Received: 15 October 2020 | Accepted: 16 December 2020

DOI: 10.1002/pamm.202000347

Deep Learning for Hyperbolic Conservation Laws with Non-convex Flux

Hadi Minbashian[1,*] and **Jan Giesselmann**[1,**]

[1] Department of Mathematics, Technical University of Darmstadt, Darmstadt, Germany

In this work, we investigate the capabilities of deep neural networks for solving hyperbolic conservation laws with non-convex flux functions. The behaviour of solutions to these problems depends on the underlying small-scale regularization. In many applications concerning phase transition phenomena, the regularization terms consist of diffusion and dispersion which are kept in balance in the limit. This may lead to the development of both classical and non-classical (or undercompressive) shock waves at the same time which makes the development of approximation schemes that converge towards the appropriate weak solution of these problems challenging. Here, we consider a scalar conservation law with cubic flux function as a toy model and present preliminary results of an ongoing work to study the capabilities of a deep learning algorithm called PINNs proposed in [1] for solving this problem. It consists of a feed-forward network with a hyperbolic tangent activation function along with an additional layer to enforce the differential equation.

1 Problem Statement

Here we consider scalar hyperbolic conservation laws (HCL) of the form

$$\partial_t u + \partial_x f(u) = 0 \text{ in } [0,T] \times \mathbb{R}, \tag{1}$$

with some given initial data u_0 and flux function f, which is smooth and non-convex. It is well-known that solutions of (1) may develop shock discontinuities in finite time even for smooth initial data. Therefore, one needs to consider weak solutions for this problem, but it turns out that weak solutions are not unique. The uniqueness of solutions is guaranteed if attention is restricted to (weak) solutions satisfying infinitely many entropy inequalities, see [2]. These solutions correspond to the vanishing viscosity limit, that is $u = \lim_{\epsilon \to 0} u^\epsilon$ where u^ϵ solves

$$\partial_t u^\epsilon + \partial_x f(u^\epsilon) = \epsilon \partial_{xx} u^\epsilon. \tag{2}$$

These solutions are called *entropy solutions* or *classical solutions*.

In this paper, for simplicity, we consider Riemann initial data of the form

$$u(0,x) = u_0(x) = \begin{cases} u_- & \text{if } x < 0, \\ u_+ & \text{if } x > 0. \end{cases} \tag{3}$$

According to the theory of HCLs, classical shock solutions to (1) with initial data (3) will satisfy the Lax entropy condition

$$f'(u_-) > \sigma(u_-, u_+) > f'(u_+) \tag{4}$$

where

$$\sigma(u_-, u_+) := \frac{f'(u_+) - f'(u_-)}{u_+ - u_-} \tag{5}$$

is the speed of the shock.

It should be noted, however, that imposing infinitely many entropy inequalities is somewhat questionable from a thermodynamics viewpoint: The second law of thermodynamics provides only one entropy inequality corresponding to the physical entropy but not entropy inequalities for all concave functions. This observation might seem punctilious but for non-convex flux functions f different entropy inequalities rule out different weak solutions and imposing any single entropy inequality does not ensure the uniqueness of solutions. Thus, there is a significant interest in alternative selection criteria. One option is to consider, again, limits of the diffusive regularization (2), but there is a richer family of solution dynamics which are validated from a physical point of view in the context of multi-phase flows. These solutions can be realized as limits of diffusion-dispersion regularizations of (1), namely

$$\partial_t u^\epsilon_\alpha + \partial_x f(u^\epsilon_\alpha) = \epsilon \partial_{xx} u^\epsilon_\alpha + \alpha \epsilon^2 \partial_{xxx} u^\epsilon_\alpha. \tag{6}$$

[*] Corresponding author: minbashian@mathematik.tu-darmstadt.de
[**] giesselmann@mathematik.tu-darmstadt.de

The limit solutions $u_\alpha = \lim_{\epsilon\to 0} u_\alpha^\epsilon$, depend on small-scale effects in particular the choice of diffusion-dispersion ratio $\alpha > 0$. They might contain classical shocks (i.e. satisfying (4)) as well as non-classical (also called undercompressive) shocks representing subsonic phase boundaries in the theory of phase transitions [3]. Solutions containing undercompressive shocks are called *non-classical solutions*. Indeed, for any $\alpha > 0$ the solution u_α will satisfy an entropy inequality for one entropy but not for others.

There is a large number of modern as well as classical schemes, e.g. Lax-Friedrichs scheme, that provably converge to classical solutions that are obtained as limits of (2), see [4]. In contrast, the literature for numerical schemes converging to particular non-classical solutions is rather shallow. Some numerical methods based on the diffusive-dispersive regularization (6) have been proposed, see [4] and references therein. However, the finite difference schemes in [4] depend on the shock strength and one needs to increase the order of the scheme to maintain the diffusion-dispersion balance while the approximate solution is still of first order for (1).

This motivates our interest in employing other approaches to find computational methods for approximating u_α. In particular, we conjectured that (6) could be solved efficiently by so-called Physics Informed Neural Networks (PINNs) and that the neural networks are well suited to handle the extrapolation $\epsilon \to 0$. It turns out, however, that solving (6) by PINNs is much more challenging than we thought in particular when ϵ is small. The remainder of this paper is devoted to outlining the methodology we used and the problems we encountered. It turns out that some of these problems can be overcome if time stepping is used in the neural network.

It is worth noting that diffusive-dispersive regularizations are not the only available selection criteria in hyperbolic conservation laws with non-convex flux functions. Another choice is the so-called kinetic relations. They have also been used for constructing numerical schemes but they seem fairly hard to handle (in particular, in multi-d) in the context of PINNs. This is due to the need to track phase boundaries (i.e. the position of the non-classical shock) and to evaluate traces of the (numerical) solution on both sides of the interface.

2 Approximation Method: Deep Neural Networks

Given the above challenges in designing numerical methods for approximating non-classical solutions of HCLs with non-convex fluxes and, on the other hand, recent success in applying Deep Neural Networks (DNNs) in many research areas, we believe that it is interesting to investigate the capabilities of DNNs for solving (1-3). Here, for simplicity, we consider the cubic flux $f(u) = u^3$ for which one can obtain closed formulas for the solutions u_α of the corresponding Riemann problem, which makes the assessment of the approximate solutions easy.

It is well-known that neural networks, even with a single hidden layer, can serve as universal function approximators [5]. However, the usual deficiency of these networks is that they are blind to the underlying physics of the problem. This manifests itself in the necessity of huge data sets for training. Recently, the machine learning community has witnessed a significant effort to alleviate this need by making neural networks aware of physics. Here, we try to solve the above problem using PINNs which has been introduced in [1]. Below, we briefly review the idea of solving PDEs using deep learning based on [1].

The cornerstone of the PINNs that we are going to investigate here is a simple deep feed-forward neural network with a hyperbolic tangent activation function with no regularization. This network is then coupled with an additional layer which helps to enforce the equation (6) using automatic differentiation at each grid point. To do so, the residual of (6), i.e.

$$\mathrm{Res}(u_\alpha^\epsilon) := \partial_t u_\alpha^\epsilon + \partial_x f(u_\alpha^\epsilon) - \epsilon \partial_{xx} u_\alpha^\epsilon - \alpha\epsilon^2 \partial_{xxx} u_\alpha^\epsilon, \tag{7}$$

is minimized over a set of points of a tensor-product space-time grid

$$\mathcal{X}_{h,k} := \{(t_i, x_j) \in [0,T]\times[a,b] : t_i = ik, x_j = a + jh\}, \tag{8}$$

where h and k are the spatial and temporal grid sizes for equidistant grids in space and time, respectively with $k = Ch$ and $0 < C < 1$. Furthermore, we enforce the initial and boundary conditions over a set of grid points coming from the corresponding boundary points of the above grid. We indicate the inner points in $\mathcal{X}_{h,k}$ by $\mathcal{X}_{h,k}^I$ and the initial/boundary points by $\mathcal{X}_{h,k}^B$. In fact, the above grid $\mathcal{X}_{h,k} = \mathcal{X}_{h,k}^I \cup \mathcal{X}_{h,k}^B$ serves as the training set and we consider the set $\mathcal{X}^{\mathrm{pred}} := \mathcal{X}_{\frac{h}{2},\frac{k}{2}} \backslash \mathcal{X}_{h,k}$ as the test set to assess the prediction performance of the network.

Mathematically speaking, the approximate solution $\tilde{u}(\mathbf{x})$ where $\mathbf{x} := (t,x)^\top \in \mathbb{R}^2$ is defined by the following deep feed-forward neural network:

$$\tilde{u}_\alpha^\epsilon(\mathbf{x}) = \psi\left(\psi\left(\cdots\psi\left(\psi\left(W^1\mathbf{x}+b^1\right)W^2+b^2\right)\cdots W^{K-1}+b^{K-1}\right)W^K+b^K\right), \tag{9}$$

where $\{W^j\}_{j=1}^K$ and $\{b^j\}_{j=1}^K$ with $W^j \in \mathbb{R}^{N\times N}$, $b^j \in \mathbb{R}^N$ for $j = 2, \cdots, K-1$ are the weights and biases of the network to be found via training. Here we assume, for simplicity, that all hidden layers have the same number of neurons, namely N neurons, see Fig. 1. The activation function ψ is chosen to be the hyperbolic tangent. We train the parameters $\{W^j\}_{j=1}^K$ and $\{b^j\}_{j=1}^K$ by minimizing the mean square residual as the loss function, i.e.

$$\mathrm{MSE} := \mathrm{MSE}_I + \mathrm{MSE}_B := \frac{1}{|\mathcal{X}_{h,k}^I|}\sum_{\mathbf{x}\in\mathcal{X}_{h,k}^I} |\mathrm{Res}(\tilde{u}_\alpha^\epsilon(\mathbf{x}))|^2 + \frac{1}{|\mathcal{X}_{h,k}^B|}\sum_{\mathbf{x}\in\mathcal{X}_{h,k}^B} |u_\alpha^\epsilon(\mathbf{x}) - \tilde{u}_\alpha^\epsilon(\mathbf{x})|^2. \tag{10}$$

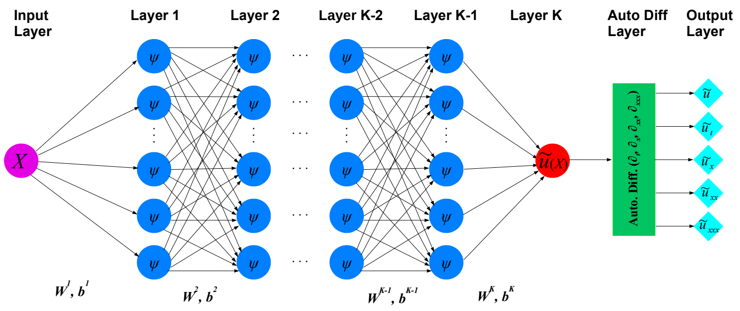


Fig. 1: Physics-Informed Neural Networks (PINNs) for PDEs.

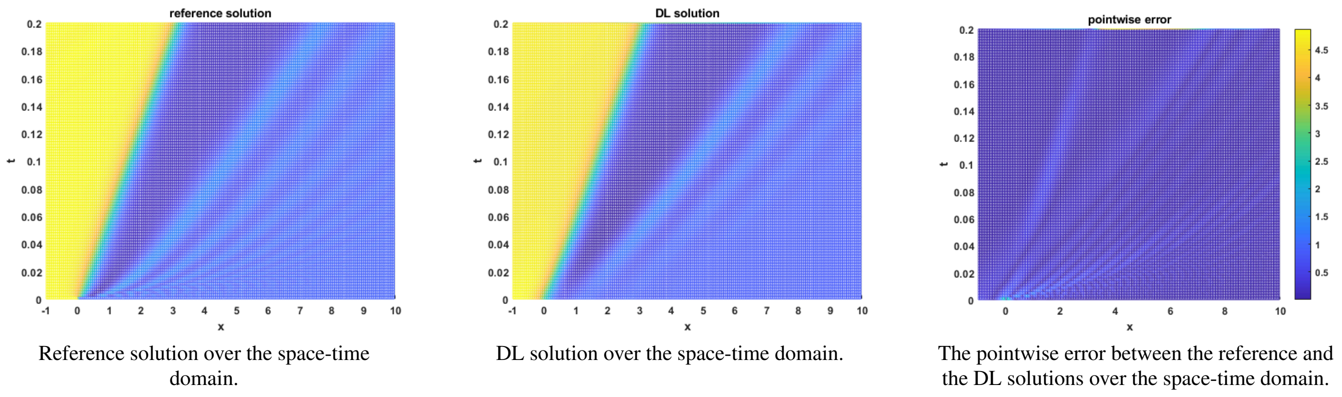
3 Results

In this section, we report some preliminary results for solving (1-3) using PINNs [1] described above based on the regularized equation (6). Since we are using the regularized version of the problem, we expect to see smeared out shocks in the solution. Here, we choose $\alpha = 4$ in (6) with Riemann initial data $u_- = 4,\ u_+ = -2.5$. The exact limit solution u_α, which is computed using a travelling wave argument (see [3] for details), reads

$$u_\alpha(t,x) = \begin{cases} u_- & \text{if } x \le s_1 t, \\ u_m := -u_- + \frac{1}{3}\sqrt{\frac{2}{\alpha}} & \text{if } s_1 t < x \le s_2 t, \\ u_+ & \text{if } s_2 t < x, \end{cases} \tag{11}$$

where the shock speeds are given by $s_1 = \sigma(u_-, u_m)$ and $s_2 = \sigma(u_m, u_+)$ as in (5). With the above Riemann data and α, the limit solution is composed of two right-going shocks: a slow non-classical shock and a fast classical shock with a flat area at a value u_m in between, which is expanded as time goes on, see Fig. 3. Furthermore, since we consider the regularized equation (6) as the approximate model in the deep learning (DL) algorithm as explained before, we need to have a reference solution for that equation, too. Therefore, following the procedure explained in [6] for artificial periodization of the solution, we get a reference solution using the Chebfun package [7] for the equation (6) with a slightly smoothed version of the above Riemann data. Below we consider the final time $T = 0.2$ and a rather large space interval $[-1, 10]$ to avoid local oscillations from seeing the boundary so that the boundary conditions are the same as the corresponding Riemann data at endpoints.

The reference solution with the above Riemann data as well as the DL approximate solution $\tilde{u}^\epsilon_\alpha$ with $\epsilon = 0.5, h = 0.025, k = 0.001$ is shown in Fig. 2. The DL solution is evaluated at test points in all figures. Due to a rather large value of ϵ, low-frequency oscillations are expected in both the reference solution and the DL solution. It is seen from this figure that the DL solution captures the oscillatory behaviour of the reference solution well. However, both differ substantially from the limit solution (11) due to the modelling error since the regularized equation (6) with large ϵ is far from the conservation law (1), see Fig. 3. The residual in all figures is evaluated at training points.



Reference solution over the space-time domain. DL solution over the space-time domain. The pointwise error between the reference and the DL solutions over the space-time domain.

Fig. 2: DL and reference solutions as well as the point-wise error over the space-time domain.

The relative errors in L^1- and L^2-norms for the above DL solution compared to the reference solution with 6 hidden layers and 25 neurons per layer are reported in Table 1. Indeed, we have tested PINNs with many combinations of number of layers and neurons. The number of layers we have tested ran from 4 to 20 and number of neurons per layer from 10 to 30 and we have observed that 6 layers with 25 neurons per layer gives the best result.

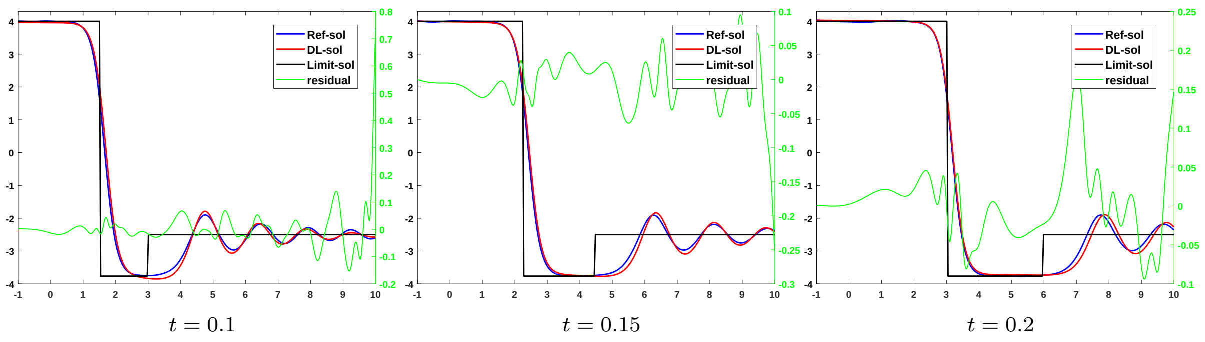


Fig. 3: DL and reference solutions as well as the limit solution at different times when $\epsilon = 0.5$. The pointwise residual is plotted on the right vertical axis.

To have a better approximation of the limit solution, one needs to decrease the regularization parameter ϵ. However, our experiments show that when we decrease ϵ, the DL algorithm is not able to give satisfactory results as the optimization procedure does not reduce the residual of the equation. This is because it gets trapped in local minima and therefore the DL solution no longer approximates the reference solution as well as before with large ϵ, see Fig. 4. Let us point out two particular shortcommings of the DL approximate solution: The (left) non-classical shock is at the wrong location and the (right) Lax-shock, as well as its attached oscillations, are smoothed out over a wide area.

Originally, we had hoped that the distance between DL solution and reference solution would be uniform for $\epsilon \to 0$. However, our experiments show that this is evidently not the case and one needs to find a way to keep $\tilde{u}^\epsilon_\alpha$ close to u^ϵ_α. In this regard, we have tried different weight initializations for the training of the network in addition to the popular random initialization but no significant improvement has been observed. Furthermore, increasing the number of layers or neurons does not seem to improve or deteriorate the solution, which could be seen as some sort of stability worth exploring as a research topic on its own.

In contrast to our results, good agreement between reference solution and DL solution was reported in [1] where a diffusive regularization of Burgers equation, i.e. (1) with $f(u) = u^2$, was considered. The problem at hand is more challenging in several ways: The flux is non-convex, there is a third order (dispersive) term in the regularization and the reference solution is highly oscillatory. Thus, it seems important to understand which of the challenges is responsible for the poor results of our PINNs. To this end, we have tried to approximate classical solutions that are associated with diffusive regularization for our flux function. In particular, these classical solutions are non-oscillatory, e.g., the solution in Fig. 6 consists of a right-going shock attached to a rarefaction wave. Our experiments show that the PINNs provide poor approximations (including an incorrect shock position) when ϵ is small, see Fig. 6. This lets us conclude that convergence of our PINNs to a very poor approximate solution including the wrong shock position is associated with non-convexity of the flux function. How the (non)-convexity of the flux function impacts the landscape of the loss function needs to be investigated further.

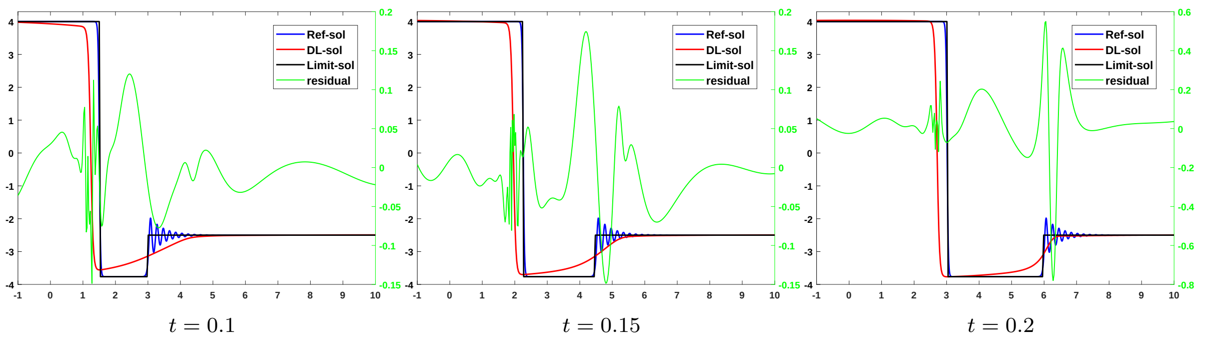


Fig. 4: DL and reference solutions as well as the limit solution at different times when $\epsilon = 0.1$ without time stepping. The DL solution does not capture the shocks due to the local minima of the non-convex loss function. This is evident from rather large point-wise residual.

One promising idea for improving the loss function landscape, for small values of ϵ, and, thereby, preventing convergence of the optimization algorithm to some local minimizer is time-stepping. To this end, we divided the time interval $[0, 0.2]$ in half and used one network per space-time sub-domain. The initial condition for the second half is obtained by evaluating the solution on the first sub-interval at its final time. The idea of time-stepping along with point adaptation has also been investigated in [8] where the goal was to enhance the resolution of the approximate solution. The results for $\epsilon = 0.1$ with and without time-stepping are reported in Table 1. These show that the time-stepping reduces the error in both L^1 and L^2 norms by about 40%, also compare the DL solution in Fig. 5 and Fig. 4. Furthermore, comparing Fig. 5 and Fig. 3 shows that reducing ϵ together with time-stepping results in a better approximation of the limit solution while the relative error between DL solution and reference solution is of the same order, see Table 1. This experiment demonstrates that time-stepping has the potential to alleviate the difficulties encountered in training the network for small ϵ, but it causes a significant increase in training costs since one network needs to be trained per sub-domain. In the case of the classical solution, time-stepping leads to a very well captured rarefaction wave and improves the approximation of the shock position but the error in shock position is still significant, see Fig.7. Comparing the errors in Table 1 for the approximate classical solution (i.e. $\alpha = 0$) with and without time-stepping reveals that time-steping reduces the error in both L^1 and L^2 norms but the error in L^2 is still large due to the wrong shock position. We will investigate time-stepping more precisely in the future.

	without time-stepping			with time-stepping	
	$\alpha = 4, \epsilon = 0.5$	$\alpha = 4, \epsilon = 0.1$	$\alpha = 0, \epsilon = 0.05$	$\alpha = 4, \epsilon = 0.1$	$\alpha = 0, \epsilon = 0.05$
$\lVert\tilde{u}^\epsilon_\alpha - u^\epsilon_\alpha\rVert_{L^1} / \lVert u^\epsilon_\alpha\rVert_{L^1}$	0.0342	0.0922	0.1494	0.0349	0.0782
$\lVert\tilde{u}^\epsilon_\alpha - u^\epsilon_\alpha\rVert_{L^2} / \lVert u^\epsilon_\alpha\rVert_{L^2}$	0.0491	0.3212	0.4654	0.1322	0.2968
$\lVert\tilde{u}^\epsilon_\alpha - u_\alpha\rVert_{L^1} / \lVert u_\alpha\rVert_{L^1}$	0.1363	0.0875	0.1524	0.0768	0.0810
$\lVert\tilde{u}^\epsilon_\alpha - u_\alpha\rVert_{L^2} / \lVert u_\alpha\rVert_{L^2}$	0.2850	0.3286	0.4698	0.1592	0.3045

Table 1: Relative error in L^1 and L^2 norms for approximation of classical solution (i.e. $\alpha = 0$) and non-classical solution (i.e. $\alpha = 4$) w.r.t. corresponding reference solution (the first two rows) and limit solution (the last two rows).

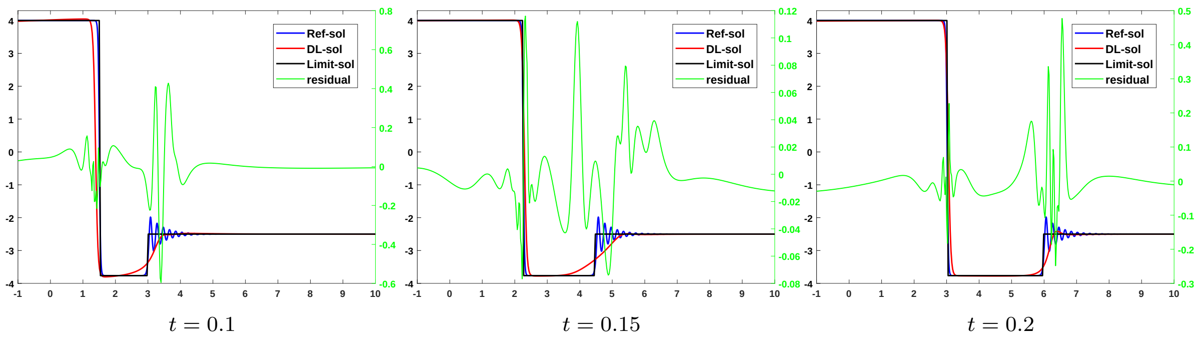


Fig. 5: DL and reference solutions as well as the limit solution (i. e. non-classical solution) at different times when $\epsilon = 0.1$ with time stepping. The solution at $t = 0.1$ is taken from the first half of the time interval, i.e. from $[0, 0.1]$.

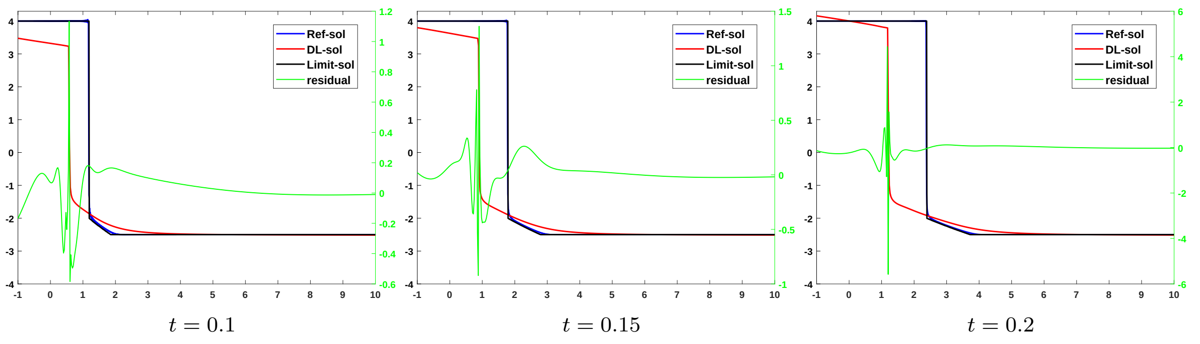


Fig. 6: DL and reference solutions associated with diffusive regularization (2) as well as the limit solution, which is a classical solution composed of a shock and a rarefaction wave at different times when $\epsilon = 0.05$. The DL solution does not capture the classical shock.

4 Conclusion

In this paper, we have reported preliminary results of ongoing research in solving partial differential equations using deep neural networks. In particular, we considered a scalar conservation law with a cubic flux and Riemann initial data. The solution to this problem develops both classical and non-classical shocks, which makes it difficult to approximate the solution using classical numerical methods as they do not respect small-scale effects e.g. capillary effects. Our results show that deep neural networks have the capability to approximate the solution of this problem by considering a diffusive-dispersive

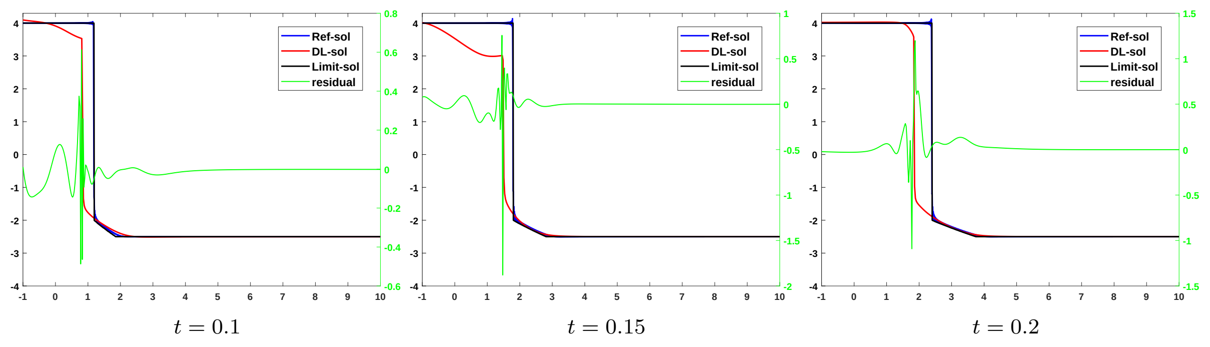


Fig. 7: DL and reference solutions associated with diffusive regularization (2) as well as the limit solution (i.e. classical solution) at different times when $\epsilon = 0.05$ with time stepping. The time-stepping helps in capturing the classical shock as well as the rarefaction wave. The solution at $t = 0.1$ is taken from the first half of the time interval, i.e. from $[0, 0.1]$

regularization which allows accounting for small-scale effects. However, our experiments show that decreasing ϵ in the regularized equation makes the optimization procedure prone to converge to local minima. This difficulty appears for classical solutions too. A promising approach that we have briefly explored so far to tackle this difficulty is time-stepping. We will investigate this strategy in more detail in the future.

Acknowledgements The work of JG is supported by the Graduate School CE within the Centre for Computational Engineering at Technische Universität Darmstadt.
Open access funding enabled and organized by Projekt DEAL.

References

[1] M. Raissi, P. Perdikaris, and G. Karniadakis, J. of Comp. Phy. **378**, 686 – 707 (2019).
[2] S. N. Kružkov, Mathematics of the USSR-Sbornik **10**(2), 217–243 (1970).
[3] P. G. LeFloch, Hyperbolic Systems of Conservation Laws: The Theory of Classical and Nonclassical Shock Waves (Birkhäuser, 2002).
[4] P. G. LeFloch and S. Mishra, Acta Numerica **23**, 743 – 816 (2014).
[5] K. Hornik, M. Stinchcombe, and H. White, Neural Networks **2**(5), 359 – 366 (1989).
[6] D. Jacobs, B. Mckinney, and M. Shearer, Journal of Differential Equations **116**(2), 448 – 467 (1995).
[7] T. A. Driscoll, N. Hale, and L. N. Trefethen, Chebfun Guide (Pafnuty Publications, 2014).
[8] C. Wight, J. Drgona, A. Tuor, D. Bacon, and D. Vrabie, GAMM Juniors' Summer School on Learning Models from Data: Model Reduction, System Identification and Machine Learning(July) (2020).