Limitations of Physics Informed Machine Learning for Nonlinear Two-Phase Transport in Porous Media

Preprint · July 2020

2 authors:

Olga Fuks
Stanford University
11 PUBLICATIONS **15** CITATIONS
SEE PROFILE

Hamdi Tchelepi
Stanford University
338 PUBLICATIONS **6,193** CITATIONS
SEE PROFILE

LIMITATIONS OF PHYSICS INFORMED MACHINE LEARNING FOR NONLINEAR TWO-PHASE TRANSPORT IN POROUS MEDIA

A PREPRINT
PUBLISHED PAPER: FUKS AND TCHELEPI (2020)

Olga Fuks
Department of Energy Resources Engineering
Stanford University
ofuks@stanford.edu

Hamdi A. Tchelepi
Department of Energy Resources Engineering
Stanford University
tchelepi@stanford.edu

July 21, 2020

ABSTRACT

Deep learning techniques have recently been applied to a wide range of computational physics problems. In this paper, we focus on developing a physics-based approach that enables the neural network to learn the solution of a dynamic fluid-flow problem governed by a nonlinear partial differential equation (PDE). The main idea of physics informed machine learning (PIML) approaches is to encode the underlying physical law (i.e., the PDE) into the neural network as prior information. We investigate the applicability of the PIML approach to the forward problem of immiscible two-phase fluid transport in porous media, which is governed by a nonlinear first-order hyperbolic PDE subject to initial and boundary data. We employ the PIML strategy to solve this forward problem without any additional labeled data in the interior of the domain. Particularly, we are interested in non-convex flux functions in the PDE, where the solution involves shocks and mixed waves (shocks and rarefactions). We have found that such a PIML approach fails to provide reasonable approximations to the solution in the presence of shocks in the saturation field. We investigated several architectures and experimented with a large number of neural-network parameters, and the overall finding is that PIML strategies that employ the nonlinear hyperbolic conservation equation in the loss function are inadequate. However, we have found that employing a parabolic form of the conservation equation, whereby a small amount of diffusion is added, the neural network is consistently able to learn accurate approximations of the solutions containing shocks and mixed waves.

Keywords two-phase transport · physics informed machine learning · partial differential equations

1 Introduction

Machine learning (ML) techniques, specifically deep learning (LeCun et al., 2015), are at the center of attention across the computational science and engineering communities. The spectrum of deep learning architectures and techniques has already achieved notable results across applications and disciplines, including computer vision and image recognition (Krizhevsky et al., 2012; He et al., 2016; Karpathy et al., 2014), speech recognition and machine translation (Hinton et al., 2012; Sutskever et al., 2014), robotics (Lillicrap et al., 2015; Mnih et al., 2016), and medicine (Gulshan et al., 2016; Liu et al., 2017). There is no doubt that the range of applications will grow and the impact of ML methods will continue to spread.

Deep learning allows neural networks composed of multiple processing layers to learn representations of raw input data with multiple levels of abstraction. These networks are known to be particularly effective at supervised learning tasks, whereby the successful application of these models usually requires the availability of large amounts of labeled data. However, in many engineering applications, data acquisition is often prohibitively expensive, and the amount of labeled data is usually quite sparse. Specifically, most computational geoscience problems related to modeling subsurface flow dynamics suffer from sparse site-specific data. Consequently, in this "sparse data" regime, it is crucial to employ

domain knowledge to reduce the need for labeled training data, or even aim to train ML models without any labeled data relying only on constraints (Stewart and Ermon, 2017). These constraints are used to encode the specific structure and properties of the output that are known to hold because of domain knowledge, e.g., known physics laws such as conservation of momentum, mass, and energy.

PIML approaches have been explored recently in a variety of computational physics problems, whereby the focus is on enabling the neural network to learn the solutions of deterministic partial differential equations (PDEs). Early works in this area date back to the 1990s (Lagaris et al., 1998; Psichogios and Ungar, 1992; Lee and Kang, 1990; Meade and Fernandez, 1994). However, in the context of modern neural network architectures, the interest in this topic has been revived in (Raissi et al., 2017, 2019; Zhu et al., 2019). These so-called physics informed machine learning (PIML) approaches are designed to obtain data-driven solutions of general nonlinear PDEs, and they may be a promising alternative to traditional numerical methods for solving PDEs, such as finite-difference and finite-volume methods. The core idea of PIML is that the developed neural network encodes the underlying physical law as prior information, and then uses this information during the training process. The approach takes advantage of the neural network capability to approximate any continuous function (Hornik et al., 1989; Cybenko, 1989). The authors of (Raissi et al., 2017) demonstrated the PIML capabilities for a collection of diverse problems in computational science (Burgers' equation, Navier-Stokes, etc). They suggested that if the considered PDE is well-posed and its solution is unique, then the PIML method is capable of achieving good predictive accuracy given a sufficiently expressive neural network architecture and a sufficient number of collocation points. In the current work, we show that the neural network approach struggles and even fails for modeling the nonlinear hyperbolic PDE that governs two-phase transport in porous media. Our experience indicates that this shortcoming of PIML for hyperbolic PDEs is not related to the specific architecture, or the choice of the hyperparameters (e.g., number of collocation points, etc.).

One important class of PDEs is that of conservation laws that describe the conservation of mass, momentum, and energy. In particular, these conservation equations describe displacement processes that are essential for modeling flow and transport in subsurface porous formations, such as water-oil or gas-oil displacements (Aziz and Settari, 1979; Orr, 2007). Numerical reservoir simulation based on solving mass conservation equations with constitutive relations for the nonlinear coefficients is used to make predictions. A major challenge in practice is that the available information/measurements (i.e., labeled data) about the specific geological formation of interest is often quite sparse. Thus, it is important to take advantage of any prior information to improve the predictive reliability of the computational models. The physics of two-phase fluid transport, e.g., water-oil displacements, is described by a nonlinear hyperbolic PDE or a system of PDEs (Orr, 2007). These nonlinear transport problems are known to be quite challenging for standard numerical methods (Aziz and Settari, 1979), and this is largely due to the presence of steep saturation fronts and mixed waves (shocks and spreading waves) in the solution. Specifically, we are interested in solving Riemann problems – initial value problems, when the initial data consist of two constant states separated by a jump discontinuity at $x = 0$.

There are significant efforts aimed at figuring out the potential of machine learning in the modeling of flow processes in large-scale subsurface formations. Thus, it is extremely important to understand the limitations of PIML schemes for making computational predictions of reservoir displacement processes. Here, we investigate the application of physics informed machine learning approach to the "pure" forward problem of nonlinear two-phase transport in porous media. We evaluate the performance of the PIML framework for this problem with different flux (fractional flow) functions. The objective is to assess how well the PIML approach performs for nonlinear flow problems with discontinuous solutions (i.e., shocks).

The paper proceeds as follows. In Section 2, we describe the two-phase transport model and the governing hyperbolic PDE that we aim to solve with a machine learning approach. In Section 3 we provide a brief overview of the physics informed machine learning framework that we use to solve the deterministic PDE. The results for the transport problem with different flux functions are presented in Section 4. Then, to understand the observed behavior of the method we provide a more detailed analysis of the trained neural networks in Section 5. Lastly, in Section 6, we summarize our findings and provide a brief discussion of the results.

2 Two-phase transport model

We consider the standard Buckley-Leverett model with two incompressible immiscible fluids, e.g., oil and water. A nonwetting phase, e.g., oil (o), is displaced by a wetting phase, e.g., water (w), in a porous medium with permeability, $k(\mathbf{x})$, and porosity, $\phi(\mathbf{x})$. Gravity and capillary effects are neglected. Under these assumptions, the pressure, p, and fluid saturations, S_α ($\alpha = \mathrm{o}, \mathrm{w}$), are governed by a coupled system of mass balance equations complemented by Darcy's equations for each phase. After some manipulation (see, e.g., (Aziz and Settari, 1979)), the system can be transformed

into the incompressibility condition for the total flux, $\mathbf{u}_{\text{tot}}$:

$$\nabla \cdot \mathbf{u}_{\text{tot}} = q_{\text{t}}, \tag{1}$$

where q_{t} is a total source (sink) term; and the conservation equation for one of the phases, e.g., water:

$$\phi(\mathbf{x})\frac{\partial S_{\text{w}}}{\partial t} + \nabla \cdot (f_{\text{w}}(S_{\text{w}}) \cdot \mathbf{u}_{\text{tot}}) = q_{\text{w}}. \tag{2}$$

Here $\mathbf{u}_{\text{tot}} = \mathbf{u}_{\text{w}} + \mathbf{u}_{\text{o}}$ is the total flux, $\mathbf{u}_\alpha$ represents the Darcy's flux for a phase ($\alpha = \text{o}, \text{w}$), function f_{w} is called the fractional flow of water or simply, flux function, and defined as follows:

$$f_{\text{w}} = \frac{\lambda_{\text{w}}}{\lambda_{\text{w}} + \lambda_{\text{o}}}, \tag{3}$$

where $\lambda_\alpha = \frac{k\, k_{\text{r}\alpha}}{\mu_\alpha}$ stands for the phase mobility, μ_α is the viscosity of the phase, $k_{\text{r}\alpha}(S_\alpha)$ is the relative phase permeability and q_{w} is a source (sink) term for water. The source or sink terms represent the effect of wells. Equation 2 is supplemented with uniform initial and boundary conditions:

$$\begin{aligned} S_{\text{w}}(\mathbf{x}, t) &= s_{\text{wi}}, \quad \forall \mathbf{x} \text{ and } t = 0, \\ S_{\text{w}}(\mathbf{x}, t) &= s_{\text{b}}, \quad \mathbf{x} \in \Gamma_{\text{inj}} \text{ and } t > 0, \end{aligned} \tag{4}$$

where s_{wi} is the initial water saturation in the reservoir, and s_{b} is the saturation at the injection well or boundary, Γ_{inj}.

In one dimensional space, the Equation 2 becomes:

$$\phi(x)\frac{\partial S_{\text{w}}}{\partial t} + u_{\text{tot}}\frac{\partial f_{\text{w}}(S_{\text{w}})}{\partial x} = 0, \tag{5}$$

and the total velocity, u_{tot}, is constant. After introducing the dimensionless variables $t_D = \int_0^t \frac{u_{\text{tot}} dt'}{\phi L}$ and $x_D = \frac{x}{L}$, where L is the length of the one-dimensional system, we can rewrite the Equation 5 as follows:

$$\frac{\partial S_{\text{w}}}{\partial t_D} + \frac{\partial f_{\text{w}}(S_{\text{w}})}{\partial x_D} = 0, \tag{6}$$

while initial and boundary conditions can be written as:

$$\begin{aligned} S_{\text{w}}(x_D, 0) &= s_{\text{wi}}, \quad \forall x_D \\ S_{\text{w}}(x_D, t_D) &= s_{\text{b}}, \quad x_D = 0 \text{ and } t_D > 0. \end{aligned} \tag{7}$$

Solving the initial value problem (6)-(7) is equivalent to solving the following nonlinear hyperbolic PDE:

$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} = 0, \tag{8}$$

with the piecewise constant initial condition,

$$u(t = 0, x) = u^0(x). \tag{9}$$

Here, $u(t, x)$ is the space-time dependent quantity of interest (conserved scalar) that needs to be solved for, and $f(u)$ is the flux function. The PDE (8) can be solved by the method of characteristics, and it can be shown that the characteristics are straight lines (see, e.g., (Lax, 1973)). If the initial data (9) is piecewise constant having a single discontinuity, i.e., a Riemann problem, the PDE solution is a self-similar function. The hyperbolic PDE of the general form (8) is the main subject of the current work, and in the following, we solve the initial value problem (8)-(9) by applying the physics informed machine learning (PIML) approach.

3 Physics Informed Machine Learning

In this section we consider the following general partial differential equation:

$$u_t + \mathcal{N}(u) = 0, \tag{10}$$

where $\mathcal{N}(\cdot)$ is a nonlinear differential operator.

Neural networks are often regarded as universal function approximators (Hornik et al., 1989; Cybenko, 1989) – which means that a feed-forward network with a single hidden layer containing a finite number of neurons can approximate any continuous function to any desired level of precision. Following the approach of (Raissi et al., 2019), the solution

$u(t,x)$ to the PDE is approximated by a deep neural network parameterized by a set of parameters θ. In other words, the solution to the PDE is represented as a series of function compositions:

$$\begin{aligned} y_1(t,x) &= \sigma(W_1 X + b_1) \\ y_2(t,x) &= \sigma(W_2 y_1 + b_2) \\ &\dots \\ y_{n_l+1}(t,x) &= W_{n_l+1} y_{n_l} + b_{n_l+1} \\ u_\theta(t,x) &= y_{n_l+1}(t,x), \end{aligned} \tag{11}$$

where the input vector, X, contains space and time coordinates, i.e., $X = (x,t)$, θ is the ensemble of all model parameters:

$$\theta = \{W_1, W_2, \dots, W_{n_l+1}, b_1, b_2, \dots, b_{n_l+1}\}, \tag{12}$$

σ is an activation function (tanh in our case) and n_l is the number of hidden layers. Defining $z_i(x) = \sigma(W_i x + b_i)$ for $i = 1, ..., n_l$ and $z_i(x) = W_i x + b_i$ for $i = n_l + 1$, we can write the solution to the PDE as follows:

$$u_\theta(t,x) = z_{n_l+1}(z_{n_l}(\dots z_2(z_1(X)))). \tag{13}$$

The residual of the PDE is just the left-hand side of the Equation (10):

$$r(t,x) = u_t + \mathcal{N}(u). \tag{14}$$

When the PDE solution is approximated by a neural network $u_\theta(t,x)$, the residual of the PDE can be also represented as the neural network with the same parameters θ:

$$r_\theta(t,x) = (u_\theta)_t + \mathcal{N}(u_\theta). \tag{15}$$

This network $r_\theta(t,x)$ can be easily derived by applying automatic differentiation to the network $u_\theta(t,x)$. Then, the shared parameters θ are learned by minimizing the following loss function:

$$\begin{aligned} L(\theta) &= L_u(\theta) + L_r(\theta), \\ L_u(\theta) &= \frac{1}{N_u} \sum_{i=1}^{N_u} |u_\theta(t_u^i, x_u^i) - u_{bc}^i|^2, \\ L_r(\theta) &= \frac{1}{N_r} \sum_{i=1}^{N_r} |r_\theta(t_r^i, x_r^i)|^2, \end{aligned} \tag{16}$$

where $\{(t_u^i, x_u^i), u_{bc}^i\}_{i=1}^{N_u}$ represent the training data on initial and boundary conditions, and $\{t_r^i, x_r^i\}_{i=1}^{N_r}$ denote the collocation points for the PDE residual, $r(t,x)$, sampled randomly throughout the domain of interest. Thus, the loss function consists of two terms: one is the mean squared error coming from the initial and boundary conditions, and the other is the mean squared error from the residual evaluated at collocation points inside the physical domain.

4 Numerical results

In our examples, we consider the nonlinear hyperbolic transport equation of the form:

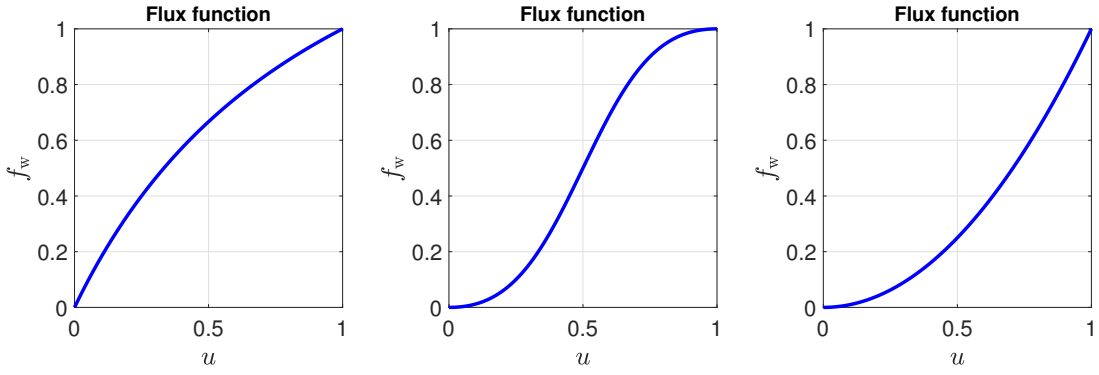
$$u_t + (f_{\mathrm{w}})_x = 0, \tag{17}$$

where $f_{\mathrm{w}} = f_{\mathrm{w}}(u)$ is the fractional flow function, i.e., flux function, and $x \in [0,1], t \in [0,1]$. The unknown solution u corresponds to water saturation, S_{w}, in Equation (6). Different flux functions produce different types of waves in the solution. In addition, we assume the following uniform initial and boundary conditions:

$$\begin{aligned} u(x,t) &= 0, \quad \forall x \text{ and } t = 0, \\ u(x,t) &= 1, \quad x = 0 \text{ and } t > 0. \end{aligned} \tag{18}$$

This setting corresponds to the injection of water at one end of the oil-filled 1-D reservoir, e.g., rock core, and the following parameters: $s_{\mathrm{wi}} = 0$, $s_{\mathrm{b}} = 1$. The conservation law (17) with initial and boundary conditions (18) forms a Riemann problem that has a self-similar solution, i.e., $u(x,t) = u(\frac{x}{t})$.

In the numerical examples, we use the fully-connected neural network architecture reported in (Raissi et al., 2019) that consists of eight hidden layers with 20 neurons per hidden layer. The hyperbolic tangent activation function is used in all hidden layers. The weights are initialized randomly according to the Xavier initialization scheme (Glorot and Bengio, 2010). The loss function is optimized with a second-order quasi-Newton method, L-BFGS-B (Nocedal and Wright, 2006). For the training data in all examples we use $N_u = 300$ randomly distributed points on initial and boundary conditions, and $N_r = 10,000$ collocation points for the residual term, sampled randomly over the interior of the domain $x \in [0,1], t \in [0,1]$. Next, we consider different flux functions $f_{\mathrm{w}}(u)$ in Equation (17).



(a) Concave (b) Non-convex (c) Convex

Figure 1: Different flux functions.

4.1 Concave flux function

If the relative phase permeabilities, $k_{r\alpha}(S_\alpha)$, are linear functions of saturation, and the ratio of the phase viscosities is denoted as $\frac{\mu_o}{\mu_w} = M$, the corresponding flux function, f_w, can be written as:

$$f_w(u) = \frac{u}{u + \frac{1-u}{M}}. \tag{19}$$

For $M > 1$, this flux function is concave, as shown in Fig. 1a for $M = 2$. The solution of Equation (17) with the flux function (19) is a rarefaction (spreading) wave:

$$u(x,t) = \begin{cases} 0, & \frac{x}{t} > M \\ \frac{\sqrt{M\frac{t}{x}}-1}{M-1}, & M \geq \frac{x}{t} \geq \frac{1}{M} \\ 1, & \frac{1}{M} \geq \frac{x}{t} \end{cases}$$

We consider the case $M = 2$. Due to the piecewise nature of the analytical solution, there are certain locations (specifically, those along the lines $\frac{x}{t} = M$ and $\frac{x}{t} = \frac{1}{M}$), where the solution is non-differentiable as derivatives of the solution are different on both sides.

However, this does not prevent the deep learning approach from learning the solution. Figure 2 presents a comparison between the exact analytical and the predicted by neural network solutions at time instances $t = 0.25, 0.5, 0.75$. In this case, the neural network produces accurate estimates of the PDE solution with some smoothing of the non-differentiable edges of the solution. The final loss at the end of training is $L(\theta) = 1.2 \cdot 10^{-3}$ and the resulting relative $\mathcal{L}^2$ norm of the prediction error of the solution (compared to the analytical solution) is $2.6 \cdot 10^{-2}$.

4.2 Non-convex flux function

In most practical settings, the interaction between two immiscible fluids flowing through the porous medium leads to highly nonlinear relative permeabilities. A simple model that captures this characteristic is the Brooks-Corey model (Brooks and Corey, 1964), which gives the power-law relationship between the relative permeability of a fluid phase and its saturation. Specifically, we use a quadratic relationship, which leads to the following flux, i.e., fractional flow, function:

$$f_w(u) = \frac{u^2}{u^2 + \frac{(1-u)^2}{M}}, \tag{20}$$

where again M is the ratio of phase viscosities. The PDE (17) with this non-convex flux function constitutes a standard Buckley-Leverett problem in porous media flow. In our example we use $M = 1$ and the corresponding flux function is depicted in Fig. 1b. We proceed by considering two cases with this flux function – with and without an additional diffusion term in the PDE.

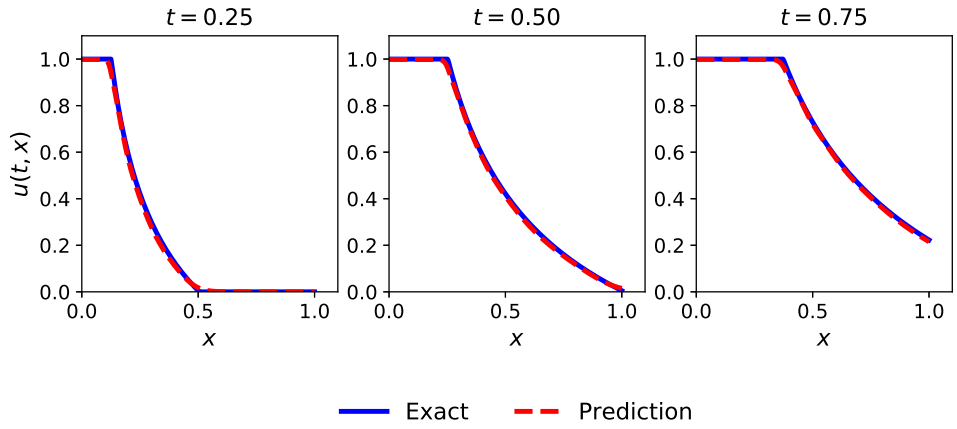


Figure 2: Comparison of the predicted by the neural network and the exact solutions of the PDE (17) with concave flux function (19), corresponding to the three different times $t = 0.25, 0.5, 0.75$.

4.2.1 Without diffusion term

In this case, the residual term (17), representing the hyperbolic PDE, is used directly in the loss function. The analytical solution to this problem contains a shock and a rarefaction wave and is constructed as follows:

$$u\left(x,t\right) = \begin{cases} 0, & \frac{x}{t} > f'_{\mathrm{w}}(u^*) \\ u(\frac{x}{t}), & f'_{\mathrm{w}}(u^*) \geq \frac{x}{t} \geq f'_{\mathrm{w}}(u=1), \\ 1, & f'_{\mathrm{w}}(u=1) \geq \frac{x}{t} \end{cases} \tag{21}$$

where u^* denotes the shock location, which is defined by the Rankine-Hugoniot condition $f'_{\mathrm{w}}(u^*) = \frac{f_{\mathrm{w}}(u^*) - f_{\mathrm{w}}(u)_{|u=0}}{u^* - u_{|u=0}}$, and $u(\frac{x}{t})$ is defined for $\frac{x}{t} \leq f'_{\mathrm{w}}(u^*)$ as $u(\frac{x}{t}) = (f'_{\mathrm{w}})^{-1}(\frac{x}{t})$. Due to the self-similarity, the analytical solution (21) has just one governing parameter – the similarity variable $\frac{x}{t}$.

Figure 3 shows that the neural network fails in this case to provide an accurate approximation of the underlying analytical solution (21). In fact, the neural network completely misses the correct location of the saturation front, which leads to high values of the loss (at the end of training it is $L(\theta) = 0.036$) and large prediction errors. In our numerical experiments, we observed that changing the neural network architecture and/or increasing the number of collocation points had little impact on the results (details of these studies are provided in the Appendix A). Thus, we think this phenomenon is not related to the choice of network architecture or its hyperparameters.

4.2.2 With diffusion term

The vanishing viscosity method for solving the initial value problems for hyperbolic PDEs (Crandall and Lions, 1983; Lax, 2006) is based on the fact that solutions of the inviscid equations, e.g., (17), including solutions with shocks, are the limits of the solutions of the viscous equations as the coefficient of viscosity tends to zero. Motivated by this approach, we add a second-order term, i.e., a diffusion term, to the right-hand side of (17) and consider the following equation:

$$u_t + f'_{\mathrm{w}}(u)u_x = \epsilon u_{xx}, \tag{22}$$

where $\epsilon > 0$ is a scalar diffusion coefficient that represents the inverse of the Péclet number, Pe – the ratio of a characteristic time for dispersion to a characteristic time for convection. When ϵ is small, i.e., Péclet number is large, the effects of diffusion are negligible and convection dominates. Letting $\epsilon \to 0$ in Equation (22) defines a vanishing diffusion solution of Equation (17), which is the one with the correct physical behavior. Also, it should be noted that Equation (22) is now a parabolic PDE, so its solution is smooth, i.e., it does not contain shocks.

Figure 4 shows neural network solutions for two different values of diffusion coefficient ϵ: $1 \cdot 10^{-2}$ ($\mathrm{Pe} = 100$) and $2.5 \cdot 10^{-3}$ ($\mathrm{Pe} = 400$). The loss values at the end of the training are $L(\theta) = 3.2 \cdot 10^{-6}$ and $2.4 \cdot 10^{-5}$, respectively.

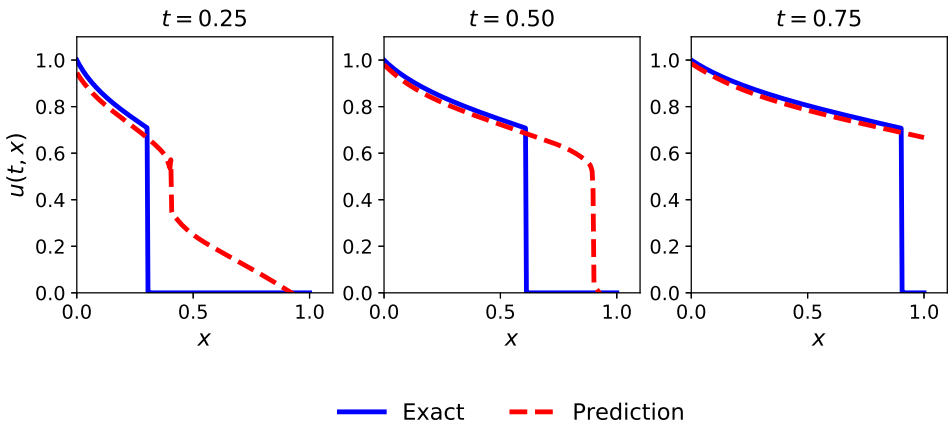


Figure 3: Comparison of the predicted by the neural network and the exact solutions of the PDE (17) with non-convex flux function (20), corresponding to the three different times $t = 0.25, 0.5, 0.75$.

Note that the loss function is different in these two cases as the loss depends on the PDE residual, which is a function of ϵ according to Equation (22). From these results, we see that adding a diffusion term to the conservation equation allows the neural network to perfectly capture the location of the saturation front even for quite small ϵ. Indeed, the solution in Fig. 4b for $\epsilon = 2.5 \cdot 10^{-3}$ is almost indistinguishable from the underlying analytical PDE solution – there is just a slight smoothing of the shock. In our numerical experiments, we also observed that if we continue to decrease the value of diffusion coefficient ϵ, e.g., $\epsilon = 1 \cdot 10^{-3}$, then the diffusion effects become too small, and the behavior of the neural network is the same as in the hyperbolic setting (i.e., zero diffusion) described in Section 4.2.1. It should be noted that the experiments in the current section – both for PDEs with and without the diffusion term – were all performed multiple times with different random seeds and random initializations, however, the results in terms of recovering the shock were equivalent.

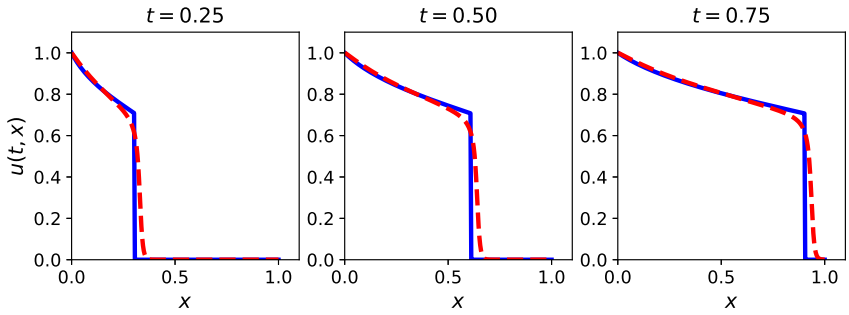
Then, we conducted similar experiments for other values of phase viscosity ratio M, such as 0.5, 5, and 10, which are also common in the subsurface transport domain. Under these settings the solutions differ in size and speed of the shock, i.e., for larger M the shock size decreases but its speed increases. However, the solution structure stays the same – the solution still consists of a shock followed by a rarefaction wave. We considered both cases for each value of the phase viscosity ratio M – with and without the diffusion term. The results of these tests and conclusions were the same as for $M = 1$ described above, thus we conclude that the observed behavior of the PIML approach is not sensitive to the value of parameter M.

It is worth mentioning that the obtained results are consistent with the previously reported results of the PIML approach in (Raissi et al., 2017). The authors of (Raissi et al., 2017) studied Burgers' equation with the diffusion term (so the shock was smoothed) and the diffusion coefficient (ϵ in our notation) was equal to $\epsilon = \frac{0.01}{\pi} \approx 3.2 \cdot 10^{-3}$. However, if one applies the PIML approach for the same settings of Burgers' equation as in (Raissi et al., 2017) but decreases the diffusion coefficient to $0.5 \cdot 10^{-3}$ or less (or sets it to zero altogether), then the network fails similarly as was described in Section 4.2.1.

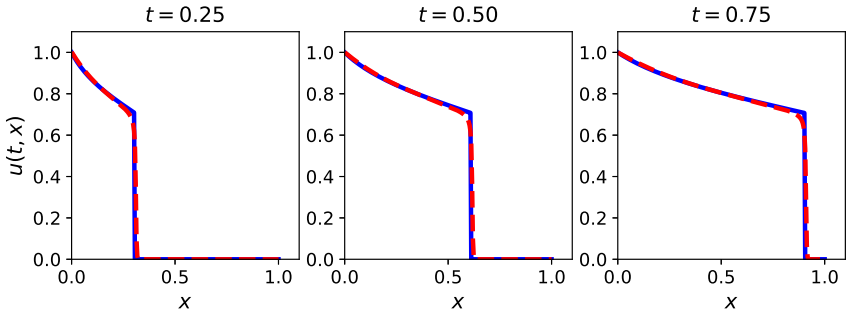
4.3 Convex flux function

Now, we move to the convex flux function, shown in Fig. 1c, which is simply a quadratic function $f_{\text{w}}(u) = u^2$. The solution is a self-sharpening wave, propagating as a shock with unit speed.

The prediction of the neural network for $t = 0.5$ in case of hyperbolic PDE (17) is shown in the left plot of Fig. 5. As in the case of the non-convex flux function, the PIML approach fails for this problem. And similar to the non-convex flux case, adding a small diffusion term, i.e., $\epsilon = 2.5 \cdot 10^{-3}$, to the PDE allows the neural network to reconstruct the solution and determine the location of the (smoothed) shock correctly (Fig. 5 on the right).



(a) $\epsilon = 1 \cdot 10^{-2}$



(b) $\epsilon = 2.5 \cdot 10^{-3}$

Figure 4: Predictions of the neural network for the PDE (22) for different values of diffusion coefficient ϵ. Exact solution corresponds to the PDE (17) without diffusion term.

5 Analysis

It is quite surprising that the neural network with several thousands of parameters is not able to yield a reasonable approximation to the analytical solution of the 1D hyperbolic PDE (17) with a non-convex flux function (20) – the solution that can be represented using a relatively simple piecewise continuous function of one parameter (21). Especially, because according to the universal approximation theorem (Cybenko, 1989) there should exist a network that can provide a close approximation of the continuous solution of (22) for any arbitrarily small ϵ (because the solution is smooth in this case), however, this is not what is observed in practice. Thus, this leads us to the conclusion that the problem is not with the solution itself, but rather with *how* we attempt to find this solution, i.e., with the optimization process, or the loss function.

For the examples described above, we provide an analysis of the obtained neural networks. Our aim here is to get a better understanding of the observed behavior of neural network approach – why it can find a solution to the problem with the additional diffusion term, i.e., the parabolic form of the PDE, but fails to do so in the case of the underlying hyperbolic PDE, i.e., when its solution contains a discontinuity. Is this due to some fundamental reasons that prevent the neural network from finding a reasonable approximate solution (non-uniqueness of the solution of the weak form), or is it because the employed optimization algorithm just cannot reach the solution? The latter can be due to the complicated nature of the non-convex landscape of the loss function, or other inherent limitations of the optimization algorithm.

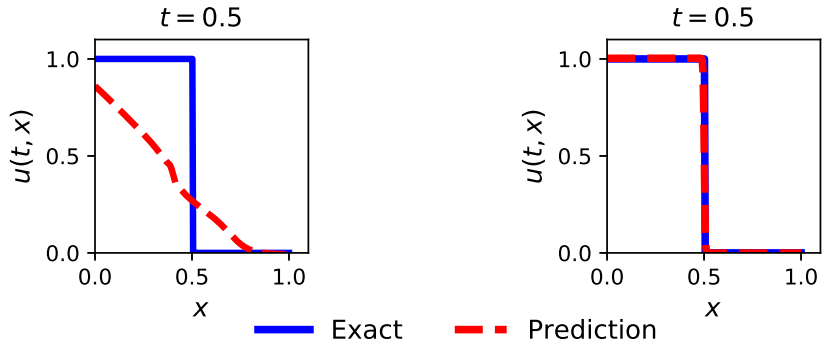


Figure 5: Predictions of the neural network at $t = 0.5$ for the case of convex flux function: on the left the prediction for the PDE (17) without diffusion term, on the right – with added diffusion term, as in Equation (22), and diffusion coefficient $\epsilon = 2.5 \cdot 10^{-3}$. Exact solutions in both cases are shown for the PDE (17) without diffusion term.

First, we investigate the training process and study the behavior of the loss and its gradients with respect to the network parameters. Then, through 2-D visualizations of the loss surface, we study how the diffusion term affects the loss landscape and the convexity of the loss near the final optimization point, i.e., a optimized set of network parameters.

5.1 Training process

Figure 6 shows the evolution of the loss function during the training process for models with different amounts of diffusion, i.e., different values of the diffusion coefficient ϵ before the second-order term in Equation (22). The x-axis in the figure denotes the steps of the L-BFGS-B optimization method. Note that the loss function being minimized is different for each model as part of the loss, corresponding to the residual term, is directly proportional to ϵ. In Fig. 6 we observe a clear trend. For larger values of ϵ the convergence rate of the optimization improves significantly, i.e., the loss is minimized in far fewer steps. On the other hand, for smaller values (i.e., $\epsilon = 0$ or $1 \cdot 10^{-3}$) the corresponding loss curve flattens out quite early during the training, and the optimization method fails to minimize the loss (the final loss is only of order 10^{-2}).

The training of the neural network can be also studied by observing the gradient of the loss with respect to the different parameters of the network, i.e., weights and biases of different layers. Figure 7 shows the $\mathcal{L}^2$ norm of the loss gradient with respect to the weights in the first layer versus the number of optimization steps (some curves were smoothed for better visualization). The curves for the models that achieve good approximation accuracy of the solution, i.e., the models with $\epsilon = 5 \cdot 10^{-2}$, $5 \cdot 10^{-3}$ and $2.5 \cdot 10^{-3}$, show a steady decrease in the norm of the gradient during training, indicating convergence of the optimization process; on the other hand, for the models that have large prediction errors, i.e., for $\epsilon = 0$ and $\epsilon = 1 \cdot 10^{-3}$, the gradients do not decrease with time, and sometimes even increase, indicating failure of the optimization process. The loss gradients with respect to the parameters in other layers of the network showed similar trends. Again, from the results shown in Fig. 7 it is obvious that the magnitude of ϵ significantly affects the behavior of the loss gradients. This behavior for $\epsilon \sim 0$ may be explained with the complicated objective function landscape, so that the quasi-Newton method fails to minimize the loss. It may be also due to the poor conditioning of the Hessian of the loss so that the desired solution lies in a very local and narrow region. Nevertheless, it is clear that the presence of the second-order term u_{xx}, i.e., presence of diffusion in the PDE, and the amount of diffusion strongly influence the training process of the physics informed network and its ability to yield accurate approximations of the solution.

5.2 Loss landscape

To visualize the surface of the loss, which is a function in the high-dimensional parameter space, one must restrict the space to a low-dimensional one (1-D or 2-D), amenable to visualization. Here, we choose to follow the approach of (Li et al., 2018), whereby to get a 2-D projection of the loss surface we choose a center point θ, corresponding to the final optimization point (i.e., final parameters of the model reshaped into a single vector) and two direction vectors, δ and η,

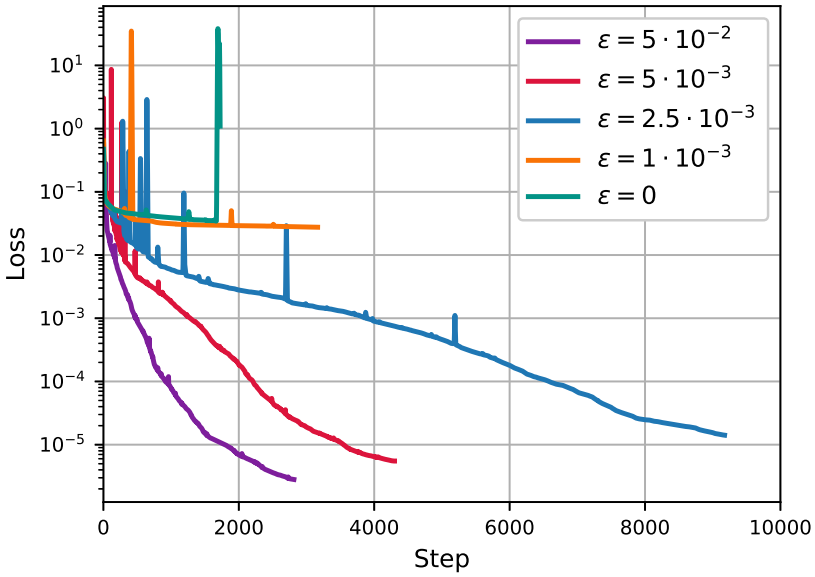


Figure 6: The loss function during training for models with different amount of added diffusion according to the Equation (22). The x-axis denotes the steps of the L-BFGS-B optimization method.

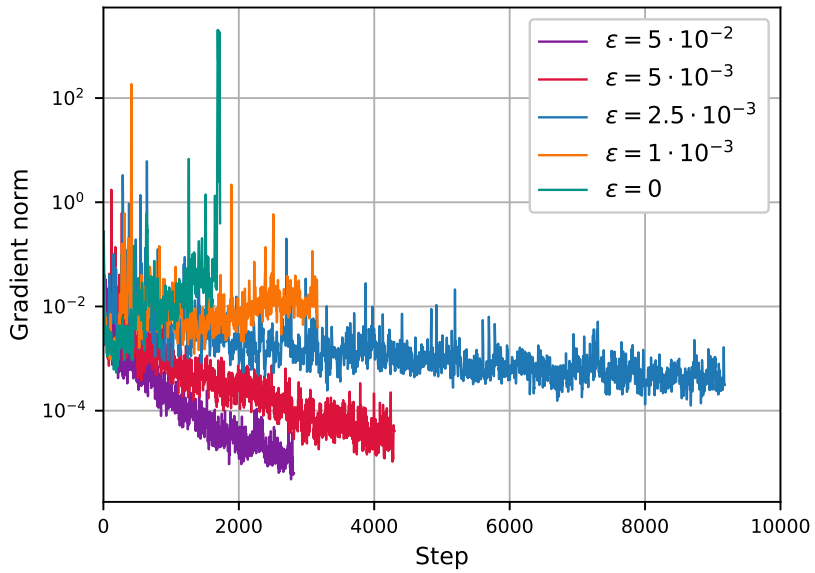
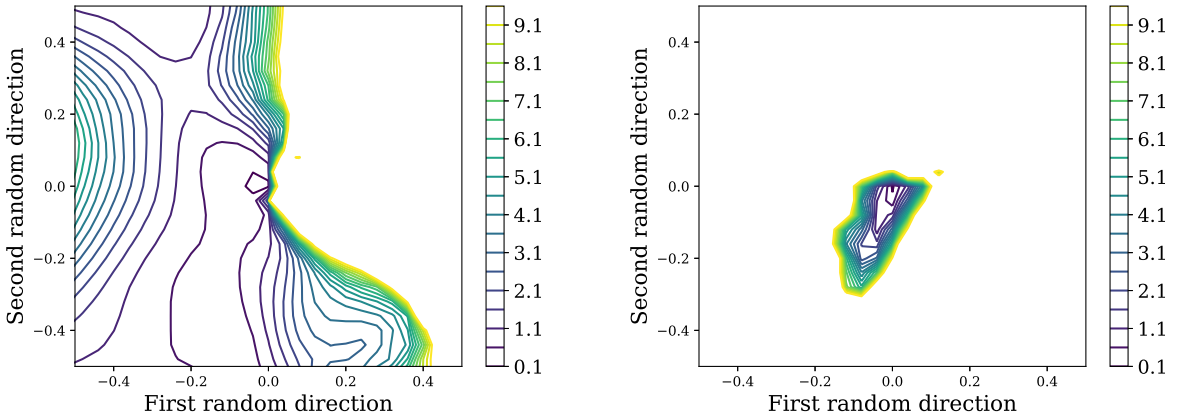
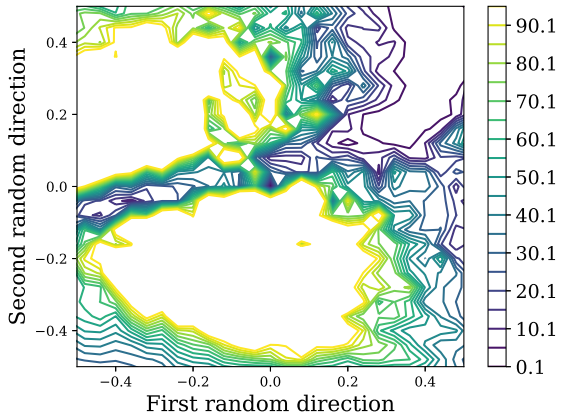


Figure 7: The evolution of $\mathcal{L}^2$ norm of the loss gradient with respect to the weights in the first layer of the network during training. The x-axis denotes the steps of the L-BFGS-B optimization method.



(a) $\epsilon = 5 \cdot 10^{-2}$ (b) $\epsilon = 2.5 \cdot 10^{-3}$



(c) $\epsilon = 0$

Figure 8: 2-D visualizations of the loss surface near final optimization point for neural networks trained with different values of diffusion coefficient ϵ in Equation (22). Note the change of scale for $\epsilon = 0$.

of the same dimension as θ. Then, we can plot the following function

$$f(\alpha, \beta) = L(\theta + \alpha\delta + \beta\eta), \tag{23}$$

where α and β are scalar parameters along vectors δ and η, respectively. The direction vectors are sampled randomly from Gaussian distribution – in the high-dimensional space these vectors with a high probability will be almost orthogonal to each other. Then, (Li et al., 2018) suggests "filter-wise" normalizing the random directions to capture the natural distance scale of the loss surface. This step ensures that elements in random vectors, δ and η, are of the same scale as the corresponding parameters of the network, i.e., weights and biases in different network layers.

For visualizations we vary both scalar parameters, α and β, in the range $(-0.5, 0.5)$. Figure 8 shows the loss surface plots for different networks near their final optimization point, i.e., set of optimized parameters. This point corresponds to $(0, 0)$ in these surface plots, and the two axes represent the two random directions, respectively. The results are shown as contour plots to make it easier to see the non-convex structures of the loss landscape. The networks differ in the amount of the added diffusion, i.e., the value of diffusion coefficient ϵ. For large diffusion, for example $\epsilon = 5 \cdot 10^{-2}$, in Fig. 8a, we observe quite large convex region, whereas for a small amount of diffusion, e.g., $\epsilon = 2.5 \cdot 10^{-3}$, this region shrinks significantly, as shown in Fig. 8b. Note the change of scale in Fig. 8c, which depicts the loss surface for the hyperbolic PDE, i.e., $\epsilon = 0$, – for proper visualization the scale of the loss had to be increased by 10 times compared to the cases with diffusion. No convex region is observed in this instance. Moreover, the loss landscape is not as smooth

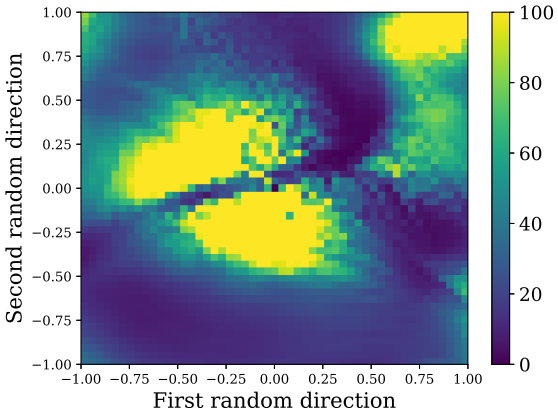


Figure 9: 2-D visualization of the loss surface of the neural network for the hyperbolic PDE (17) with non-convex flux function (20).

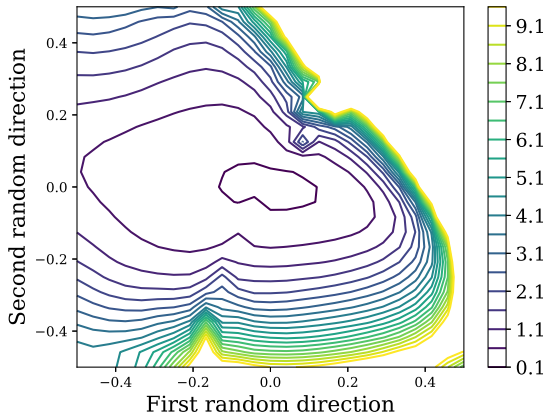


Figure 10: 2-D visualization of the loss surface of the neural network for the hyperbolic PDE (17) with concave flux function (19) (the PDE solution is smooth in this case).

as with the diffusion present – indeed, it has a lot of chaotic features, as can be seen in Fig. 8c. For visualization of the same loss surface on a larger slice of the parameter space, refer to Fig. 9. From these observations, we can conclude that the presence of the discontinuity, i.e., the shock, in the PDE solution strongly affects the properties of the resulting landscape of the corresponding loss function – specifically, its smoothness and convexity. It is not surprising that the optimization procedure struggles with this loss landscape and is unable to reach the proper solution, i.e., the one that gives a close continuous approximation of the discontinuous PDE solution (21). For comparison, we also show in Fig. 10 the loss surface of the network approximating a smooth PDE solution in case of concave flux function (19). The wide convex region of the loss surface is evident here.

6 Discussion and conclusion

We investigated the application of a physics informed machine learning (PIML) approach to the solution of one-dimensional hyperbolic PDEs that describe the nonlinear two-phase transport in porous media. The PIML approach encodes the underlying PDE into the loss function and learns the solution to the PDE without any labeled data – only using the knowledge of the initial/boundary conditions and the PDE. Our experiments with different flux functions demonstrate that the neural network approach provides accurate estimates of the solution of the hyperbolic PDE when the solution does not contain discontinuities. However, the PIML approach fails to provide a reasonable approximate

solution of the PDE when shocks are present. We found that it is necessary to add a diffusion term to the underlying PDE so that the network can recover the proper location and size of the shock, which is smoothed by diffusion. Thus, the network solves the parabolic form of the conservation equation, which leads to the correct solution with smoothing around the shock. It is interesting to note here the resemblance of this effect with finite-volume methods, whereby the conservative finite-volume discretization adds a numerical diffusion term, and as a result, the numerical solution corresponds to a parabolic equation with a finite amount of diffusion. The diffusion term can be controlled through refinement in space-time and by the use of higher-order discretization schemes.

Then, we analyzed the network training process for cases with and without diffusion in the PDE. Our study shows that the amount of added diffusion strongly affects the training of the network (e.g., the convergence rate, the behavior of the loss gradients). Moreover, we provided 2-D visualizations of the loss landscape of the neural networks near their final optimization point, that indicate that the diffusion term in the PDE smooths the loss surface and makes it more convex, while the loss surface of the hyperbolic PDE with discontinuous solution demonstrates significant chaotic and non-convex features. However, the reasons for such behavior of the loss function are not perfectly understood yet. It would be certainly interesting to derive some analytical explanation of the observed phenomena as well. Nevertheless, through the experiments and analysis conducted in the current work we show that the physics informed machine learning framework is not suited for the hyperbolic PDEs with discontinuous solutions considered here.

Acknowledgements

We thank Total for their financial support of our research on "Uncertainty Quantification". The authors are also grateful to the Stanford University Petroleum Research Institute for Reservoir Simulation (SUPRI-B) for financial support of this work.

References

K. Aziz and A. Settari. *Petroleum reservoir simulation*. Applied Science Publishers, 1979.

R. Brooks and T. Corey. Hydraulic properties of porous media. *Hydrology Papers, Colorado State University*, 24, 1964.

M. G. Crandall and P. L. Lions. Viscosity solutions of Hamilton-Jacobi equations. *Transactions of the American mathematical society*, 277(1):1–42, 1983.

G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989.

O. Fuks and H. A. Tchelepi. Limitations of physics informed machine learning for nonlinear two-phase transport in porous media. *Journal of Machine Learning for Modeling and Computing*, 1(1):19–37, 2020. ISSN 2689-3967.

X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proc. of the 13th International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010.

V. Gulshan, L. Peng, M. Coram, et al. Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs. *JAMA*, 316(22):2402–2410, 2016.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

G. Hinton, L. Deng, D. Yu, et al. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*, 29:82–97, 2012.

K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

A. Karpathy, G. Toderici, S. Shetty, et al. Large-scale video classification with convolutional neural networks. In *Proc. of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, pages 1725–1732, 2014.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 25, pages 1097–1105, 2012.

I. E. Lagaris, A. Likas, and D. I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *Trans. Neur. Netw.*, 9(5):987–1000, Sept. 1998. ISSN 1045-9227.

P. Lax. *Hyperbolic System of Conservation Laws and the Mathematical Theory of Shock Waves*, pages 1–48. Society for Industrial and Applied Mathematics, 1973.

P. D. Lax. *Hyperbolic partial differential equations*, volume 14. American Mathematical Soc., 2006.

Y. LeCun, Y. Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521:436–44, 2015.

H. Lee and I. S. Kang. Neural algorithm for solving differential equations. *Journal of Computational Physics*, 91(1): 110–131, 1990.

H. Li, Z. Xu, G. Taylor, et al. Visualizing the loss landscape of neural nets. In *Proc. of the 32nd International Conference on Neural Information Processing Systems*, page 6391–6401, 2018.

T. P. Lillicrap, J. . Hunt, A. Pritzel, et al. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Y. Liu, K. K. Gadepalli, M. Norouzi, et al. Detecting cancer metastases on gigapixel pathology images. Technical report, arXiv, 2017. URL https://arxiv.org/abs/1703.02442.

A. J. Meade, Jr. and A. A. Fernandez. The numerical solution of linear ordinary differential equations by feedforward neural networks. *Mathematical and Computer Modelling*, 19(12):1–25, 1994.

V. Mnih, A. P. Badia, M. Mirza, et al. Asynchronous methods for deep reinforcement learning. In *Proc. of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 1928–1937, 2016.

J. Nocedal and S. J. Wright. *Numerical Optimization*, pages 176–180. Springer, New York, NY, USA, second edition, 2006.

F. Orr. *Theory of gas injection processes*. Tie-Line Publications, 2007.

D. C. Psichogios and L. H. Ungar. A hybrid neural network-first principles approach to process modeling. *AIChE Journal*, 38(10):1499–1511, 1992. doi: 10.1002/aic.690381003.

M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics informed deep learning (Part I): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017.

M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

R. Stewart and S. Ermon. Label-free supervision of neural networks with physics and domain knowledge. In *Proc. of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, pages 2576–2582. AAAI Press, 2017.

I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Proc. of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3104–3112, 2014.

Y. Zhu, N. Zabaras, P.-S. Koutsourelakis, and P. Perdikaris. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *Journal of Computational Physics*, 394: 56–81, 2019.

A Sensitivity study for the Buckley-Leverett problem

For the case described in Section 4.2.1, we perform a sensitivity study. Our aim here is to understand whether the result obtained in Section 4.2.1 for the Buckley-Leverett problem (nonlinear transport with a non-convex flux function) is strongly dependent on the particular choice of the network architecture and the different hyperparameters of the method, such as the number of training data points on initial and boundary conditions N_u and the number of collocation points N_r in the interior of the domain.

First, we fix the network architecture to eight hidden layers with 20 neurons per hidden layer, and we vary the number of initial and boundary training data N_u in the range $(100, 600)$ and the number of collocation points N_r in the range $(1000, 20000)$. The final values of the loss function at the end of the training for these experiments are shown in Table 1. In all these cases, the network failed to yield a reasonable approximation of the shock; as the result, we observe a relatively large value of the loss function (i.e., $\sim 10^{-2}$). From Table 1, it is also clear that the network performance is not a strong function of the number of initial and boundary training data points and the number of collocation points.

In the next experimental set, we kept the total number of training and collocation points fixed to $N_u = 300$ and $N_r = 10,000$, and we varied the number of hidden layers in the range $(2, 12)$ and the number of neurons per hidden layer in the range $(10, 40)$. With these ranges, the total number of network parameters varied from 151 to over $18,000$. Table 2 reports the value of the loss function at the end of the training for these different architectures. Again, the observed trend is quite consistent – the final result is not weakly sensitive to the particular network architecture.

Moreover, the PDE solutions $u(t, x)$ predicted by the neural networks in all these cases were quite similar to the ones reported in Section 4.2.1, where the network completely fails to approximate the shock.

In addition, we experimented with the application of standard regularization to the network weights – the technique typically used in machine learning to decrease overfitting. Specifically, we added to the loss function $L(\theta)$ a regularization term of the form $l_{\text{reg}} = \beta W^T W$ (where W denotes the weights of the network) and considered a range of regularization constants $\beta = [1 \cdot 10^{-5}, 5 \cdot 10^{-5}, 1 \cdot 10^{-4}, 5 \cdot 10^{-4}, 1 \cdot 10^{-3}]$. However, in these experiments we did not see any improvement in the PIML results for hyperbolic PDE.

Next, we also tested the PIML approach with different types of networks – a residual network architecture (He et al., 2016) and a convolutional neural network (CNN) (LeCun et al., 1995). For the residual network, we added skip connections after each layer in the original fully-connected architecture. With CNN architecture we used eight convolutional layers with 20 filters each, that perform 1-D convolutions and have a kernel size of 1×1 (in this case, the number of parameters is the same as in the standard fully-connected architecture reported in the paper). In these experiments, we observed a similar behavior – that the PIML approach fails for hyperbolic PDE but performs well for PDE with added diffusion term.

Table 1: Final loss at the end of training for different number of initial and boundary training data points N_u and different number of collocation points N_r. The network architecture is fixed to eight hidden layers with 20 neurons per hidden layer.

N_u \ N_r	$1,000$	$5,000$	$10,000$	$20,000$
100	$1.6 \cdot 10^{-2}$	$3.4 \cdot 10^{-2}$	$3.0 \cdot 10^{-2}$	$2.6 \cdot 10^{-2}$
300	$2.2 \cdot 10^{-2}$	$2.6 \cdot 10^{-2}$	$3.4 \cdot 10^{-2}$	$3.2 \cdot 10^{-2}$
600	$1.3 \cdot 10^{-2}$	$2.0 \cdot 10^{-2}$	$3.1 \cdot 10^{-2}$	$3.0 \cdot 10^{-2}$

Table 2: Final loss at the end of training for different number of hidden layers and different number of neurons per hidden layer. The total number of training and collocation points is fixed to $N_u = 300$ and $N_r = 10,000$.

Layers \ Neurons	10	20	40
2	$3.4 \cdot 10^{-2}$	$3.2 \cdot 10^{-2}$	$3.2 \cdot 10^{-2}$
4	$1.6 \cdot 10^{-2}$	$3.2 \cdot 10^{-2}$	$3.1 \cdot 10^{-2}$
8	$3.3 \cdot 10^{-2}$	$3.4 \cdot 10^{-2}$	$3.3 \cdot 10^{-2}$
12	$3.5 \cdot 10^{-2}$	$2.9 \cdot 10^{-2}$	$1.9 \cdot 10^{-2}$