \cdots, \mathbf{x}_n$. This ensures the response of the model does not depend on the orientation of the stencil.
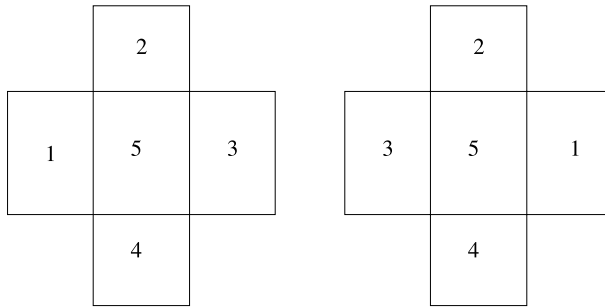
Fig. 1 Example of desired invariance, with respect to a reflection through the *y*-axis

3.2 Integration of the Method on a CFD Code

Assuming that an MLP has been trained, then the model can be fully specified by the composition of the functions g_i. Furthermore, each g_i can be fully characterized by the following information:

$$w_i, b_i, h_i(\cdot),$$

where w_i denotes the weights matrix, b_i the bias vector, and h_i the activation function.

There are two necessary steps to integrate a trained neural network with an existing code.

(i)   Generation of features: given the local solution *u*, generate the feature quantities $X(u)$.

(ii) Prediction routine: given the features $X(u)$, the classifier $\mathcal{C}$ is evaluated on the given feature vector. Once the neural network has been trained offline, the weights and biases can be loaded onto a CFD code. What remains to be implemented are the activation functions for the hidden layers and the activation function for the output layer to evaluate $\mathcal{C}$ at some given input.

Now that the neural network-based shock-indicator has been described, the full limiting strategy is detailed in Algorithm 2. We will refer to this procedure as the neural network limiter. Furthermore, the limiter which we use in conjunction with the neural network shock-indicator is the HIO limiter.

Algorithm 2 The procedure that we refer to as the neural network limiter that comprises of a classification step and a subsequent call to a generic limiter or stabilisation method

Data solution at cell i, u_i
Result stabilised solution $\tilde{u}_i$
X = generateFeatures(u_i);
label = $\mathcal{C}(X)$;
if label = 1 **then**
    $\tilde{u}_i = \Pi(u_i)$;
else
    $\tilde{u}_i = u_i$;
end

For systems, each variable is limited independently as in Sect. 2.

3.3 Measuring Performance of Model

We use two sets of performance measures, namely

(i)   label prediction measures;
(ii)   the L^1-norm of the numerical solution.

For the first set of measures, we can use typical metrics used in computational statistics and machine learning communities:

$$\frac{tp + tn}{tp + tn + fp + fn} \quad \text{(accuracy)},$$

$$\frac{tp}{tp + fn} \quad \text{(recall)},$$

$$\frac{tp}{tp + fp} \quad \text{(precision)},$$

where tp is the number of correctly predicted positive labels, tn the number of correctly predicted negative labels, fp the number of incorrectly predicted positive labels, and fn the number of incorrectly predicted negative labels. We note that it is important to consider recall and precision because the distribution of labels is expected to be imbalanced. Judging the performance of a classifier only through the accuracy could be misleading, as a classifier can have a high accuracy by always predicting the majority label.

For the second performance measure, we consider the L^1-norm because it is the relevant measure for a CFD code and we can study the effect of this method on the error of the numerical solution.

4 Dataset

The dataset is an integral part of data-driven studies. It contains the data for which we want to learn a mapping for. The task is to learn a function which indicates whether a discontinuity is present in the solution or not. The dataset is the set containing N samples $\{X_i, y_i\}_{i=1}^N$, where X_i denotes some local properties of the solution $u(x)$ (features) and y_i (labels) indicates the existence (or not) of a discontinuity.

For the one-dimensional case, the dataset is generated by performing many runs on a one-dimensional DG code solving the advection equation for different initial conditions, orders and mesh sizes (see Table 1), and the labels are obtained by running the HIO

Table 1 Runs used to generate the one-dimensional dataset

Initial condition	$\rho = \sin(2\pi x),$	$\begin{cases} 8, & 0.25 \leqslant x \leqslant 0.75, \\ 2, & \text{otherwise}, \end{cases}$	$\begin{cases} \frac{1}{2}\sin(2\pi x), & x \leqslant 0.3, \\ 0, & \text{otherwise}, \end{cases}$	$x \in [0,1]$
Advection speed	$-1, 1$			
Mesh size	8, 16, 32, 64, 128			
Order	2, 3			

Fig. 2 Example of a dataset entry. The dashed line denotes the limited solution, whereas the full line denotes the unlimited solution. Furthermore, the crosses "×" denote cells which are flagged as troubled cells
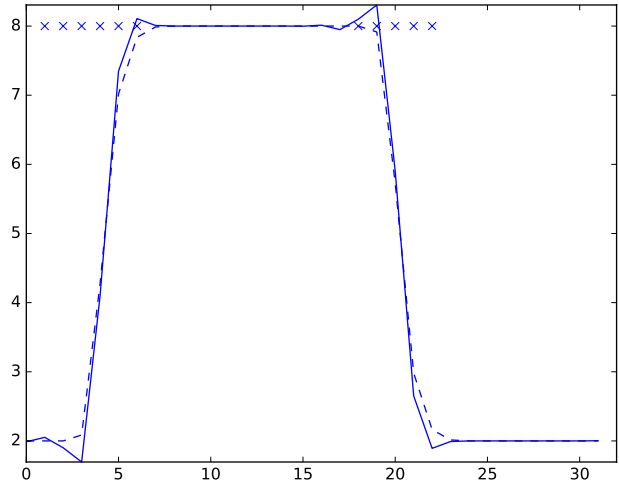
limiter [22]. Because the HIO limiter is sometimes triggered (and modifies the solution) even when the solution is smooth, we introduced a further threshold to generate a positive label: if the limited solution deviates from the unlimited solution by more than $\zeta = 1\%$, then we get a positive label.

A visual representation of an entry on the dataset can be seen in Fig. 2, where each discrete point in x, the associated solution $u(x)$ and its local properties generate a data point in the feature space, and the corresponding label is denoted by the presence of a cross (if the cell is troubled) or its absence (if the cell is not troubled). Empirically, we noted that the cells which are flagged by the HIO limiter are similar to a well-tuned TVD limiter with the chosen threshold ζ.

4.1 Features

The features X are the different quantities used to describe the local solution u. For the sake of generalisation, we choose features which are readily available in different numerical methods, such as the cell mean value, values at interface, divided differences between neighbours (see Table 2 for the complete description of features).

Furthermore, to introduce some magnitude invariance, we normalise physical values (such as averages or pointwise values):

$$u_{\text{normal}}(u_*) = \frac{u_* - u_{\min}}{\mid u_{\max} \mid + \mid u_{\min} \mid} - \frac{u_{\max} - u_*}{\mid u_{\max} \mid + \mid u_{\min} \mid}, \tag{5}$$

where $u_{\max} = \max(\bar{u}_i, \bar{u}_{i+1}, \bar{u}_{i-1})$ and $u_{\min} = \min(\bar{u}_i, \bar{u}_{i+1}, \bar{u}_{i-1})$ (taken only over averages). This is a very important step, as a naive normalisation can lead to a non-informative representation of the feature vector. The reason why this normalisation was chosen was because it made intuitive sense to measure the deviation of the degrees of freedom with respect to the maximum and minimum of the local patch.

Table 2 Features table for the one-dimensional problem

ID	Feature name	Description
1	h	Cell width
2	$\bar{u}_i$	Average value of solution at cell i
3	$\bar{u}_{i+1}$	Average value of solution at cell $i + 1$
4	$\bar{u}_{i-1}$	Average value of solution at cell $i - 1$
5	$u^+_{i-\frac{1}{2}}$	Value of solution at interface $i - \frac{1}{2}$ reconstructed in cell i
6	$u^-_{i+\frac{1}{2}}$	Value of solution at interface $i + \frac{1}{2}$ reconstructed in cell i
7	$u^-_{i-\frac{1}{2}}$	Value of solution at interface $i - \frac{1}{2}$ reconstructed at cell $i - 1$
8	$u^+_{i+\frac{1}{2}}$	Value of solution at interface $i + 1$ reconstructed at cell $i + 1$
9	$\mathrm{d}u_{i+1}$	Undivided difference between $\bar{u}_i$ and $\bar{u}_{i+1}$
10	$\mathrm{d}u_{i-1}$	Undivided difference between $\bar{u}_i$ and $\bar{u}_{i-1}$
11	$\mathrm{d}u_i$	Undivided difference between $\bar{u}_{i+1}$ and $\bar{u}_{i-1}$, divided by 2

4.2 Extension to Two-Dimensional Problems

The extension to two-dimensional problems is done in a straightforward manner, using the initial conditions detailed in Table 3 to generate the dataset.

To obtain the labels, the HIO limiter is used again with a threshold $\zeta = 0.25\%$ for maximal difference between the limited and unlimited solutions. This is necessary as although the HIO limiter might not degrade smooth solutions, it is still triggered (Fig. 3).

Furthermore, the features used are shown on Table 4 and the features are normalised in the same manner as in the one-dimensional case.

5 Transfer Learning

With the objective to generate a shock-indicator function which does not depend on the underlying numerical scheme, we are interested in using a shock-indicator function trained with simulation data from a particular solver (numerical scheme and mesh type) and testing its performance in a different solver (in particular, for a different numerical scheme

Table 3 Runs used to generate the two-dimensional dataset

Initial condition	$\begin{cases} 8, & 0.25 \leqslant |\mathbf{x}| \leqslant 0.75, \\ 2, & \text{otherwise,} \end{cases}$
	$\exp(-10((x_1 - 0.5)^2 + (x_2 - 0.5)^2)^2), \quad \mathbf{x} \in [0, 1]^2$
Advection speed	$(-1, -1), (1, 1), (1, 0), (0, 1)$
Mesh size	16, 32, 64, 128
Order	2, 3

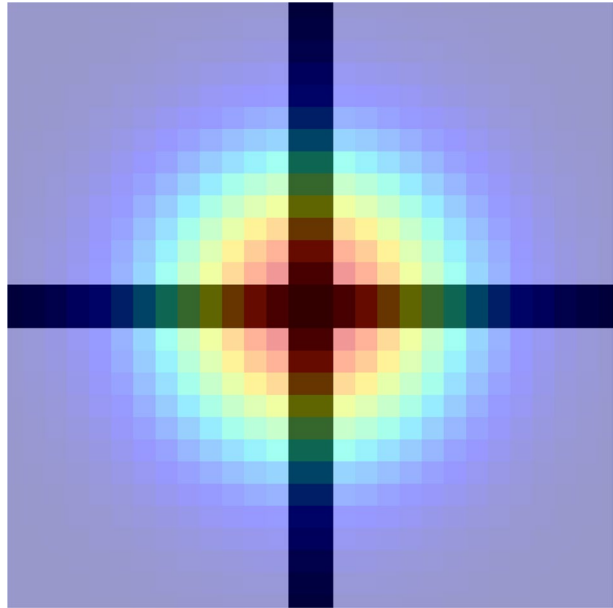**Fig. 3** The HIO limiter wrongly getting activated in a smooth solution

Table 4 Features table for the two-dimensional problem

ID	Feature name	Description
1	Δx	Cell x-width
2	Δy	Cell y-width
3	$\bar{u}_{i,j}$	Average value of solution at cell i
4	$\bar{u}_{i+1,j}$	Average value of solution at cell $i+1, j$
5	$\bar{u}_{i-1,j}$	Average value of solution at cell $i-1, j$
6	$\bar{u}_{i,j+1}$	Average value of solution at cell $i, j+1$
7	$\bar{u}_{i,j-1}$	Average value of solution at cell $i, j-1$
8	$u^{+}_{i-\frac{1}{2},j}$	Value of solution at interface $i-\frac{1}{2}$ reconstructed in cell i,j
9	$u^{-}_{i+\frac{1}{2},j}$	Value of solution at interface $i+\frac{1}{2}$ reconstructed in cell i,j
10	$u^{+}_{i-\frac{1}{2},j}$	Value of solution at interface $i-\frac{1}{2}$ reconstructed at cell $i-1,j$
11	$u^{-}_{i+\frac{1}{2},j}$	Value of solution at interface $i+\frac{1}{2}$ reconstructed at cell $i+1,j$
12	$u^{+}_{i,j-\frac{1}{2}}$	Value of solution at interface $j-\frac{1}{2}$ reconstructed in cell i,j
13	$u^{-}_{i,j+\frac{1}{2}}$	Value of solution at interface $i+\frac{1}{2}$ reconstructed in cell i,j
14	$u^{+}_{i,j-\frac{1}{2}}$	Value of solution at interface $i-\frac{1}{2}$ reconstructed at cell $i,j-1$
15	$u^{-}_{i,j+\frac{1}{2}}$	Value of solution at interface $i+\frac{1}{2}$ reconstructed at cell $i,j+1$
16	$\mathrm{d}u_{i+1,j}$	Divided difference between $\bar{u}_{i,j}$ and $\bar{u}_{i+1,j}$
17	$\mathrm{d}u_{i-1,j}$	Divided difference between $\bar{u}_{i,j}$ and $\bar{u}_{i-1,j}$
18	$\mathrm{d}u_{i,j_x}$	Divided difference between $\bar{u}_{i+1,j}$ and $\bar{u}_{i-1,j}$, divided by 2
19	$\mathrm{d}u_{i,j+1}$	Divided difference between $\bar{u}_{i,j}$ and $\bar{u}_{i,j+1}$
20	$\mathrm{d}u_{i,j-1}$	Divided difference between $\bar{u}_{i,j}$ and $\bar{u}_{i,j-1}$
21	$\mathrm{d}u_{i,j_y}$	Divided difference between $\bar{u}_{i,j+1}$ and $\bar{u}_{i,j-1}$, divided by 2
22	$u_{\max}$	Maximum value between the averages in the considered patch
23	$u_{\min}$	Minimum value between the averages in the considered patch

and mesh type). Furthermore, we want to explore different strategies which can be used to perform the transfer learning.

We describe how to adapt a neural network shock-indicator that has been trained on data from a modal DG scheme on a Cartesian mesh to a neural network shock-indicator that works on a residual distribution (RD) scheme ([1, 2, 32, 33] for a brief introduction) on a Cartesian mesh and an unstructured triangular mesh.

The motivations to study this type of problem are, first, for some numerical schemes, there are limiters which are designed to be parameter free which rely on a particular feature of the underlying numerical scheme. It would be desirable if limiters designed for a particular numerical scheme could be generalised to be used for different numerical schemes. Second, there might be a particular numerical solver for which it is easier to generate labeled data.

Traditional supervised machine learning operates under the assumption that training and testing data are taken from the same input space and the same data distribution. However, this assumption does not always hold. Transfer learning aims to produce an effective model for a target task with limited or no labeled training data, using knowledge from a different, but related problem.

Formally, the problem of transfer learning can be written as let $\mathcal{X}_s$ be the source instance space. In this space, each instance $x_s \in \mathcal{X}_s$ is represented by a feature vector $\mathbf{x}_s \in \mathcal{X}_s$, where $\mathcal{X}_s$ denotes the source feature space. Let $\mathcal{X}_t$ be the target instance space and similarly, $x_t \in \mathcal{X}_t$ is represented by a feature vector $\mathbf{x}_t \in \mathcal{X}_t$, where $\mathcal{X}_t$ denotes the target feature space. In the case of heterogeneous transfer learning, we have that $\mathcal{X}_t \neq \mathcal{X}_s$.

Most heterogeneous transfer learning solutions involve a transformation of the feature spaces: a symmetric transformation takes both feature spaces $\mathcal{X}_t, \mathcal{X}_s$ and learns a feature transformation to project each vector onto a common space for adaptation purposes $\mathcal{X}_c$ [42], whereas an asymmetric transformation transforms the source feature space to align with the target feature space (or vice-versa). This approach is appropriate when the source and target have the same class label space and one can transform between $\mathcal{X}_t$ and $\mathcal{X}_s$.

In this case, the source dataset is generated as detailed in Sect. 4 and the target datasets are described below (Sect. 5.2.1). The end goal is to understand to what extent a shock-indicator designed for one specific class of numerical schemes (in this work, modal DG on Cartesian meshes), can be effective on a different numerical scheme (RD, both for structured and unstructured meshes).

5.1 One-Dimensional Case

We use a trained neural network shock-indicator and we integrate it with a one-dimensional RD code. Note, we must find a projection of the solution computed with the RD scheme to the feature space of the considered neural network. The one-dimensional case is not difficult, in particular, because when designing the feature space for the one-dimensional limiter, quantities which are readily available in most numerical schemes were chosen.

5.2 Two-Dimensional Case

The two-dimensional case is not as simple, as the shock-indicator is trained using simulation data from Cartesian meshes, and the target problems are defined not only on Cartesian meshes but also on triangular, unstructured meshes.

To this end, we must find a common feature space between these two methods. In particular, we test the two simple strategies:

– mapping to feature space of the Cartesian model,
– followed by a retraining phase.

Mapping to unstructured mesh While the mapping to structured square meshes is quite straightforward, for unstructured meshes, we project the numerical solution of the RD scheme, defined on triangular elements, to the feature space of the classifier (defined on a Cartesian mesh, as defined in Table 4). The feature transformation can be found in Table 5.

Table 5 Features transformation

ID	Feature	Remapped	Description
1	Δx	$\sqrt{a}$	Square root of element area
2	Δy	$\sqrt{a}$	Square root of element area
3	$\bar{u}_{i,j}$	$\bar{u}_i$	Average value of solution at triangle i
4	$\bar{u}_{i+1,j}$	$\bar{u}_{i,1}$	Average value of solution at neighbouring cell 1 of cell i
5	$\bar{u}_{i-1,j}$	$\bar{u}_{i,2}$	Average value of solution at neighbouring cell 2 of cell i
6	$\bar{u}_{i,j+1}$	$\bar{u}_{i,3}$	Average value of solution at neighbouring cell 3 of cell i
7	$\bar{u}_{i,j-1}$	$\frac{1}{4}\left(\bar{u}_{i,j} + \sum_{k}\bar{u}_k\right)$	Average value of solution at patch
8	$u^+_{i-\frac{1}{2},j}$	$u(x_{e_1})$	Value of solution at mid-point of edge shared between triangle i and neighbour 1
9	$u^-_{i+\frac{1}{2},j}$	$u(x_{e_2})$	Value of solution at mid-point of edge shared between triangle i and neighbour 2
10	$u^+_{i-\frac{1}{2},j}$	$u(x_{e_3})$	Value of solution at mid-point of edge shared between triangle i and neighbour 3
11	$u^-_{i+\frac{1}{2},j}$	$\bar{u}_i$	Average of solution at triangle i
12	$u^+_{i,j-\frac{1}{2}}$	$u(x_{e_1})$	Value of solution at mid-point of edge shared between triangle i and neighbour 1
13	$u^-_{i,j+\frac{1}{2}}$	$u(x_{e_2})$	Value of solution at mid-point of edge shared between triangle i and neighbour 2
14	$u^+_{i,j-\frac{1}{2}}$	$u(x_{e_3})$	Value of solution at mid-point of edge shared between triangle i and neighbour 3
15	$u^-_{i,j+\frac{1}{2}}$	$\bar{u}_i$	Average of solution in triangle i
16	$du_{i+1,j}$	$\bar{u}_i - \bar{u}_{i,1}$	Undivided difference between $\bar{u}_i$ and $\bar{u}_{i,1}$
17	$du_{i-1,j}$	$\bar{u}_i - \bar{u}_{i,2}$	Undivided difference between $\bar{u}_i$ and $\bar{u}_{i,1}$
18	du_i	$\frac{1}{2}(\bar{u}_{i,1} - \bar{u}_{i,2})$	Undivided difference between $\bar{u}_{i,1}$ and $\bar{u}_{i,2}$, divided by 2
19	$du_{i,j+1}$	$\bar{u}_i - \bar{u}_{i,3}$	Undivided difference between $\bar{u}_i$ and $\bar{u}_{i,3}$
20	$du_{i,j-1}$	$\bar{u}_i - \bar{u}_i$	Placeholder undivided difference
21	du_j	$\frac{1}{2}(\bar{u}_{i,1} - \bar{u}_{i,3})$	Undivided difference between $\bar{u}_{i,1}$ and $\bar{u}_{i,3}$, divided by 2
22	u_{max}	u_{max}	Maximum value between the averages in the considered patch
23	u_{min}	u_{min}	Minimum value between the averages in the considered patch

Again, as mentioned in Sect. 3.1.2, to enforce directional invariance, we perform the aggregated prediction by permuting the stencil for the triangular meshes. This seems to help with the asymmetry introduced by replacing a side of the rectangle (in the feature space defined for the RKDG data) with the mean of the solution patch.

Retraining A model which was trained with the data generated by the DG scheme is loaded and retrained using the dataset as described in Sect. 5.2.1.

To avoid the phenomena of catastrophic forgetting of neural networks [16], which describes the lack of ability to learn different tasks in a sequence, in the retraining phase, a hybrid dataset containing elements from the target and source dataset is used, with a parameter λ which determines the ratio to be taken from each dataset.

The retraining algorithm is detailed in Algorithm 3.

Algorithm 3 Retraining phase

Data trained classifier $\mathcal{C}$, source dataset $\mathcal{S}$, target dataset $\mathcal{T}$, dataset ratio λ

Result transferred classifier $\mathcal{C}_t$

Remap features from target dataset to source dataset feature space: $\mathcal{T}_r = \mathrm{P}(\mathcal{T})$;

Form training dataset $\mathcal{D}_t$ that contains training examples of $\mathcal{S}$ and $\mathcal{T}_r$, with a ratio specified by λ;

Form validation dataset $\mathcal{D}_v$ that contains validation examples from target dataset: $\mathcal{T}_r$;

for number of epochs **do**

    Compute training loss: $\mathcal{L}(\mathcal{D})$;

    Use Adam algorithm to compute parameter update;

    Compute validation loss: $\mathcal{L}(\mathcal{D}_v)$;

    if early stopping criteria is met **then**

   finish training;

    end

end

5.2.1 RD Dataset

To generate the dataset, small meshes are constructed (see Fig. 4). We represent the solution in each control volume as a linear combination of the polynomial basis. Because there is no straightforward way to generate a labeled dataset through simulations, we impose continuous and discontinuous functions, randomly varying the orientation of the discontinuity.

Fig. 4 Example meshes
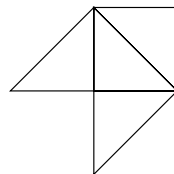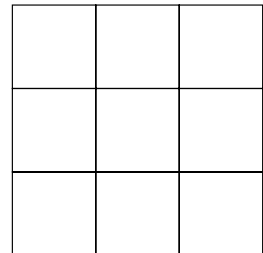
(a) Triangular mesh   **(b)** Cartesian mesh

Using this method, we can generate a large dataset of examples which is close to the task at hand.

6 One-Dimensional Numerical Experiments

This section is split in three parts. First, we show the performance of several trained neural networks on an unseen validation set by measuring the accuracy, recall and precision. In the second part, we choose a model that performed well and we integrate it with a CFD code. The model runs as a black-box (denoted as NN) limiter and we compare its performance to the MIN limiter and the hierarchical HIO limiter through the L^1 error norm. We perform some tests for the linear advection equation and the Euler system of equations. The initial conditions are chosen as different from the ones used for the training. Finally, the transferred limiter is tested in the context of a one-dimensional RD scheme.

6.1 Detection Rate

We measure the performance of several models (their architectures are given in Appendix A) on an unseen test set. Their performances are detailed in Table 6). Going forward, we select Model 4 as it performs well and the resulting size of the weights matrices per layer is significantly smaller than Model 3. It is debatable whether the differences between Models 3, 4 and 5 are statistically significant. Furthermore, surprisingly, we note that the weighted loss function did not improve the chosen performance metrics.

6.2 Numerical Validation

6.2.1 Linear Advection

Consider a linear advection equation with $a \in \mathbb{R}$:

$$\frac{\partial}{\partial t}u + a\frac{\partial}{\partial x}u = 0 \tag{6}$$

and periodic boundary conditions.

Case of a Gaussian pulse We consider the following initial condition:

Table 6 Performances

Model	Description	Accuracy/%	Recall/%	Precision/%
Random	Randomly guessing	50.00	θ^2	50.00
Model 1	2 hidden layers (HL)	77.96	69.17	24.13
Model 2	3 HL	95.52	80.66	84.19
Model 3	4 HL	95.15	79.26	82.67
Model 4	5 HL	95.14	81.71	81.01
Model 5	5 HL + weighted loss ($\omega = 5$)	95.67	79.05	86.39

θ denotes the estimate of the ratio between positive and negative labels, which could be measured empirically through the training data. Empirically, we have $\hat{\theta} = 0.1$.

$$u_0(x) = 1 + 3\exp(-100(x - 0.5)^2), \quad (x, t) \in [0, 1] \times \mathbb{R}^+ \tag{7}$$

with the advection velocity $a = 1$.

The convergence is shown in Table 7 after one full crossing for orders 2 and 3. In Fig. 5, we show how the maxima is clipped using different limiters for grid sizes of $N = 40$ and $N = 80$. We note that the MIN limiter clips the maximum value of the solution, as expected. The HIO limiter behaves as the MIN limiter for the second-order case, but for the third order, it does not limit the solution. The neural network-based (NN) limiter also limits the solution at the second order and $N = 40$, but it improves as N increases. The NN limiter seems to be slightly less diffusive than the HIO limiter for the second-order case; furthermore, the performance does not depend (as much as the HIO limiter) on the order of the method.

Case of a smooth pulse and square hat The following initial conditions contain a smooth Gaussian pulse and a hat function:

$$u_0(x) = \begin{cases} 2, & |x - 0.7| \leqslant 0.1, \\ 1 + \exp\left(-\frac{(x-0.25)^2}{2\times 0.05^2}\right), & \text{otherwise,} \end{cases} \quad (x, t) \in [0, 1] \times \mathbb{R}^+, \tag{8}$$

again with the advection velocity $a = 1$.

The convergence is shown in Table 8 after one full crossing for orders 2 and 3. Furthermore, in Fig. 6, we show how the different limiters perform, for a grid size of $N = 40$ and $N = 80$ and at the second and the third order. Namely, we notice that at the second order and $N = 40$, the solution given by the NN limiter is similar to the unlimited one, with less undershoots, however, increasing the resolution to $N = 80$ seems to degrade the performance. Overall, in this case, the NN limiter seems able to get rid of most under and overshoots associated with discontinuities, while having a similar performance to the HIO limiter.

Table 7 L^1 error for one crossing of the Gaussian pulse (7) using different limiters

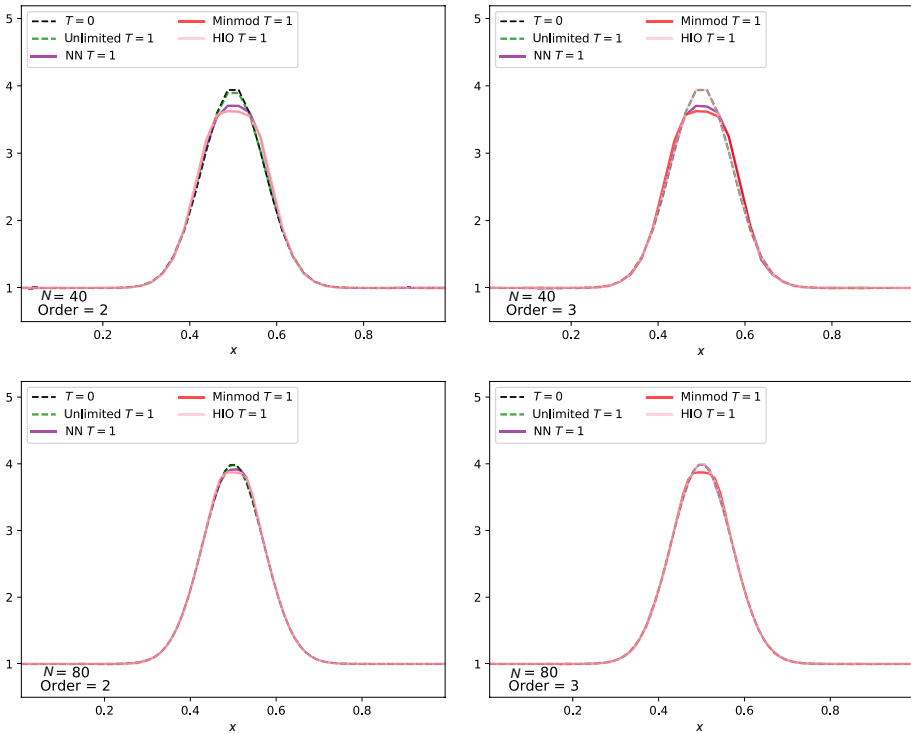| N	No limiter		MIN		HIO		NN	
(a) Order 2								
16^2	6.74E−02	0	1.26E−01	0	1.23E−01	0	6.79E−02	0
32^2	2.00E−02	0.5	4.81E−02	0.4	4.14E−02	0.4	1.97E−02	0.5
64^2	3.45E−03	1.7	1.31E−02	1.4	9.11E−03	1.6	3.45E−03	1.8
128^2	5.42E−04	2.1	2.75E−03	1.6	1.59E−03	1.9	5.41E−04	2.1
(b) Order 3								
16^2	6.92E−03	0	1.26E−01	0	1.17E−01	0	1.88E−02	0
32^2	4.58E−04	0.9	4.80E−02	0.4	2.92E−02	0.4	7.91E−04	0.7
64^2	7.93E−05	3.9	1.31E−02	1.4	6.85E−03	2.0	1.08E−04	4.6
128^2	4.36E−05	3.2	2.75E−03	1.6	1.20E−03	2.0	5.15E−05	3.7

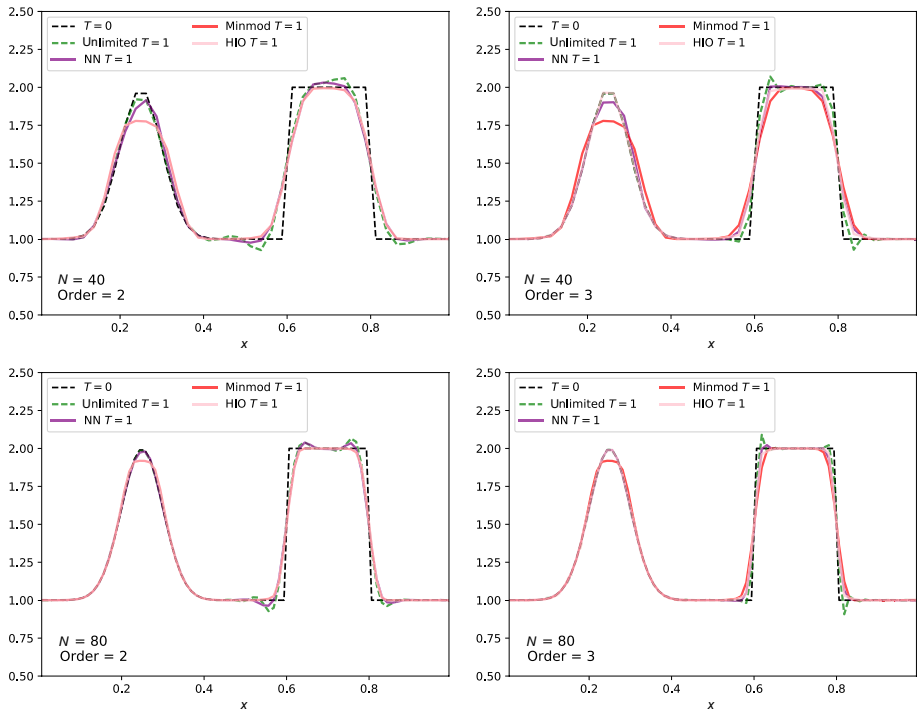**Fig. 5** Maxima clipping of the Gaussian pulse (7) after one full crossing for approximation order of 2 (left) and 3 (right) for grid size $N = 40$ and $N = 80$

6.2.2 One-Dimensional Euler Equation

Now we consider the one-dimensional Euler equations, which describe the behavior of an inviscid flow. This system of equations describes the evolution of a density ρ, a velocity v, a pressure p and total energy E,

Table 8 L^1 error for one crossing of the Gaussian pulse and hat function (8) using different limiters

N	No limiter		MIN		HIO		NN	
(a) Order 2								
16^2	3.04E−02	0	5.37E−02	0	4.94E−02	0	3.20E−02	0
32^2	2.13E−02	0.6	2.63E−02	0.5	2.63E−02	0.5	2.12E−02	0.6
64^2	1.12E−02	0.5	1.16E−02	1.0	1.01E−02	0.9	1.11E−03	0.6
128^2	8.33E−03	0.7	7.74E−03	1.1	7.74E−03	1.1	8.24E−03	0.8
(b) Order 3								
16^2	1.17E−02	0	5.37E−02	0	5.27E−02	0	2.09E−02	0
32^2	7.61E−03	0.8	2.63E−02	0.5	1.63E−02	0.5	1.05E−02	0.7
64^2	4.63E−03	0.6	1.16E−02	1.0	7.58E−03	1.7	5.73E−03	1.0
128^2	2.91E−03	0.7	7.74E−03	1.1	4.21E−03	1.4	3.37E−03	0.9

Fig. 6 Maxima clipping of the Gaussian and hat pulses (8) after one full crossing for approximation order of 2 (left) and 3 (right) for grid size $N = 40$ and $N = 80$

$$\frac{\partial}{\partial t}\rho + \frac{\partial}{\partial x}(\rho v) = 0, \tag{9}$$

$$\frac{\partial}{\partial t}(\rho v) + \frac{\partial}{\partial x}(\rho v^2 + p) = 0, \tag{10}$$

$$\frac{\partial}{\partial t}E + \frac{\partial}{\partial x}((E + p)v) = 0. \tag{11}$$

The system is closed with equation of state for an ideal gas

$$\rho e = \frac{p}{\gamma - 1},$$

where $e = E - \frac{1}{2}\rho v^2$ is the internal energy.

The NN limiter is applied sequentially for each variable.

Case of Sod shock tube We consider the standard Sod shock tube test, given by the initial conditions:

$$(\rho, v, p)(x, 0) = \begin{cases} (1.0, 0, 1.0), & 0 < x \leqslant 0.5, \\ (0.125, 0, 0.1), & 0.5 < x < 1.0, \end{cases} \quad (x, t) \in [0, 1] \times [0, 0.24], \tag{12}$$

and $\gamma = 1.4$ and gradient-free boundary conditions.
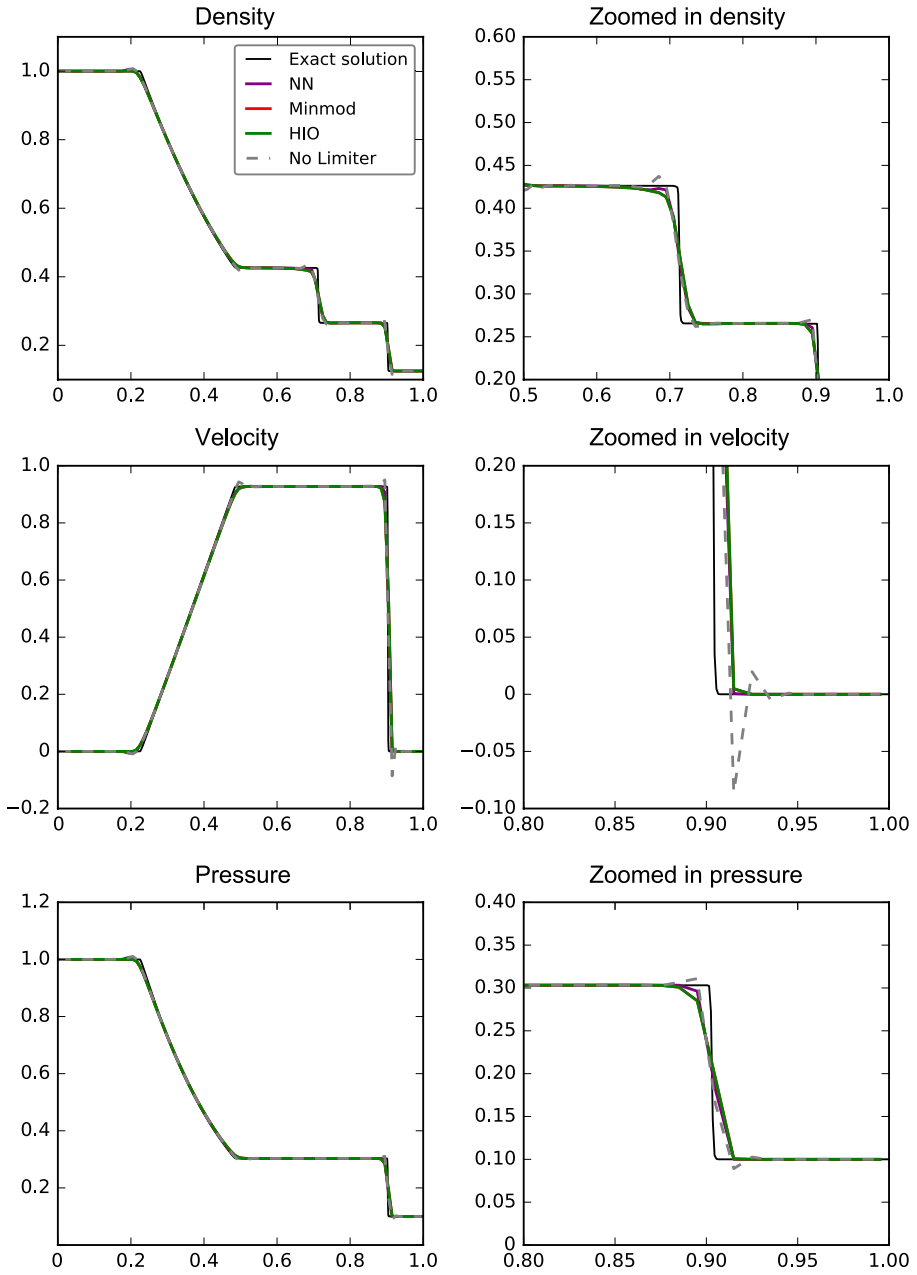
Fig. 7 Density, velocity, and pressure fields for the Sod shock tube (12) at $T = 0.24$ for grid size $N = 100$ and approximation order 2
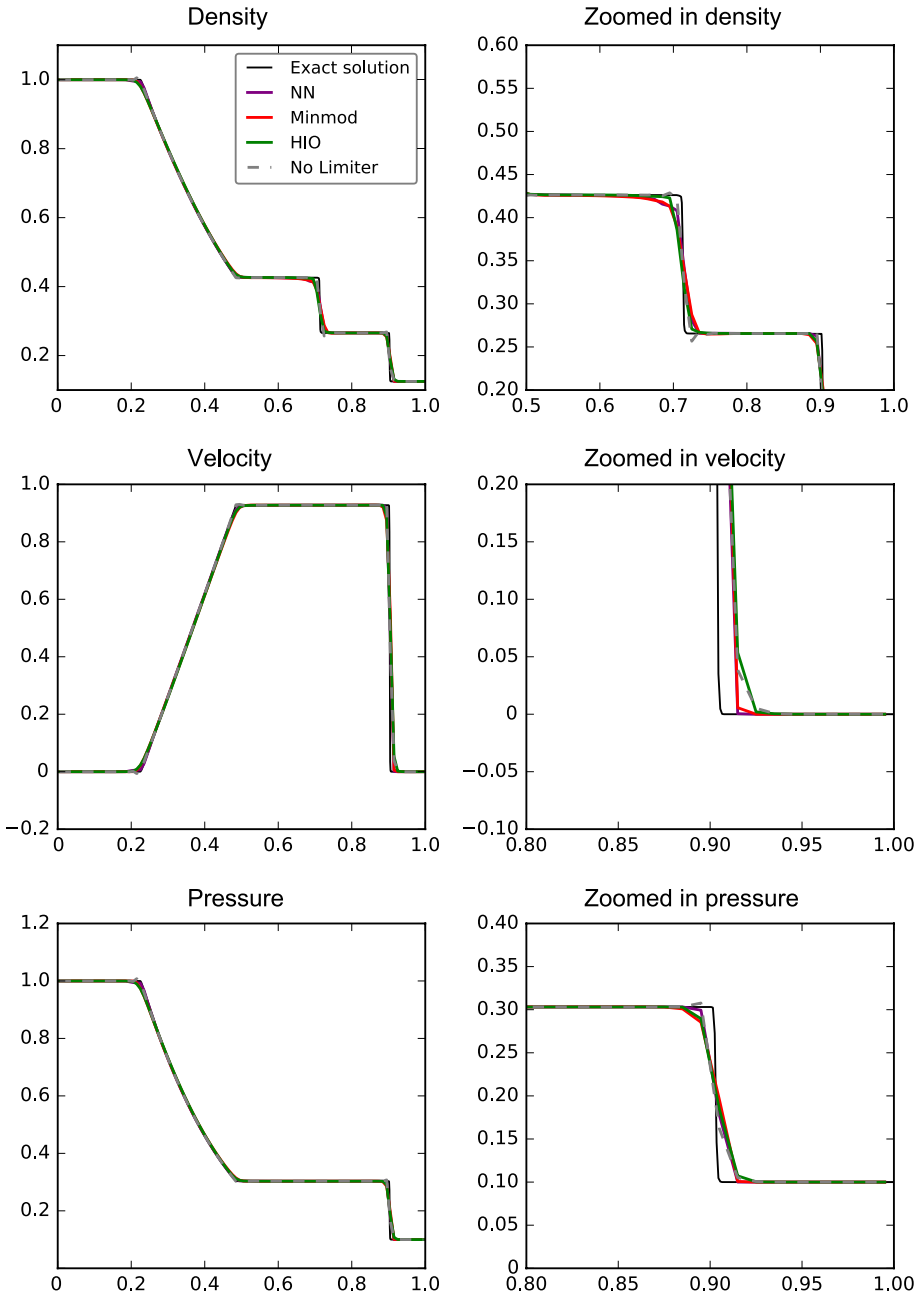
Fig. 8 Density, velocity, and pressure fields for the Sod shock tube (12) at $T = 0.24$ for grid size $N = 100$ and approximation order 3
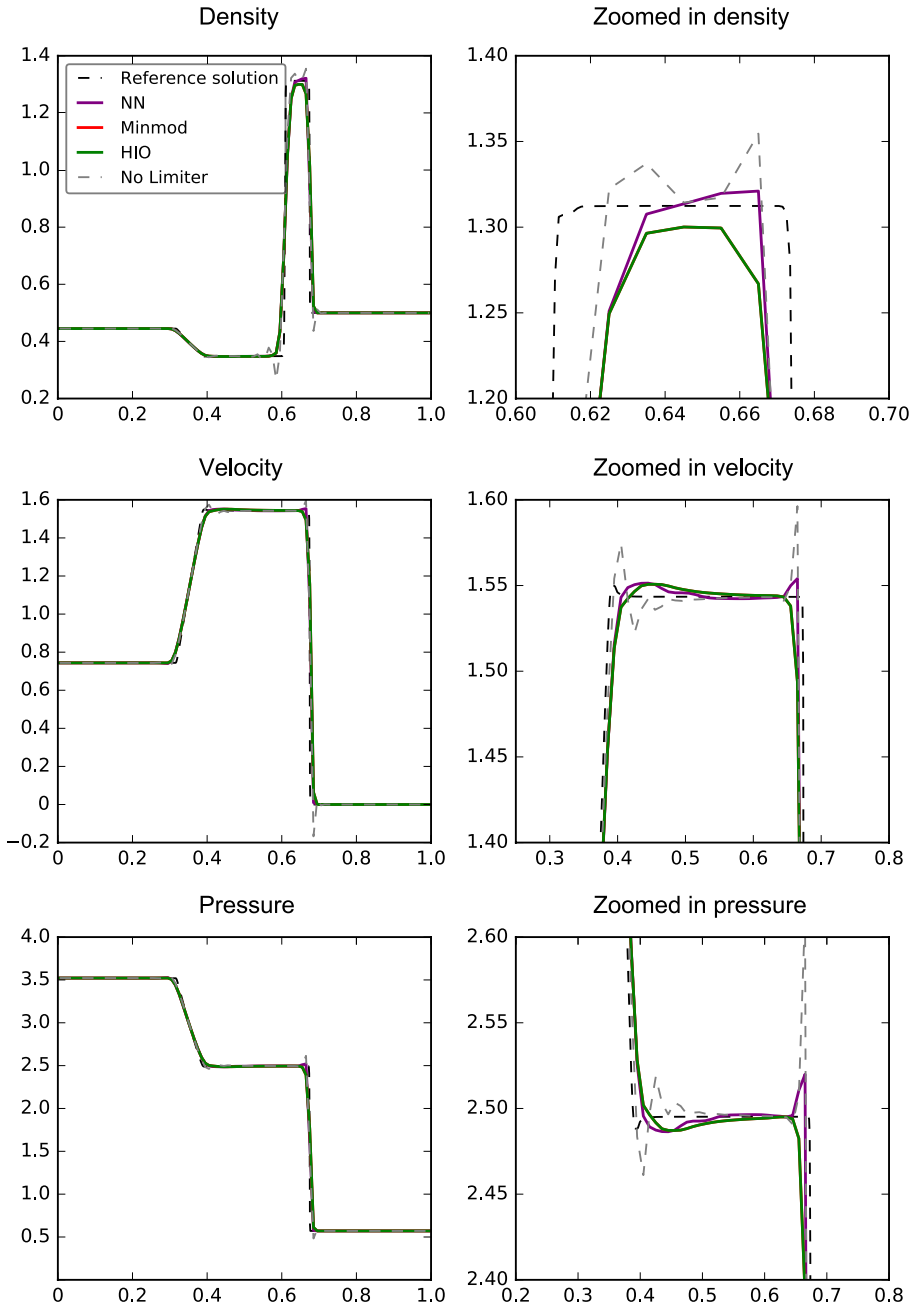
Fig. 9 Density, velocity, and pressure fields for the Sod shock tube (13) at $T = 0.24$ for grid size $N = 100$ and approximation order 2
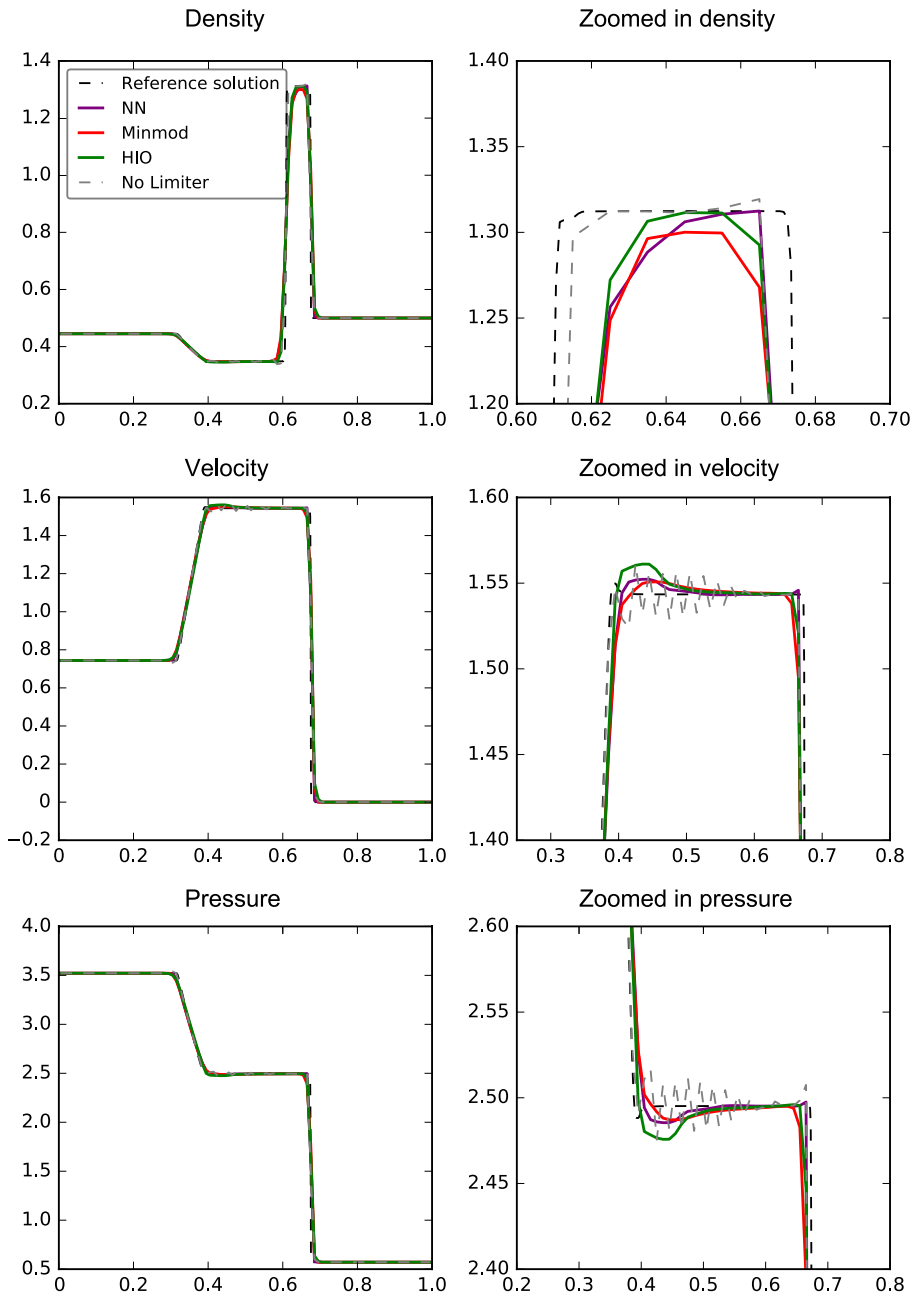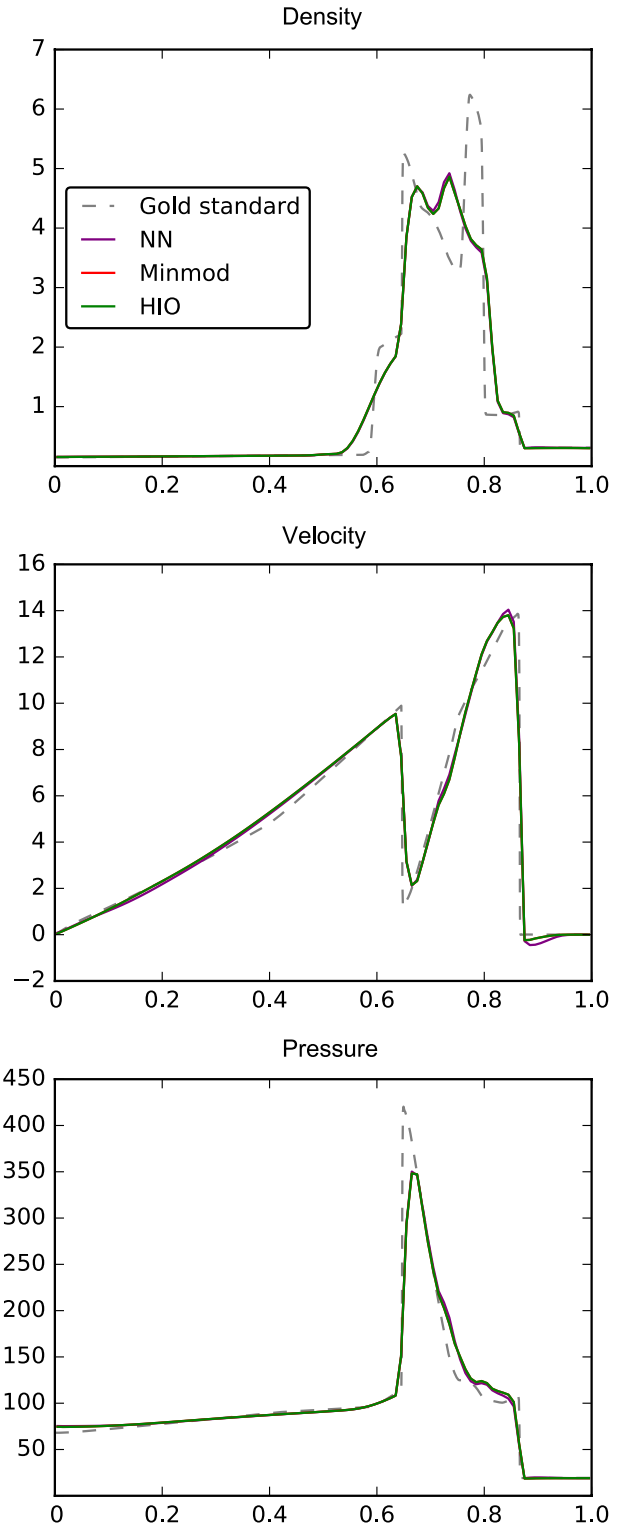
Fig. 10 Density, velocity, and pressure fields for the Sod shock tube (13) at $T = 0.24$ for grid size $N = 100$ and approximation order 3

Fig. 11 Density, velocity, and pressure fields of the blast wave interaction (14) at $T = 0.038$ for grid size $N = 100$ and approximation order 2

For both the second- and third-order results and a fixed resolution $N = 100$, the density, velocity and pressure fields at $T = 0.24$ are shown in Figs. 7 and 8. We note the solution produced by the NN limiter seems oscillation free and similar to the solution of the HIO limiter at both orders.

Case of Lax shock tube We consider the Lax shock tube test [25], given by the initial conditions:

$$(\rho, v, p)(x, 0) = \begin{cases} (0.445, 0.698, 3.528), & 0 < x \leqslant 0.5, \\ (0.5, 0, 0.571), & 0.5 < x < 1.0, \end{cases} \quad (x, t) \in [0, 1] \times [0, 0.08], \tag{13}$$

and $\gamma = 1.4$ and gradient-free boundary conditions.

In Figs. 9 and 10, we show the comparison between different limiters at $T = 0.08$, for schemes of order 2 and 3, respectively. In Fig. 9, we show the density, velocity, and pressure fields for a second-order scheme and resolution $N = 100$. We note the NN limiter seems less diffusive than the HIO limiter, but also not all overshoots are well controlled. This is more obvious in the velocity and pressure fields. In Fig. 10, the same quantities are shown, but for a third-order scheme. Again, we note both the HIO and NN limiters are less diffusive than the MIN limiter.

Case of blast wave Next we consider the interacting blast waves test, given by the initial conditions:

$$(\rho, v, p)(x, 0) = \begin{cases} (1.0, 0, 1\ 000.0), & 0 < x \leqslant 0.1, \\ (1.0, 0, 0.01), & 0.1 < x \leqslant 0.9, \\ (1.0, 0, 100.0), & 0.9 < x < 1.0, \end{cases} \quad (x, t) \in [0, 1] \times [0, 0.038] \tag{14}$$

with $\gamma = 1.4$ and reflexive boundary conditions.

In Figs. 11 and 12, we show the comparison between different limiters at $T = 0.038$ for different orders. The unlimited solution is not shown because the code crashes due to the pressure becoming negative shortly after the start of the simulation for orders higher than 1. The dashed line denotes a high-resolution solution, run with $N = 1\ 000$, the third order with the HIO limiter. We can note that the NN limiter is not as good at suppressing oscillations as the MIN limiter and the HIO limiter, but stabilises the solution enough to finish the run. Furthermore, we note that the peak is better preserved, which means that it looks like the limiting is less strong than the MIN limiter and the HIO limiter.

6.3 Transfer to RD

In this section, we show the performance of the NN limiter applied to the RD scheme, and we compare with a state of the art limiting technique, MOOD [3]. Other stabilisation strategies for the RD require parameter tuning which are problem dependent.

Case of Sod shock The initial conditions are given as in (12). A qualitative result is shown in Fig. 13. It can be noted that the NN limiter is slightly more diffusive than MOOD, but that it seems to control some of the undershoots better.

Fig. 12 Density, velocity, and pressure fields of the blast wave interaction (14) at $T = 0.038$ for grid size $N = 100$ and approximation order 3
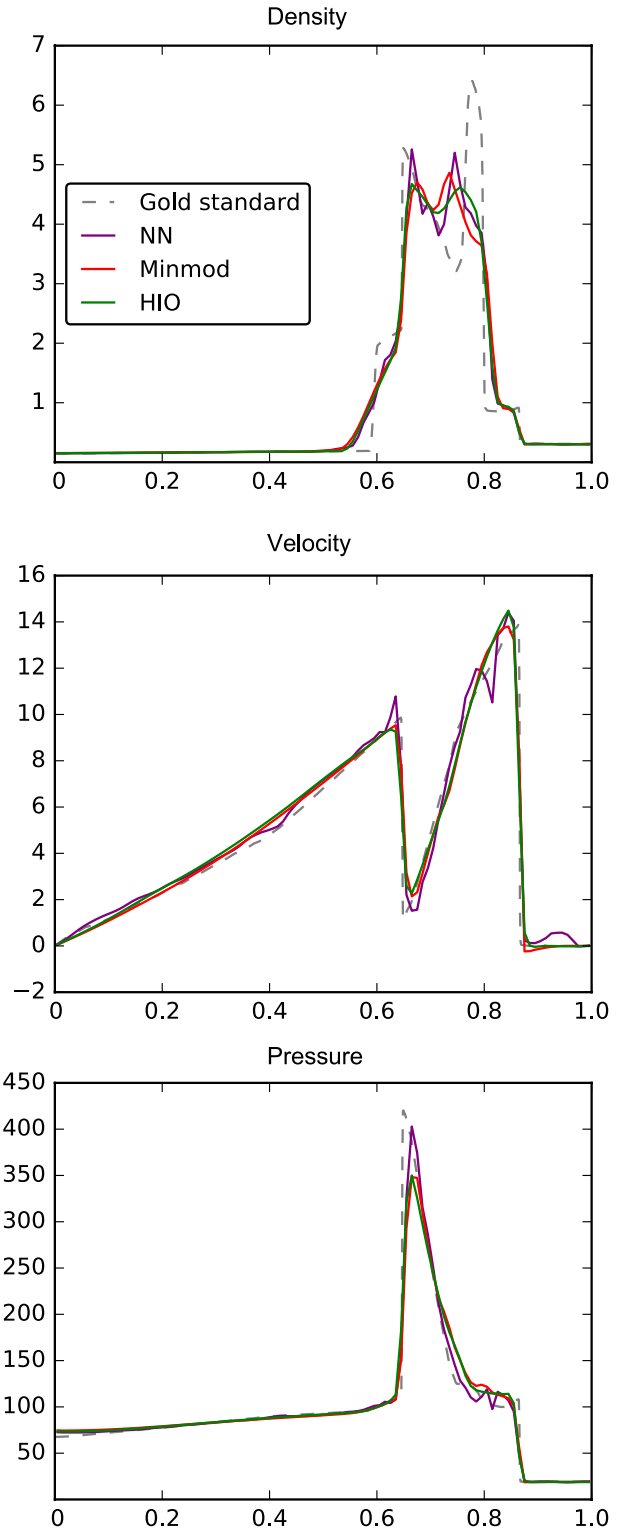
Case of blast wave The initial conditions are given as in (14). A qualitative result is shown in Fig. 14. In this example, one can see that MOOD is significantly less diffusive than the NN limiter.

7 Two-Dimensional Numerical Experiments

Similar to Sect. 6, we first train a set of neural networks varying the number of neurons and layers, and select the one with best recall/precision score.

We then compare the performance of the NN limiter with the MIN limiter and the HIO limiter through the L^1 error norm. We perform some tests for the linear advection equation and Euler system of equations. The initial conditions are chosen to be different from the ones used for the training. In Sect. 7.3, we show the results for the transfer to a RD scheme, for structured and unstructured meshes.

7.1 Numerical Validation

7.1.1 Linear Advection

Consider a linear advection equation with $\mathbf{a} \in \mathbb{R}^2$:

$$\frac{\partial}{\partial t} u + \mathbf{a} \cdot \nabla u = 0 \tag{15}$$

and periodic boundary conditions.

Smooth initial condition We consider the following initial conditions, which contain a smooth function:

$$u_0(x) = 1 + \sin(2\pi r)^{10}, \quad (x, t) \in [0, 1] \times \mathbb{R}^+ \tag{16}$$

with the advection velocity $\mathbf{a} = (1, 1)$ and periodic boundary conditions.

The errors and convergence rates are shown in Table 9 after one full crossing for orders 2 and 3 and different methods. The key-point to note is that the error between the unlimited case and the NN case are very similar, meaning that the shock-detector was not triggered as much as in the cases of the MIN and the HIO.

Case of smooth pulse and square hat We consider the following initial conditions, which contain a smooth Gaussian pulse and a hat function, defined in $(\mathbf{x}, t) \in [0, 1]^{2 \times} \mathbb{R}^+$:

$$u_0(x) = \begin{cases} 2, & (|\mathbf{x} - 0.25|, |y - 0.5|) \leqslant (0.1, 0.1), \\ 1 + \exp\left(-100(||\mathbf{x} - \mathbf{x}_1||^2)\right), & x \geqslant 0.5, \\ 1, & \text{otherwise,} \end{cases} \tag{17}$$

again with the advection velocity $\mathbf{a} = (1, 1)$, $\mathbf{x}_1 = (0.75, 0.5)$.

The errors and convergence rates are shown in Table 10 after one full crossing for orders 2 and 3 and different methods.

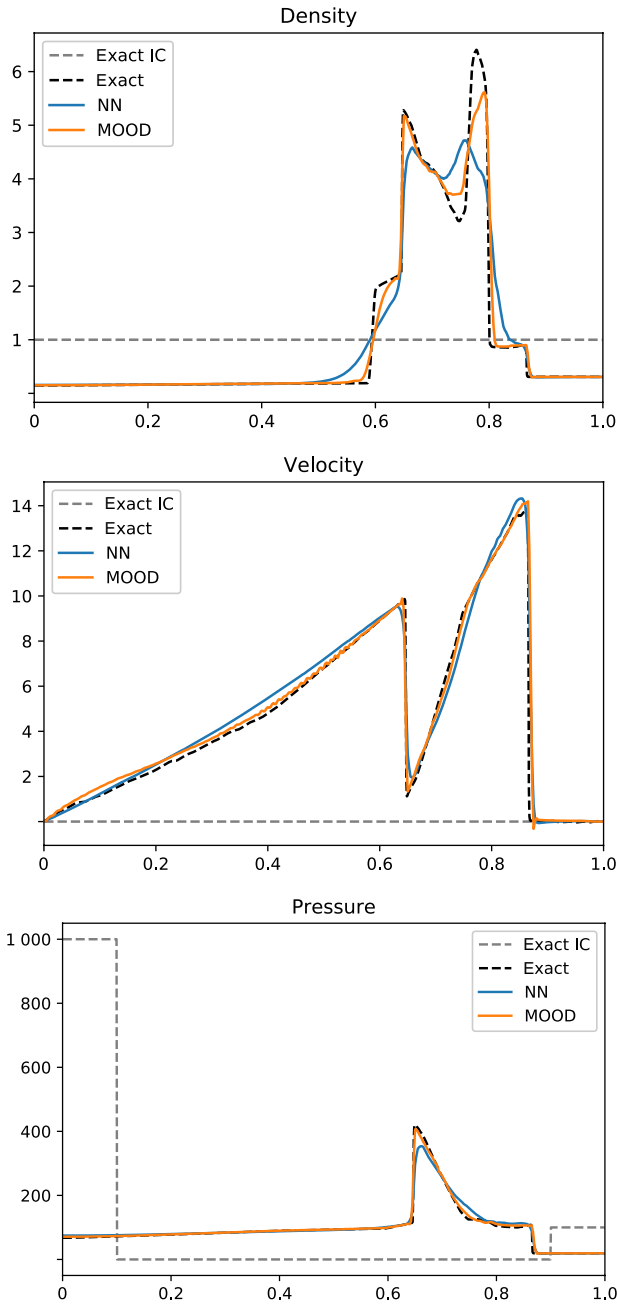**Fig. 13** Density, velocity, and pressure fields of the Sod shock
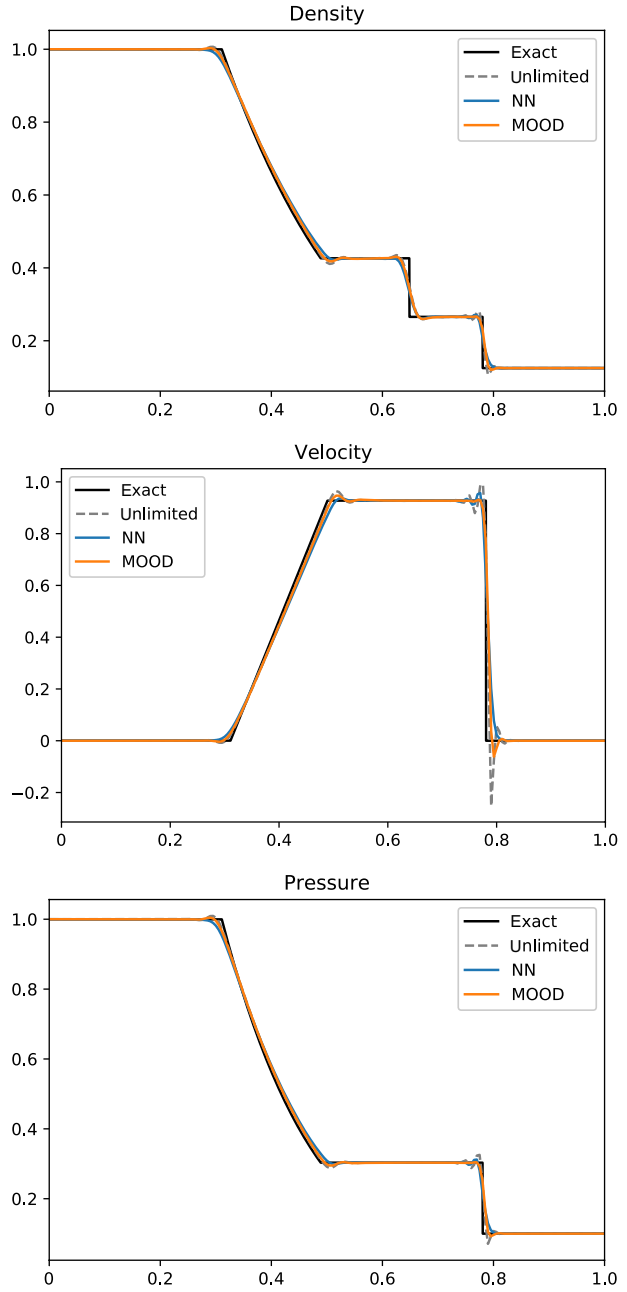
Fig. 14 Density, velocity, and pressure fields of the blast wave test case (14)

Table 9 L^1 error for one crossing of the smooth ring (16) using different limiters

N	No limiter		MIN		HIO		NN	
(a) Order 2								
16^2	6.74E−02	0	1.26E−01	0	1.23E−01	0	6.79E−02	0
32^2	2.00E−02	0.5	4.81E−02	0.4	4.14E−02	0.4	1.97E−02	0.5
64^2	3.45E−03	1.7	1.31E−02	1.4	9.11E−03	1.6	3.45E−03	1.8
128^2	5.42E−04	2.1	2.75E−03	1.6	1.59E−03	1.9	5.41E−04	2.1
(b) Order 3								
16^2	6.92E−03	0	1.26E−01	0	1.17E−01	0	1.88E−02	0
32^2	4.58E−04	0.9	4.80E−02	0.4	2.92E−02	0.4	7.91E−04	0.7
64^2	7.93E−05	3.9	1.31E−02	1.4	6.85E−03	2.0	1.08E−04	4.6
128^2	4.36E−05	3.2	2.75E−03	1.6	1.20E−03	2.0	5.15E−05	3.7

Table 10 L^1 error for one crossing of the Gaussian pulse and hat function (17) using different limiters

N	No limiter		MIN		HIO		NN	
(a) Order 2								
16^2	3.04E−02	0	5.37E−02	0	4.94E−02	0	3.20E−02	0
32^2	2.13E−02	0.6	2.63E−02	0.5	2.63E−02	0.5	2.12E−02	0.6
64^2	1.12E−02	0.5	1.16E−02	1.0	1.01E−02	0.9	1.11E−03	0.6
128^2	8.33E−03	0.7	7.74E−03	1.1	7.74E−03	1.1	8.24E−03	0.8
(b) Order 3								
16^2	1.17E−02	0	5.37E−02	0	5.27E−02	0	2.09E−02	0
32^2	7.61E−03	0.8	2.63E−02	0.5	1.63E−02	0.5	1.05E−02	0.7
64^2	4.63E−03	0.6	1.16E−02	1.0	7.58E−03	1.7	5.73E−03	1.0
128^2	2.91E−03	0.7	7.74E−03	1.1	4.21E−03	1.4	3.37E−03	0.9

7.2 Two-Dimensional Euler Equation

Now we consider the two-dimensional Euler equations, which describe the behaviour of an inviscid flow. This system of equations describes the evolution of a density ρ, a velocity vector $\mathbf{v} = (v_1, v_2)$, a pressure p and total energy E,

$$\partial_t \rho + \nabla \cdot (\rho \mathbf{v}) = 0, \tag{18}$$

$$\partial_t \rho \mathbf{v} + \nabla \cdot \left(\rho \mathbf{v} \otimes \mathbf{v} + p \mathcal{I}_3 \right) = 0, \tag{19}$$
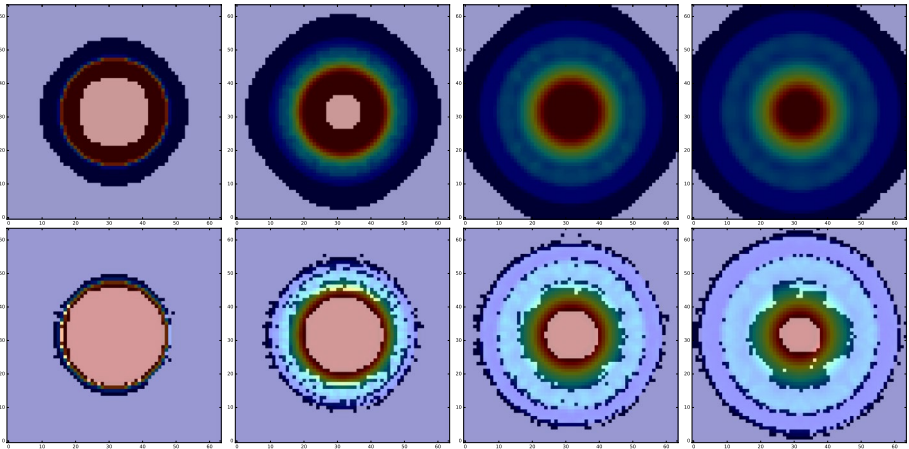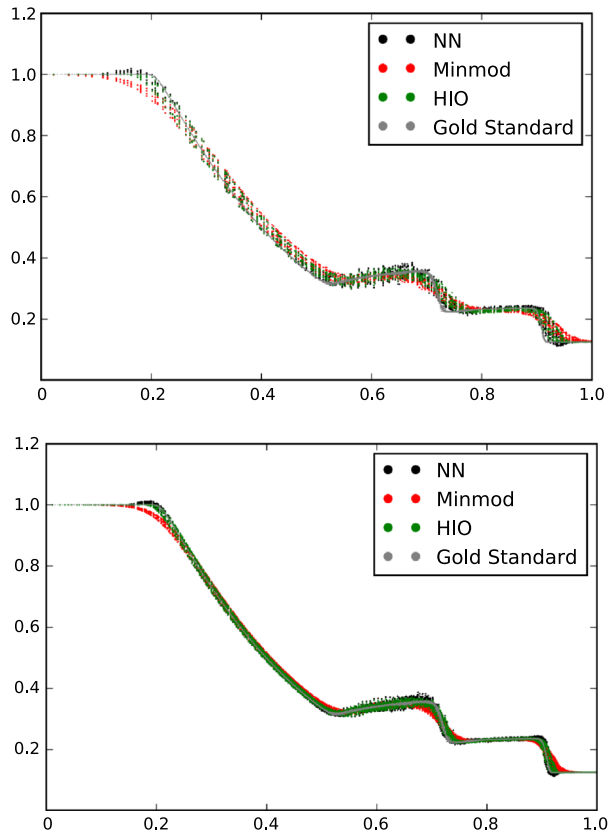
Fig. 15 Detection comparison between HIO limiter (top row) and NN limiter (bottom row) at different times ($t = 0.01, 0.1, 0.2$ and 0.24)

Fig. 16 Scatter plot of density of the two-dimensional Sod shock problem at $T = 0.24$ for $N_{elem} = 64^2$ and $N_{elem} = 128^2$ and approximation order 2

$$\partial_t E + \nabla \cdot (E + p)\mathbf{v} = 0. \tag{20}$$

The system is closed equation of state for an ideal gas

$$\rho e = \frac{p}{\gamma - 1},$$

where $e = E - \frac{1}{2}\rho|\mathbf{v}|^2$ is the internal energy.

Case of the two-dimensional Sod shock tube We consider the radial Sod shock tube test, given by the initial conditions:

$$(\rho, v_x, v_y, p)(x, 0) = \begin{cases} (1.0, 0, 0, 1.0), & 0 < r \leqslant 0.5, \\ (0.125, 0, 0, 0.1), & 0.5 < r < 1.0, \end{cases} \tag{21}$$

where $r = \sqrt{x^2 + y^2}$, $(x, y) \in [0, 1]^2$, $\gamma = 1.4$ and gradient-free boundary conditions.

The solution maps at different increasing times $t = 0.01, 0.1, 0.2$ and 0.24 are shown in Fig. 15. In the top row, the solution maps using the HIO limiter are shown, as well as the cells in which the limiter was triggered (dark regions). We note that there are regions without discontinuities which are still being limited. In the bottom row, we show the solution maps when using the NN shock detector and the areas where the shock-detector detects a shock (and which are subconsequently limited) are overlaid. We note that at the initial time the detection looks quite symmetric; however, over time, this is no longer the case. We also note that the shock fronts are being tracked, while the smooth regions are not being limited.

In Fig. 16, we show the projection of the solution along the radial axis, for $N = 64^2$ and $N = 128^2$. This allows us to see how the different limiters preserve the symmetry of the problem, as the perfect solution would have very little scatter. We see that both the NN shock detector and the HIO limiter appear to perform similarly, and the MIN limiter is more diffusive.
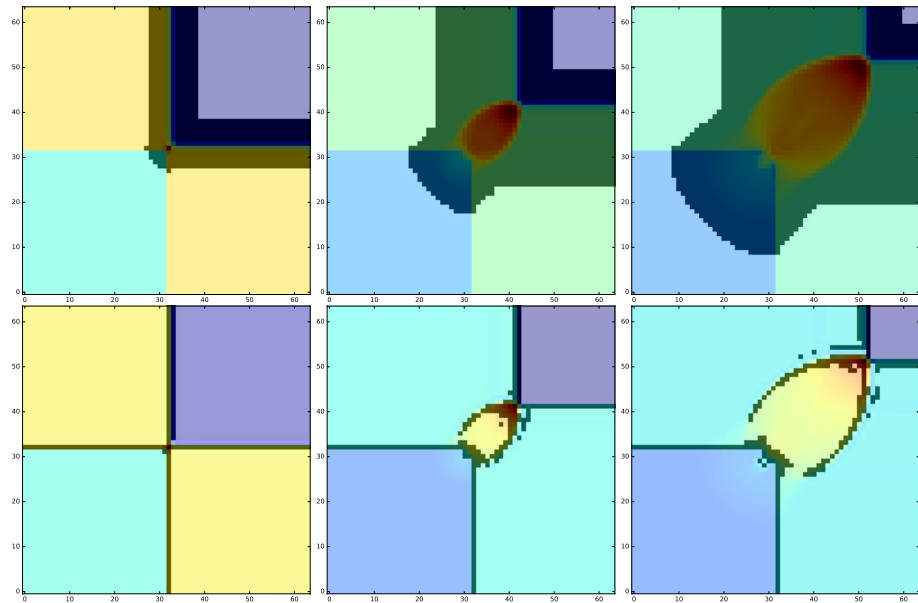

Fig. 17 Detection comparison between HIO limiter (top row) and NN limiter (bottom row) at different times ($t = 0.01, 0.1$ and 0.2)

Case of the Riemann problem 12 We consider a two-dimensional Riemann problem (configuration 12) [24]. The initial data are

$$(\rho, v_x, v_y, p)(x, 0) = \begin{cases} (1, 0.727\ 6, 0, 1), & x < 0, y > 0, \\ (0.8, 0, 0, 1), & x < 0, y < 0, \\ (0.531\ 3, 0, 0, 0.4), & x > 0, y > 0, \\ (1, 0, 0.727\ 6, 1), & x > 0, y < 0, \end{cases} \tag{22}$$

for $\gamma = 1.4$ and gradient-free boundary conditions.

The solution maps at different increasing times $t = 0.01, 0.1$ and 0.2 are shown in Fig. 17. As before, the top row shows the simulation using the HIO limiter is shown, as well as the cells in which the limiter was triggered. Similar to the previous case, there are regions where there is no discontinuity which are still being limited. In the bottom row, we show the solution map when using the NN shock detector and again, we denote the areas where the shock detector detects a shock (and which are sub-consequently limited). We note that the symmetry is quite well preserved across time.
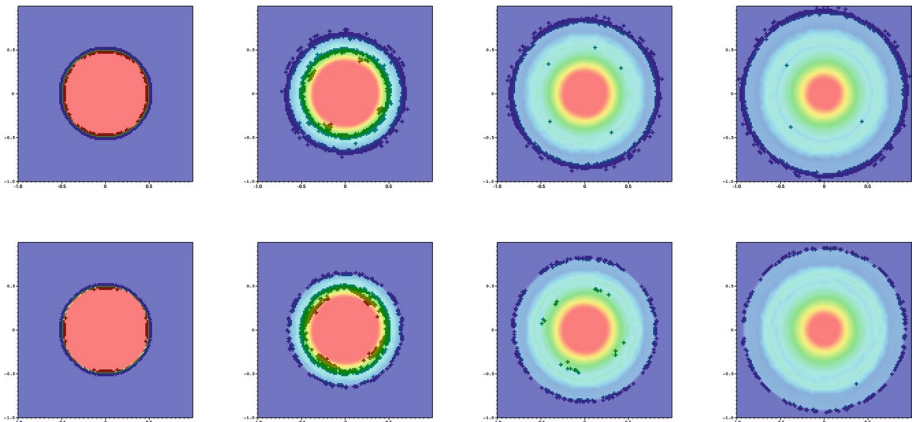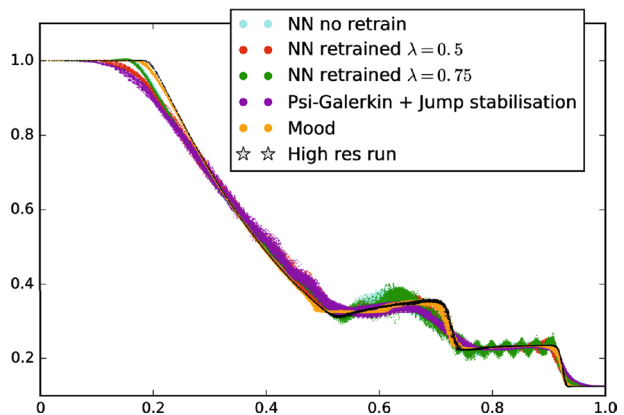

Fig. 18 Detection comparison between the NN limiter without retraining (top row) and with retraining (bottom row) on a structured mesh at different times ($t = 0.01, 0.1, 0.2$ and 0.24)

Fig. 19 Left: comparison between different limiters for the RD scheme on a structured mesh

7.3 Transfer to RD

In this section, we report the results of the NN limiter applied to an RD scheme. In particular, we report on the performance of the NN shock detector when no retraining phase is performed versus when it is retrained on a reduced dataset using data from numerical runs of an RD scheme.

7.3.1 Two-dimensional Sod Shock Tube

In Fig. 18, we show the performance of the two-dimensional shock-indicator function on a two-dimensional RD scheme on a Cartesian mesh which has been trained on RKDG data alone (top row) and retrained using the RD data (bottom row), at different time snapshots $T = 0.01, 0.1, 0.2$, and 0.24. We note that initially, the shock is well captured, but over time
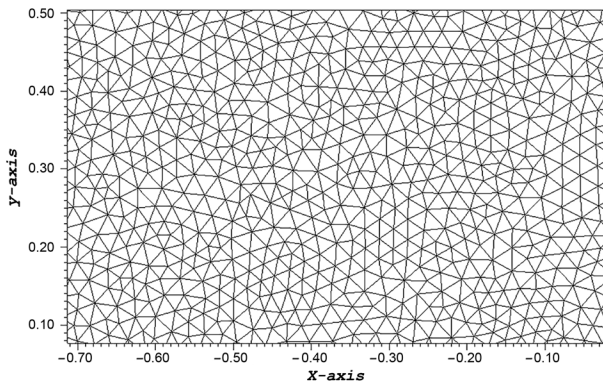

Fig. 20 Zoom on unstructured grid used in numerical experiments shown in Fig. 21
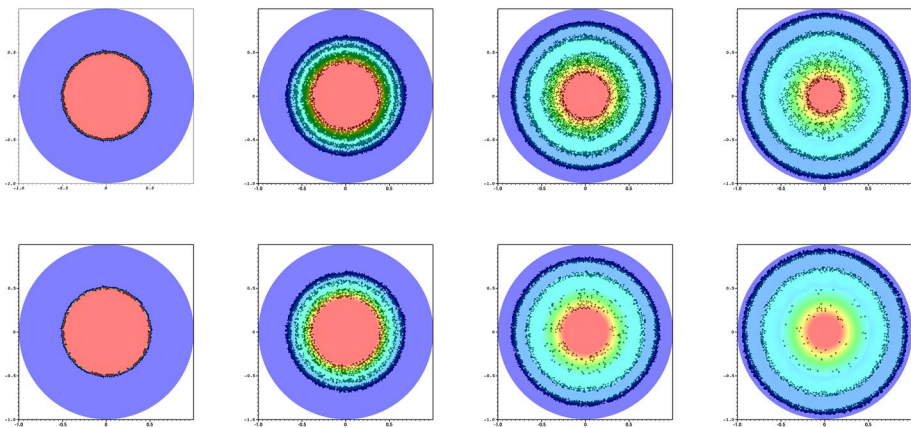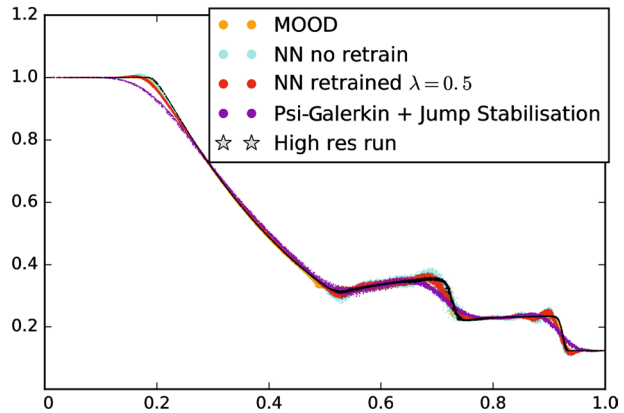

Fig. 21 Detection comparison between the NN limiter without retraining (top row) and with retraining (bottom row) on an unstructured mesh at different times ($t = 0.01, 0.1, 0.2$ and 0.24)

Fig. 22 Comparison between different limiters for the RD scheme on an unstructured mesh

the shock front is no longer well captured. In particular, we can note that the symmetry of the solution with retraining (bottom row) seems to be broken. This is more evident in Fig. 19, where again, the radial projection of the solution is shown. The optimal solution would show a thin spread. The λ parameter shown in the figure provides the ratio between the (re)training data coming from the RKDG scheme and RD scheme. We can see that the more samples are taken from the RD scheme, the less effective the shock-detector becomes. Comparing the performance of the NN limiter with other RD limiters, we can note that it is less diffusive than the RD Psi-Galerkin scheme with Burmann jump stabilisation [8], but significantly worse than the MOOD limiter [3].

Moving onto unstructured grids (see Fig. 20 for one of the grids used), we compare the performance of the transferred neural network shock-indicator without and with retraining on the RD data on an unstructured grid. In Fig. 21, we show again solution maps using the shock-indicator trained only on RKDG data (top row) and retrained using the RD data (bottom row), at different time snapshots $T = 0.01, 0.1, 0.20$, and 0.24. We can see that the adapted limiter detects troubled cells in the shock fronts. In the unstructured case, the rarefaction is no longer limited. In Fig. 22, we show again the radial projection of the two-dimensional Sod shock. In this case, it seems like retraining the shock-indicator yields a solution which is more diffusive, but less oscillatory. Comparing with other limiters available for the RD, we can note that the performance seems similar to the MOOD limiter, and the NN-based limiter is less diffusive than the Psi-Galerkin scheme with jump stabilisation.

8 Discussion

We showed that it is possible to learn a parameter-free shock-detector (after the training phase) from data. While the performance on the training set is relatively good (as observed in Sect. 6.1), the on-the-fly performance could be improved. One way would be to have a more representative training dataset in conjunction with the careful design of the loss function during the training phase (for example, to include information on the maximum preserving principle).

For the advection cases, we observed that this limiter was by far less diffusive than the MIN limiter. In the systems cases, some oscillations were corrected but there were other oscillations which were not stabilized enough, although none of the simulations crashed
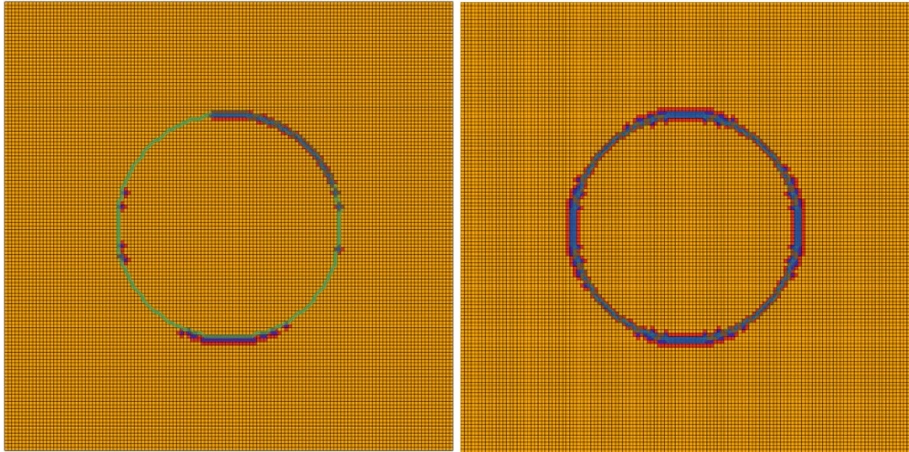
Fig. 23 No symmetry invariance considerations versus explicit symmetry invariance

(whereas the unlimited solution was too unstable, namely for the blast wave case). Overall, the direct application of this model to unseen initial conditions for the advection equation and to the Euler system was somewhat successful.

We then explored strategies of transferring a trained shock-indicator function to a different numerical scheme, namely to a RD scheme. We tested two strategies: one of simply transforming the feature vector from a RD solution onto a feature vector from a RKDG solution, and another one of retraining a trained classifier on a reduced RD dataset. What we observed was that it was possible to use the shock-indicator function in a RD scheme, but that retraining on a reduced dataset did not make significant difference (perhaps it even deteriorated the performance of the shock-indicator). This is something to explore in future work.

The attractive property of this type of shock-indicator is that, once trained, it can operate as a black-box, parameter-free shock detector. We have verified that both in one- and two-dimensional problems we were able to attain better results when comparing to limiters which have not been properly hand-tuned. However, in the case of the RD, the transferred NN limiter did not perform as well as the MOOD limiter [3, 9, 41].

Some important difficulties we were faced with were as follows.

(i)   Lack of theoretical guarantees and quality of the numerical solution

Throughout the time we worked on this problem, it became apparent that certain properties of the shock capturing function do not arise without a certain amount of careful considerations. For example, it was particularly concerning the fact that the detection was not symmetric with respect to a defined stencil (see Fig. 23). To overcome this problem, the literature typically suggests either a feature transformation which renders a certain property invariant or data augmentation by generating training examples that cover such invariance [27]. We empirically observed that merely performing data augmentation did not perform very well and it led to longer training times (due to the enlargement of the dataset). We then used an ensemble classification where the stencil is permuted and the prediction is averaged (as detailed

in Sect. 3.1.2), such that, for example, a prediction on a particular stencil and the same stencil mirrored along the x-axis yield the same response. This improved the detection a lot in problems that had an obvious symmetry (like the two-dimensional Sod shock).

(ii) Computational performance degradation

    We verified that, when integrating the shock-indicator with existing codes, the classification step was expensive (it entails, at best, a series of matrix multiplications, and at worst, an additional feature vector computation). This meant that even if the limiter was not triggered as often due to the shock-indicator step, the overall cost of using this particular shock-indicator was higher than with a traditional limiter, causing a slow down in all codes we considered.

(iii) Lack of supporting theory on how to optimize, tune and generate deep neural networks

    To this day, the optimization of hyperparameters (such as architecture, shape, size, learning rates related to the NN) is approached mostly through trial and error. While several works [29] estimate lower bounds and necessary complexity of the network to capture a given complexity of the function to be learned, these results remain rather far from concrete applications.

8.1 Future Developments

Taking the previous list of main difficulties that we were faced with during this work, we can guide the discussion about our planned future developments and directions.

(i) Lack of theoretical guarantees and quality of the numerical solution

    What we proposed in this work was exact invariance of the network with respect to rotations of the stencil. This adds some cost to the method during the classification phase. Recent ideas to introduce not invariance of the NN, but equivariance, has been explored in [12, 13] in the context of computer vision, and could be an interesting direction going forward.

(ii) Computational performance degradation

    We verified integrating the shock-indicator with existing codes lead always to a computational performance degradation. This is not to say it is always going to be too computationally expensive—there have been some successful examples of learning a reduced network from a larger NN, and with this, a large computational gain has been observed [19].

(iii) Lack of supporting theory on how to optimize, tune and generate deep NNs

    Using the python package hyperopt [5], designed to optimize hyperparameters over awkward search spaces with real-valued, discrete, and conditional dimensions, which makes it ideal for iterating machine learning hyper-parameters, we have been able to obtain similar performing neural network-based shock-indicators without any hand-tuning and which surprisingly had much less degrees of freedom than the one used in this work.

    In particular, defining the hyper-parameter space as

- number of hidden layers, $n \in \{1, 2, 3, 4\}$;
- number of neurons per layers $p \in \{8, \cdots, 128\}$, allowed us to search for a good configuration of the deep NN.

(iv) Better understanding of how to perform transfer learning

The transfer learning approach we took in this project was a very simple (and possibly naive one). We are interested in exploring symmetric feature space transformations, which take both (or many) feature spaces $\mathcal{X}_1, \cdots, \mathcal{X}_n$ (one can imagine coming from different numerical schemes) and finding a common feature space for adaptation purposes.

9 Conclusion

The purpose of this work was primarily to demonstrate the potential of using learning algorithms in CFD codes to automate away some parameter tuning, which is a common practice when using stabilisation methods, as well as to explore the idea of transferring knowledge across numerical schemes. In particular, we detailed the different stages necessary to train a black-box shock-detector that can be integrated with a limiter in different codes. To this end, we described how to construct a dataset for the training phase, how to set up a deep NN (a multilayer perceptron) that detects shocks, and how to integrate the trained model with existing CFD codes. We performed numerical experiments to validate the performance of the shock-detector (paired with a limiter) in the context of scalar and systems of equations. Furthermore, we use the model trained on the data generated with a DG code on an RD code, exploring different ways to perform the domain adaptation.

We then explored strategies of transferring a trained shock-indicator function to a different numerical scheme, namely, to an RD scheme.

We also found that for the exact task of shock-detection, a neural network-based limiter has some notable drawbacks in comparison to some limiters which are both quite agnostic to the underlying numerical method and require minimal parameter tuning (e.g., MOOD [3]). In particular, due to the lack of a systematic way to introduce domain knowledge onto the data-driven model (e.g., positivity of some quantities or the notion of maximum principle, rotation invariance, to name a few), we are of the opinion this neural network-based limiter is in the stage of being a prototype. However, it is our belief that these ideas can be applied to other problems which depend on certain local properties of the numerical solution, ultimately contributing towards CFD codes which are robust to different initial conditions and that require less parameter tuning to produce readily usable results.

With this in mind, we covered a few of the future improvements planned for this NN-based limiter.

Compliance with Ethical Standards

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

Appendix A MLP Architectures

We detail the architectures tested in this work in Table A1. For this work, we found the lower bound (2^{28}) for the number of weights and fixed the activation functions to be ReLU (as detailed in Sect. 3.1.1). What we observed was that this quantity was not producing very good models. Thus, we used (2^{32}) non-zero weights, which presupposes a constant C of order 10. It is left to specify the distribution of the weights across the different layers. The number of weights and neurons is related by multiplying d through the number of neurons per layers to get the total number of weights.

Table A1 Architectures

Model	Layers	Neurons per layer	Description
Model 1	2	16 384:16 384^3	–
Model 2	3	2 048:1 024:512	–
Model 3	4	512:256:256:128	–
Model 4	5	256:128:64:64:32	–
Model 5	5	256:128:64:64:32	With weighted loss $\omega = 5$

Memory requirements were too high when keeping the same amount of weights (because in this case, weights = neurons), so this model was reduced.

References

1. Abgrall, R.: Residual distribution schemes: current status and future trends. Comput. Fluids **35**(7), 641–669 (2006). https://doi.org/10.1016/j.compfluid.2005.01.007
2. Abgrall, R., Bacigaluppi, P., Tokareva, S.: High-order residual distribution scheme for the time-dependent Euler equations of fluid dynamics. Comput. Math. Appl. **78**(2), 274–297 (2019). https://doi.org/10.1016/j.camwa.2018.05.009
3. Bacigaluppi, P., Abgrall, R., Tokareva, S.: "A posteriori" limited high order and robust residual distribution schemes for transient simulations of fluid flows in gas dynamics. arXiv:1902.07773 (2019)
4. Beck, A., Zeifang, J., Schwarz, A., Flad, D.: A neural network based shock detection and localization approach for discontinuous Galerkin methods (2020). https://doi.org/10.13140/RG.2.2.20237.90085
5. Bergstra, J., Yamins, D., Cox, D.: Making a science of model search: hyperparameter optimization in hundreds of dimensions for vision architectures. In: Dasgupta, S., McAllester, D. (eds.) Proceedings of the 30th International Conference on Machine Learning, Proceedings of Machine Learning Research, vol. 28, pp. 115–123. PMLR, Atlanta (2013). http://proceedings.mlr.press/v28/bergstra13.html
6. Biswas, R., Devine, K.D., Flaherty, J.E.: Parallel, adaptive finite element methods for conservation laws. Appl. Numer. Math. **14**(1), 255–283 (1994). https://doi.org/10.1016/0168-9274(94)90029-9
7. Bottou, L., Curtis, F., Nocedal, J.: Optimization methods for large-scale machine learning. SIAM Rev. **60**(2), 223–311 (2018). https://doi.org/10.1137/16M1080173
8. Burman, E., Fernández, M.A.: Continuous interior penalty finite element method for the time-dependent Navier-Stokes equations: space discretization and convergence. Numer. Math. **107**(1), 39–77 (2007). https://doi.org/10.1007/s00211-007-0070-5
9. Clain, S., Diot, S., Loubère, R.: Multi-dimensional optimal order detection (MOOD)—a very high-order finite volume scheme for conservation laws on unstructured meshes. In: Fořt, J., Fürst, J., Halama, J., Herbin, R., Hubert, F. (eds.) Finite Volumes for Complex Applications VI Problems & Perspectives, pp. 263–271. Springer, Berlin (2011)

10. Cockburn, B., Shu, C.W.: TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws II: general framework. Math. Comput. **52**(186), 411–435 (1989). http://www.jstor.org/stable/2008474

11. Cockburn, B., Shu, C.W.: The Runge-Kutta discontinuous Galerkin method for conservation laws V. J. Comput. Phys. **141**(2), 199–224 (1998). https://doi.org/10.1006/jcph.1998.5892

12. Cohen, T.S., Geiger, M., Weiler, M.: A general theory of equivariant CNNs on homogeneous spaces. In: Wallach, H., Larochelle, H., Beygelzimer, A., Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32, pp. 9145–9156. Curran Associates, Inc. (2019). http://papers.nips.cc/paper/9114-a-general-theory-of-equivariant-cnns-on-homogeneous-spaces.pdf

13. Cohen, T.S., Weiler, M., Kicanaoglu, B., Welling, M.: Gauge equivariant convolutional networks and the icosahedral CNN. arXiv:1902.04615 (2019)

14. Dafermos, C.: Hyperbolic Conservation Laws in Continuum Physics. Grundlehren der mathematischen Wissenschaften. Springer, Berlin (2009). https://books.google.com/books?id=49bXK26O_b4C

15. Github repository. https://github.com/hanveiga/1d-dg-nn

16. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. The MIT Press, Massachusetts (2016). http://www.deeplearningbook.org

17. Gottlieb, S.: On high order strong stability preserving Runge-Kutta and multi step time discretizations. J. Sci. Comput. **25**(1), 105–128 (2005). https://doi.org/10.1007/BF02728985

18. Harten, A., Lax, P.D.: On a class of high resolution total-variation-stable finite-difference schemes. SIAM J. Numer. Anal. **21**(1), 1–23 (1984). http://www.jstor.org/stable/2157043

19. He, Y. et al.: Streaming end-to-end speech recognition for mobile devices. In: 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6381–6385 (2019). https://doi.org/10.1109/ICASSP.2019.8682336

20. Hoens, T.R., Chawla, N.V.: Imbalanced Datasets: From Sampling to Classifiers. Wiley, New York (2013). https://doi.org/10.1002/9781118646106.ch3

21. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. ICLR (2015). arXiv:abs/1412.6980

22. Krivodonova, L.: Limiters for high-order discontinuous Galerkin methods. J. Comput. Phys. **226**, 879–896 (2007). https://doi.org/10.1016/j.jcp.2007.05.011

23. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Proceedings of the 25th International Conference on Neural Information Processing Systems—Volume 1, pp. 1097–1105. Curran Associates Inc., USA (2012). http://dl.acm.org/citation.cfm?id=2999134.2999257

24. Kurganov, A., Tadmor, E.: Solution of two-dimensional Riemann problems for gas dynamics without Riemann problem solvers. Numerical Methods for Partial Differential Equations **18**(5), 584–608 (2002). https://doi.org/10.1002/num.10025.

25. Lax, P.D.: Weak solutions of nonlinear hyperbolic equations and their numerical computation. Commun. Pure Appl. Math. **7**(1), 159–193 (1954). https://doi.org/10.1002/cpa.3160070112

26. Mhaskar, H., Liao, Q., Poggio, T.A.: When and why are deep networks better than shallow ones? In: AAAI, pp. 2343–2349 (2017)

27. Mikolajczyk, A., Grochowski, M.: Data augmentation for improving deep learning in image classification problem. In: 2018 International Interdisciplinary PhD Workshop (IIPhDW), pp. 117–122 (2018)

28. Morgan, N.R., Tokareva, S., Liu, X., Morgan, A.: A machine learning approach for detecting shocks with high-order hydrodynamic methods. https://doi.org/10.2514/6.2020-2024

29. Petersen, P., Voigtlaender, F.: Optimal approximation of piecewise smooth functions using deep ReLU neural networks. arxiv:1709.05289 (2017)

30. Prechelt, L.: Early Stopping—But When? In: Montavon G., Orr G.B., Müller K.R. (eds.) Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science, vol. 7700, pp. 53–67, Springer, Berlin, Heidelberg (2012)

31. Ray, D., Hesthaven, J.S.: An artificial neural network as a troubled-cell indicator. J. Comput. Phys. **367**, 166–191 (2018). https://doi.org/10.1016/j.jcp.2018.04.029

32. Ricchiuto, M., Abgrall, R.: Explicit Runge-Kutta residual distribution schemes for time dependent problems: second order case. J. Comput. Phys. **229**(16), 5653–5691 (2010). https://doi.org/10.1016/j.jcp.2010.04.002

33. Ricchiuto, M., Abgrall, R., Deconinck, H.: Application of conservative residual distribution schemes to the solution of the shallow water equations on unstructured meshes. J. Comput. Phys. **222**(1), 287–331 (2007). https://doi.org/10.1016/j.jcp.2006.06.024

34. Rojas, R.: Networks of width one are universal classifiers. In: Proceedings of the International Joint Conference on Neural Networks, vol. 4, pp. 3124–3127 (2003). https://doi.org/10.1109/IJCNN.2003.1224071

35. Rojas, R.: Deepest Neural Networks. arxiv:1707.02617 (2017)

36. Schaal, K. et al: Astrophysical hydrodynamics with a high-order discontinuous Galerkin scheme and adaptive mesh refinement. Mon. Not. R. Astron. Soc. **453**(4), 4278–4300 (2015). https://doi.org/10.1093/mnras/stv1859

37. Snyman, J.: Practical Mathematical Optimization: an Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms. Applied Optimization. Springer, New York (2005). https://books.google.ch/books?id=0tFmf_UKl7oC

38. Sutskever, I., Hinton, G.E.: Deep, narrow sigmoid belief networks are universal approximators. Neural Comput. **20**(11), 2629–2636 (2008). https://doi.org/10.1162/neco.2008.12-07-661

39. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Proceedings of the 27th International Conference on Neural Information Processing Systems—Volume 2, pp. 3104–3112. MIT Press, Cambridge (2014). http://dl.acm.org/citation.cfm?id=2969033.2969173

40. Veiga, M.H., Abgrall, R.: Towards a general stabilisation method for conservation laws using a multi-layer perceptron neural network: 1d scalar and system of equations. In: European Conference on Computational Mechanics and VII European Conference on Computational Fluid Dynamics, vol. 1, pp. 2525–2550 (2018). https://doi.org/10.5167/uzh-168538

41. Vilar, F.: A posteriori correction of high-order discontinuous Galerkin scheme through subcell finite volume formulation and flux reconstruction. J. Comput. Phys. **387**, 245–279 (2019). https://doi.org/10.1016/j.jcp.2018.10.050

42. Weiss, K.R., Khoshgoftaar, T.M., Wang, D.: A survey of transfer learning. J. Big Data **3**, 1–40 (2016)

43. Xu, B., Wang, N., Chen, T., Li, M.: Empirical evaluation of rectified activations in convolutional network. CoRR (2015). arXiv:abs/1505.00853