




OPEN

## Physics-informed attention-based neural network for hyperbolic partial differential equations: application to the Buckley–Leverett problem

Ruben Rodriguez-Torrado<sup>1,2,4</sup>, Pablo Ruiz<sup>1</sup>, Luis Cueto-Felgueroso<sup>2</sup>, Michael Cerny Green<sup>1</sup>, Tyler Friesen<sup>1</sup>, Sebastien Matringe<sup>3</sup> & Julian Togelius<sup>1,4</sup>

Physics-informed neural networks (PINNs) have enabled significant improvements in modelling physical processes described by partial differential equations (PDEs) and are in principle capable of modeling a large variety of differential equations. PINNs are based on simple architectures, and learn the behavior of complex physical systems by optimizing the network parameters to minimize the residual of the underlying PDE. Current network architectures share some of the limitations of classical numerical discretization schemes when applied to non-linear differential equations in continuum mechanics. A paradigmatic example is the solution of hyperbolic conservation laws that develop highly localized nonlinear shock waves. Learning solutions of PDEs with dominant hyperbolic character is a challenge for current PINN approaches, which rely, like most grid-based numerical schemes, on adding artificial dissipation. Here, we address the fundamental question of which network architectures are best suited to learn the complex behavior of non-linear PDEs. We focus on network architecture rather than on residual regularization. Our new methodology, called physics-informed attention-based neural networks (PIANNs), is a combination of recurrent neural networks and attention mechanisms. The attention mechanism adapts the behavior of the deep neural network to the non-linear features of the solution, and break the current limitations of PINNs. We find that PIANNs effectively capture the shock front in a hyperbolic model problem, and are capable of providing high-quality solutions inside the convex hull of the training set.

Deep neural networks (DNNs) have achieved enormous success in recent years because they have significantly expanded the scope of possible tasks that they can perform, given sufficiently large datasets<sup>1</sup>. The range of applications is extraordinary, from natural language processing<sup>2,3</sup>, image analysis<sup>4</sup> and autonomous driving<sup>5</sup>, to earthquake forecasting<sup>6</sup>, hydrological modeling<sup>7,8</sup>, playing videogames<sup>9,10</sup>, generating game content<sup>11</sup>, and, more recently, numerical differentiation<sup>12</sup> and scientific computing<sup>13,14</sup>.

Neural networks can approximate the solution of differential equations<sup>15,16</sup>, in particular high-dimensional partial differential equations (PDEs)<sup>17,18</sup>. One of the most promising approaches to efficiently solve non-linear PDEs is physics-informed neural networks (PINNs)<sup>19–21</sup>. PINNs are trained to solve supervised learning tasks constrained by PDEs, such as the conservation laws in continuum theories of fluid and solid mechanics<sup>16,22–24</sup>. In addition to fluid and solid mechanics, PINNs have been used to solve a big amount of applications governed by differential equations such as radioactive transfer<sup>25,26</sup>, gas dynamics<sup>27,28</sup>, water dynamics<sup>29</sup>, Euler equation<sup>30,31</sup>, numerical integration<sup>32</sup>, chemical kinetics<sup>33,34</sup> and optimal control<sup>35,36</sup>.

The idea behind PINNs is to train the network using automatic differentiation (AD) by calculating and minimizing the residual, usually constrained by initial and boundary conditions, and possibly observed data<sup>19</sup>. PINNs have the potential to serve as on-demand, efficient simulators for physical processes described by differential equations (the *forward* problem)<sup>19,24,37</sup>. Once they have been trained, PINNs can potentially run faster and yield better predictions than standard numerical simulators of complex real-world phenomena<sup>21</sup>. PINNs may also

<sup>1</sup>OriGen.AI, New York, USA. <sup>2</sup>Universidad Politécnica de Madrid, Madrid, Spain. <sup>3</sup>Hess Corporation, Houston, TX, USA. <sup>4</sup>New York University, New York, USA. ✉email: rubentorrado@origen.ai

be used to assimilate data and observations into numerical models, or be used in parameter identification (the *inverse problem*)<sup>19,38</sup> and uncertainty quantification<sup>39–42</sup>.

Learning solutions of nonlinear PDEs using current network architectures presents some of the same limitations of classical numerical discretization schemes. For instance, one of the main limitations of classical numerical methods (e.g. finite differences, volumes, elements) is the need to devise suitable *upwind* discretizations that yield smooth and accurate solutions near the shock fronts. Methods that use *centered* approximations, without numerical dissipation, such as standard finite differences or the Galerkin finite element method, lead to solutions that are polluted by spurious oscillations around the shock front, often leading to numerical instabilities. PINN might present similar issues if the same schemes are used to calculate residuals.

A paradigmatic example is the solution of hyperbolic PDEs. Hyperbolic conservation laws describe a plethora of physical systems in gas dynamics, acoustics, elastodynamics, optics, geophysics, and biomechanics<sup>43</sup>. Hyperbolic PDEs are challenging to solve numerically using classical discretization schemes, because they tend to form self-sharpening, highly-localized, nonlinear shock waves that require specific approximation strategies and fine meshes<sup>44</sup>. Solving hyperbolic PDEs seems to be challenging for neural networks as well, as the ability of current PINNs to learn PDEs with a dominant hyperbolic character relies on adding artificial dissipation<sup>45–48</sup>, or on using a priori knowledge to increase the number of training points along the shock trajectories<sup>39</sup> or adaptive activation functions<sup>49,50</sup>.

In this work, we propose a new perspective on solving hyperbolic PDEs and traditional limitations of classical numerical methods using deep learning. The proposed method relies in two core ideas: (1) A modified PINN architecture can provide a more general method for solving hyperbolic conservation law problems without a priori knowledge or residual regularization. (2) Relating network architecture with the physics encapsulated in a given PDE is possible and has a beneficial impact. Our hypothesis is that sophisticated, physics-specific network architectures (i.e. networks whose internal hierarchy is related to the physical processes being learned) may be more effectively trained and easier understood than standard feed-forward multilayer perceptrons.

The proposed new architecture uses *attention mechanisms* to automatically detect shocks in the solution of hyperbolic PDEs. Our use of attention mechanisms to enrich PINN network architectures is inspired by recent advances in deep learning for language processing and translation<sup>51,52</sup>. The resulting network is a combination of *gated recurrent units (GRUs)* and attention mechanisms; we call this a physics-informed attention-based neural network (PIANN). The combination of both elements in the architecture allows for determination of the most relevant information (recurrent neural network with memory) to adapt the behavior of the deep neural network to approximate sharp shocks without the necessity of residual regularization or a priori knowledge (attention mechanism).

Previous works as<sup>19,46,53</sup>, introduced initial and boundary conditions in the formulation as a penalty term in the objective function. The main drawback of this approach is, if this term was not exactly zero after training, the boundary condition is not completely satisfied. Other authors as Lagaris et al.<sup>54</sup> or more recently, Schiassi et al.<sup>55,56</sup> make use of the Extreme Theory of Functional Connections<sup>57</sup> to enforce the initial and boundary conditions in the solution. As in these works, we also enforce the initial and boundary conditions as hard constraints.

We test our PIANN approach by solving a classical hyperbolic model problem, namely the Buckley–Leverett equation<sup>44,58</sup>. The Buckley–Leverett equation with non-convex flux function is an excellent benchmark to test the overall potential of PIANNs in solving hyperbolic PDEs. We find that PIANNs effectively capture the shock front propagation and are capable of providing high quality solutions for mobility ratios inside the convex hull of training set. Remarkably, PIANNs are able to provide smooth, accurate shock fronts without explicitly introducing additional constraints or dissipation in the residual, through an artificial diffusion term or upwinding the spatial derivatives.

The rest of the paper is organized as follows. The mathematical formulation of the problem, describing incompressible two-phase flow in porous media is introduced in “Section [Problem formulation](#)”. “Section [Methodology](#)” introduces our methodology to address the problem using deep neural networks. The proposed PIANN architecture is described in “Section [PIANN architecture](#)”. Experimental results are reported in “Section [Results](#)” and finally, “Section [Discussion](#)” concludes the paper.

## Problem formulation

The problem of interest is that of two immiscible fluids flowing through a horizontal porous medium. We further assume that the fluids and overall system are incompressible. The Buckley–Leverett (BL) equation<sup>58</sup> describes the evolution in time and space of the wetting-phase (water) saturation. Let  $u_M : \mathbb{R}_0^+ \times \mathbb{R}_0^+ \rightarrow [0, 1]$  be the solution of the BL equation

$$\frac{\partial u_M}{\partial t}(x, t) + \frac{\partial f_M}{\partial x}(x, t) = 0, \quad (1)$$

$$u_M(x, 0) = 0, \forall x > 0, \quad \text{Initial condition} \quad (2)$$

$$u_M(0, t) = 1, \forall t \geq 0, \quad \text{Boundary condition} \quad (3)$$

where  $u_M$  represents the wetting-phase saturation,  $f_M$  is the fractional flow function and  $M$  is the mobility ratio of the two fluid phases. We use the subscript  $M$  to indicate that, once the form of the constitutive relations is specified, the solutions of problem (1)–(3) are characterized solely by the mobility ratio.

This first-order hyperbolic equation is of interest as its solution can display both smooth solutions (rarefactions) and sharp fronts (shocks). Although the solution to this problem can be obtained analytically for simple

one-dimensional settings, the precise and stable resolution of these shocks poses well-known challenges for numerical methods<sup>44</sup>.

Physics-informed neural networks (PINNs) have been tested on this problem by Fuks and Tchelepi<sup>45</sup> who report good performance for concave fractional flow functions. In addition, Fraces and Tchelepi<sup>59</sup> provide an accurate solution introducing two physical constraints and subsequently modifying the fractional flux equation by a piece-wise form which differentiable form allow to capture the shock. However, these constraints are problem dependent and do not provide a general solution for hyperbolic equations in heterogeneous media. Whether the solution of the BL problem with non-convex flux function can be learnt by deep neural networks without the aid of artificial physical constraints remains an open question.

We take  $f_M$  to be the S-shaped flux function

$$f_M(x, t) = \frac{u_M(x, t)^2}{u_M(x, t)^2 + \frac{1}{M}(1 - u_M(x, t))^2}, \tag{4}$$

for which we can obtain the analytical solution of the problem:

$$u_M(x, t) = \begin{cases} 0, & \frac{x}{t} > f'_M(u^*), \\ (f'_M)^{-1}(x/t), & f'_M(u^*) \geq \frac{x}{t} \geq f'_M(u = 1), \\ 1, & f'_M(u = 1) \geq \frac{x}{t}, \end{cases} \tag{5}$$

where  $f'_M(u^*) = [f_M(u^*) - f_M(u)|_{u=0}]/(u^* - u|_{u=0})$ , and  $u^*$  represents the shock location defined by the Rankine–Hugoniot condition<sup>44</sup>. Note that equation (5) describes a family of functions of space, time, characterized by fluid mobility ratio  $M$ ,  $u_M(x, t)$ . In “Section Results” we use this analytical solution to test the accuracy of the PIANN model.

### Methodology

Let  $\mathcal{G} := \{(x_i, t_j) \in \mathbb{R}_0^+ \times \mathbb{R}_0^+ : i = 0, \dots, N, j = 0, \dots, T\}$  be a discrete version of the domain of  $u_M$ . We define our PIANN as a vector function  $\mathbf{u}_\theta : \mathbb{R}_0^+ \times \mathbb{R}^+ \rightarrow [0, 1]^{N+1}$ , where  $\theta$  are the weights of the network to be estimated during training. The inputs for the proposed architecture are pairs of  $(t, M)$  and the output is a vector where the  $i$ -th component is the solution evaluated in  $x_i$ . Notice the different treatment applied to spatial and temporal coordinates. Whereas  $t$  is a variable of the vector function  $\mathbf{u}_\theta$ , the locations where we calculated the solution  $x_0, \dots, x_N$  are fixed in advance. The output is a saturation map and therefore its values have to be in the interval  $[0, 1]$ .

For the sake of readability, we introduce the architecture of  $\mathbf{u}_\theta$  in “section PIANN architecture”. However, we note already here that in order to enforce the boundary condition, we let our PIANN learn only the components  $\mathbf{u}_\theta(t, M)_1, \dots, \mathbf{u}_\theta(t, M)_N, \forall t \neq 0$  and then we concatenate the component  $\mathbf{u}_\theta(t, M)_0 = 1$ . [Non-Dirichlet boundary conditions would require to be included as a term of the loss function.] To enforce the initial conditions, we set  $\mathbf{u}_\theta(0, M)_i = 0, i = 1, \dots, N$ . To enforce that the solution be in the interval  $[0, 1]$ , a sigmoid activation function is applied to each component of the last layer of our PIANN.

The parameters of the PIANN are estimated according to the physics-informed learning approach, which states that  $\theta$  can be estimated from the BL equation Eq. (1), the initial conditions Eq. (2) and boundary conditions Eq. (3), or in other words, no examples of the solution are needed to train a PINN.

After utilizing the information provided by the initial and boundary conditions enforcing  $\mathbf{u}_\theta(0, M)_i = 0, i = 1, \dots, N$  and  $\mathbf{u}_\theta(t, M)_0 = 1$ , respectively, we now define a loss function based on the information provided by Eq. (1). To calculate the first term we propose two options. The first option is a central finite difference approximation, that is,

$$\mathcal{R}_1(\theta, M)_{i,j} = \frac{\mathbf{u}_\theta(t_{j+1}, M)_i - \mathbf{u}_\theta(t_{j-1}, M)_i}{t_{j+1} - t_{j-1}}, \quad i = 1, \dots, N - 1, \quad j = 1, \dots, T - 1. \tag{6}$$

Alternatively, we can calculate the derivative of our PIANN with respect to  $t$  since we know the functional form of  $\mathbf{u}_\theta$ . It can be calculated using the automatic differentiation tools included in many machine learning libraries, such as Pytorch. Thus, we propose a second option to calculate this term as  $\mathcal{R}_1(\theta, M)_{i,j} = \partial \mathbf{u}_\theta(t, M)_i / \partial t|_{t=t_j}$ .

The second term of Eq. (1), the derivative of the flux with respect to the spatial coordinate, is approximated using central finite difference as

$$\mathcal{R}_2(\theta, M)_{i,j} = \frac{\mathbf{f}_\theta(t_j, M)_{i+1} - \mathbf{f}_\theta(t_j, M)_{i-1}}{x_{i+1} - x_{i-1}}, \quad i = 1, \dots, N - 1, \quad j = 1, \dots, T - 1, \tag{7}$$

where the vector of fluxes at the  $i$ -th location  $x_i$  is calculated as

$$\mathbf{f}_\theta(t, M)_i = \frac{\mathbf{u}_\theta(t, M)_i^2}{\mathbf{u}_\theta(t, M)_i^2 + \frac{(1 - \mathbf{u}_\theta(t, M)_i)^2}{M}}, \quad i = 0, \dots, N. \tag{8}$$

The spatial coordinate  $x$  is included as a fixed parameter in our architecture.

The loss function to estimate the parameters of the PINN is given as

$$\mathcal{L}(\theta) = \sum_M \|\mathcal{R}_1(\theta, M) + \mathcal{R}_2(\theta, M)\|_F^2, \tag{9}$$

where  $\|\cdot\|_F$  is the Frobenius norm.

There are two main lines in PINNs literature to deal with the initial and boundary conditions. The first one is to include them as a penalty term in the objective function<sup>19,46,53</sup>. The main drawback of this approach is, if this penalty term is not zero after training, the initial and boundary conditions are not totally satisfied by the solution. To solve this problem, we directly enforce the initial and boundary conditions in the architecture of our PIANN. Thus, we follow the path of the second line in PINN literature<sup>54–56</sup> that enforces initial and boundary conditions as hard constraints. With respect to the first line, this provides several advantages. First, we enforce a stronger constraint that does not allow any error on the initial and boundary conditions. Second, the PIANN does not need to learn these conditions by itself, and it can instead concentrate exclusively on learning the parameters that minimize the residuals of the BL equation. Third, there are no weights to be tuned to control the effect of the initial and boundary conditions in the final solution.

The parameters of the PIANN are estimated using the *Adam* optimizer<sup>60</sup> to minimize Eq. (9) with respect to  $\theta$ . Training algorithm pseudo-code is provided in Algorithm 1.

---

#### Algorithm 1 Training algorithm pseudo-code for PIANN

---

**Require:**  $\mathcal{T}$  a set of time values  $t$ ,  $\mathcal{M}$  a set of values for  $M$ , and initial values for neural net parameters  $\theta$  (random initialization in our case).

- 1: **repeat**
  - 2: Forward pass of PIANN to calculate all  $\mathbf{u}_\theta(t, M), \forall (t, M) \in \mathcal{T} \times \mathcal{M}$
  - 3: Calculate the first term of the residual  $\mathcal{R}_1(\theta, M)$  as in eq. (6).
  - 4: Calculate the values of the flux function  $\mathbf{f}_\theta(t, M)$  using eq. (8).
  - 5: Calculate the second term of the residual  $\mathcal{R}_2(\theta, M)$  using the values of flux function in eq. (7).
  - 6: Calculate the objective function  $\mathcal{L}(\theta)$  as in eq. (9).
  - 7: Calculate the derivatives of  $\mathcal{L}(\theta)$  with respect to  $\theta$  using automatic differentiation.
  - 8: Update the parameters  $\theta$  of the PIANN according to *Adam* optimizer steps.
  - 9: **until** convergence
  - 10: **output** The optimized parameters of the PIANN  $\theta$
- 

### PIANN architecture

Although it has been demonstrated that neural networks are universal function approximators, the proper choice of architecture generally aids learning and certain challenging problems (e.g. solving non-linear PDEs) may require more specific architectures to capture all their properties (see for instance Extreme Learning Machine<sup>61,62</sup>). For that reason, we have proposed a new architecture, inspired by<sup>51</sup>, to solve non-linear PDEs with discontinuities under two assumptions.

First, to automatically detect discontinuities we need an architecture that can exploit the correlations between the values of the solution for all spatial locations  $x_1, \dots, x_N$ . Second, the architecture has to be flexible enough to capture different behaviors of the solution at different regions of the domain. To this end, we propose the use of encoder-decoder GRUs<sup>2</sup> for predicting the solution at all locations at once, with the use of a recent machine learning tool known as attention mechanisms<sup>52</sup>.

Our approach presents several advantages compared to traditional simulators: (i) Instead of using just neighboring cells' information to calculate  $\mathbf{u}$  as in numerical methods, our architecture uses the complete encoded sequence input of the grid to obtain  $\mathbf{u}_i$ , allowing us to capture non-local relationships that numerical methods struggle to identify. (ii) The computer time for the forward pass of neural networks models is linear with respect to the number of cells in our grid. In other words, our method is a faster alternative with respect to traditional numerical methods of solving PDEs such as finite difference.

Figure 1 shows an outline of the proposed architecture. We start feeding the input pair  $(t, M)$  to a single fully connected layer. Thus, we obtain  $\mathbf{h}^0$  the initial hidden state of a sequence of  $N$  GRU blocks (yellow). Each of them corresponds to a spatial coordinate  $x_i$  which is combined with the previous hidden state  $\mathbf{h}^{i-1}$  inside the block.

This generates a set of vectors  $\mathbf{y}^1, \dots, \mathbf{y}^N$  which can be understood as a representation of the input in a latent space. The definitive solution  $\mathbf{u}$  (we omit the subindex  $\theta$  for simplicity) is reached after a new sequence of GRU blocks (blue) whose initial hidden state  $\mathbf{d}^0$  is initialized as  $\mathbf{h}^N$  to preserve the memory of the system.

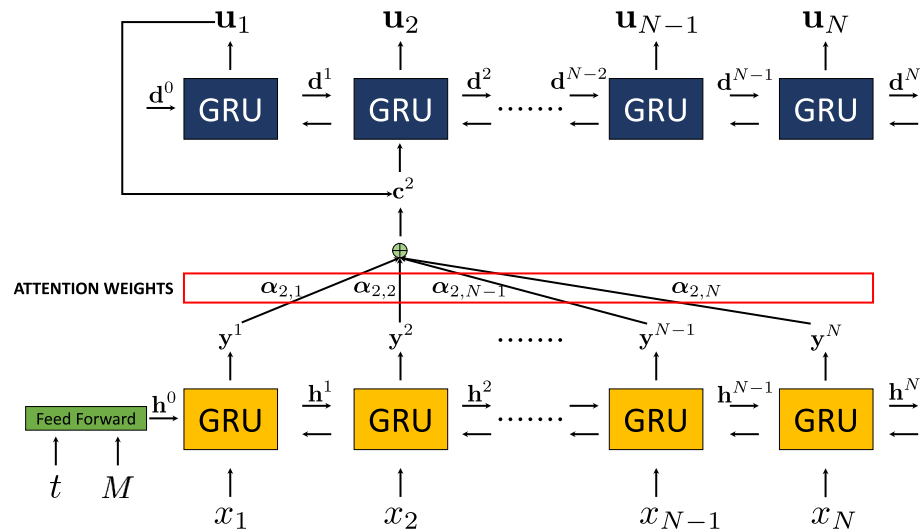
In addition to the hidden state  $\mathbf{d}^i$ , the  $i$ -th block  $g_i$  is fed with a concatenation of the solution at the previous location and a context vector, that is

$$\mathbf{u}_i = g_i([\mathbf{u}_{i-1}, \mathbf{c}^i], \mathbf{d}^{i-1}). \quad (10)$$

How the context vector is obtained is one of the key aspects of our architecture, since it will provide the PINN with enough flexibility to fit to the different behaviors of the solution depending on the region of the domain. Inspired by<sup>51</sup>, we introduce an attention mechanism between both GRU block sequences. Our attention mechanism is a single fully connected layer,  $a$ , that learns the relationship between each component of  $\mathbf{y}^j$  and the hidden states of the (blue) GRU sequence,

$$\mathcal{E}_{i,j} = a(\mathbf{d}^{i-1}, \mathbf{y}^j). \quad (11)$$

Then, the rows of matrix  $\mathcal{E}$  are normalized using a softmax function as



**Figure 1.** Architecture of physical attention neural network for the prediction of the variable  $u_2$ .

$$\alpha_{i,j} = \frac{\exp(\mathcal{E}_{i,j})}{\sum_{j=1}^N \exp(\mathcal{E}_{i,j})}, \tag{12}$$

and the context vectors are calculated as

$$c^i = \sum_{j=1}^N \alpha_{i,j} y^j, i = 1, \dots, N. \tag{13}$$

The coefficients  $\alpha_{i,j}$  can be understood as the degree of influence of the component  $y^j$  in the output  $u_i$ . This is one of the main innovations of our work to solve hyperbolic equations with discontinuities. The attention mechanism automatically determines the most relevant encoded information of the full sequence of the input data to predict the  $u_i$ . In other words, attention mechanism is a new method that allows one to determine the location of the shock automatically and provide more accurate behavior of the PIANN model around this location. This new methodology breaks the limitations explored by other authors<sup>39,45,46</sup> since is able to capture the discontinuity without specific prior information or the regularization term of the residual.

This is the first paper to use attention mechanisms to solve PDEs. Furthermore, we show that attention mechanisms are particularly effective in capturing the challenging nonlinear features of hyperbolic PDEs with discontinuous solution.

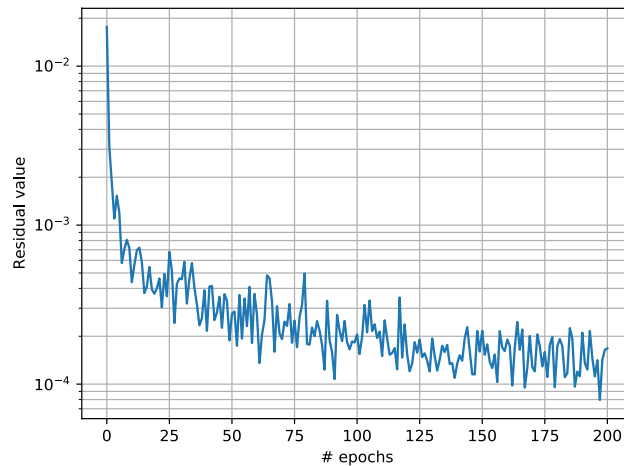
### Results

To test the proposed PIANN methodology, we perform a set of numerical experiments where we train our network to learn the solutions of the Buckley–Leverett problem (1)–(3) for a wide range mobility ratios. We compare our numerical predictions with the analytical solution given in Eq. (5).

The training set is given by a grid  $\mathcal{G} = \{(x_i, t_j) \in \mathbb{R}_0^+ \times \mathbb{R}_0^+ : x_i \in \{0, 0.01, \dots, 0.99, 1\}, t_j \in \{0, 0.01, \dots, 0.49, 0.5\}\}$ , and a set of values of  $M \in \{2, 4, 6, \dots, 100\}$ , which produces  $N = 101$ ,  $T = 51$ , and a total of 257,550 points. We want to emphasize that no examples of the solution are known at these points, and therefore no expensive and slow simulators are required to build the training set. To estimate the parameters of the PIANN we minimize Eq. (9) by running *Adam* optimizer<sup>60</sup> for 200 epochs with a learning rate of 0.001.

Figure 2 shows the residual value for the testing dataset for the different epoch for  $M = 4.5$ . We can observe a fast convergence of the method and a cumulative value of the residual smaller than  $10^{-4}$  after a few epochs. This demonstrates that we are minimizing the residuals in Eq. (1) and subsequently solving the the equation that governs BL.

Figure 3 shows the comparison between the analytical solution (red) and the solution obtained by our PIANN (blue) for different  $M$  used during training. Top, middle and bottom rows correspond to  $M = 2$ ,  $M = 48$  and  $M = 98$ , respectively, and the columns from left to right, correspond to different time steps  $t = 0.04$ ,  $t = 0.20$ , and  $t = 0.40$ , respectively. We can distinguish three regions of interest in the solution. Following increasing  $x$  coordinates, the first region on the left is one where the water saturation varies smoothly following the rarefaction part of the solution. The second region is a sharp saturation change that corresponds to the shock in the solution and the third region is ahead of the shock, with undisturbed water saturation values that are still at zero. For all cases, we observe that the PIANN properly learns the correct rarefaction behavior of the first region and approximates the analytical solution extremely well. In the third region, the PIANN also fits to the analytical



**Figure 2.** Residual values for each epoch for  $M = 4.5$  values in a semilog scale.

solution perfectly and displays an undisturbed water saturation at zero. As for any classical numerical methods, the shock region is the most challenging to resolve.

Around the shock front, the PIANN seems unable to perfectly propagate a strict discontinuity, and the represented behavior is a smeared shock that is smoothed over and displays small non-monotonic artifacts upstream and downstream of the front. The location of the shock is, however, well captured, suggesting that the network has correctly learnt the conservative nature of the BL equation. The non-monotonic behavior is reminiscent of the behavior observed in higher-order finite difference or finite volume methods, where slope-limiters are often used to correct for the non-physical oscillations.

The PIANN needs to learn where the shock is located in order to fit differently to both sides of it. This is the role of the attention mechanism of our architecture. On top of each figure we have visualized the attention map introduced by<sup>51</sup> for every timestep. These maps visualize the attention weight  $\alpha_{i,j}$  to predict the variable  $\mathbf{u}_i$ . We observe that in all cases the attention mechanism identifies the discontinuity, water front, and subsequently modifies the behavior of the network in the three different regions described above. This shows that attention mechanism provides all the necessary information to capture discontinuities automatically without the necessity of training data or a prior knowledge. Finally, it is important to note that attention weights of the attention mechanism are constant when the shock/discontinuity disappears.

We test the behavior of our methodology to provide solutions for BL at points within the training range of  $M$ :  $M = 4.5$  and  $M = 71$ . In other words, we want to check the capability of our PIANN model to interpolate solutions. Figure 4 shows that our PIANNs provide solutions that correctly detect the shock. Figure 5 shows the absolute error for both mobility ratios. As we can observe, the absolute error is zero after the shock and the average absolute error is smaller than  $10^{-3}$ .

In addition to interpolation, we have studied the behavior of PIANN for extrapolation, in other words, to solve BL for mobility ratios outside the initial training set,  $M \in [2, 100]$ . Specifically, we explore the network predictions for values of  $M$  outside the training set:  $M = 140$ ,  $M = 250$  and  $M = 500$ . Figure 6 compares the predicted and the exact solutions of the PDEs. As expected, the accuracy degrades when  $M$  is far from the original training set. We observe that our method predicts the behavior of the shock and accurate solution for  $M = 140$  and  $M = 250$ . However, the shock is totally missed for  $M = 500$  and, as such, retraining the model is recommended for larger values of  $M$ . It is important to note that the results show that our method is stable and converges for the different cases.

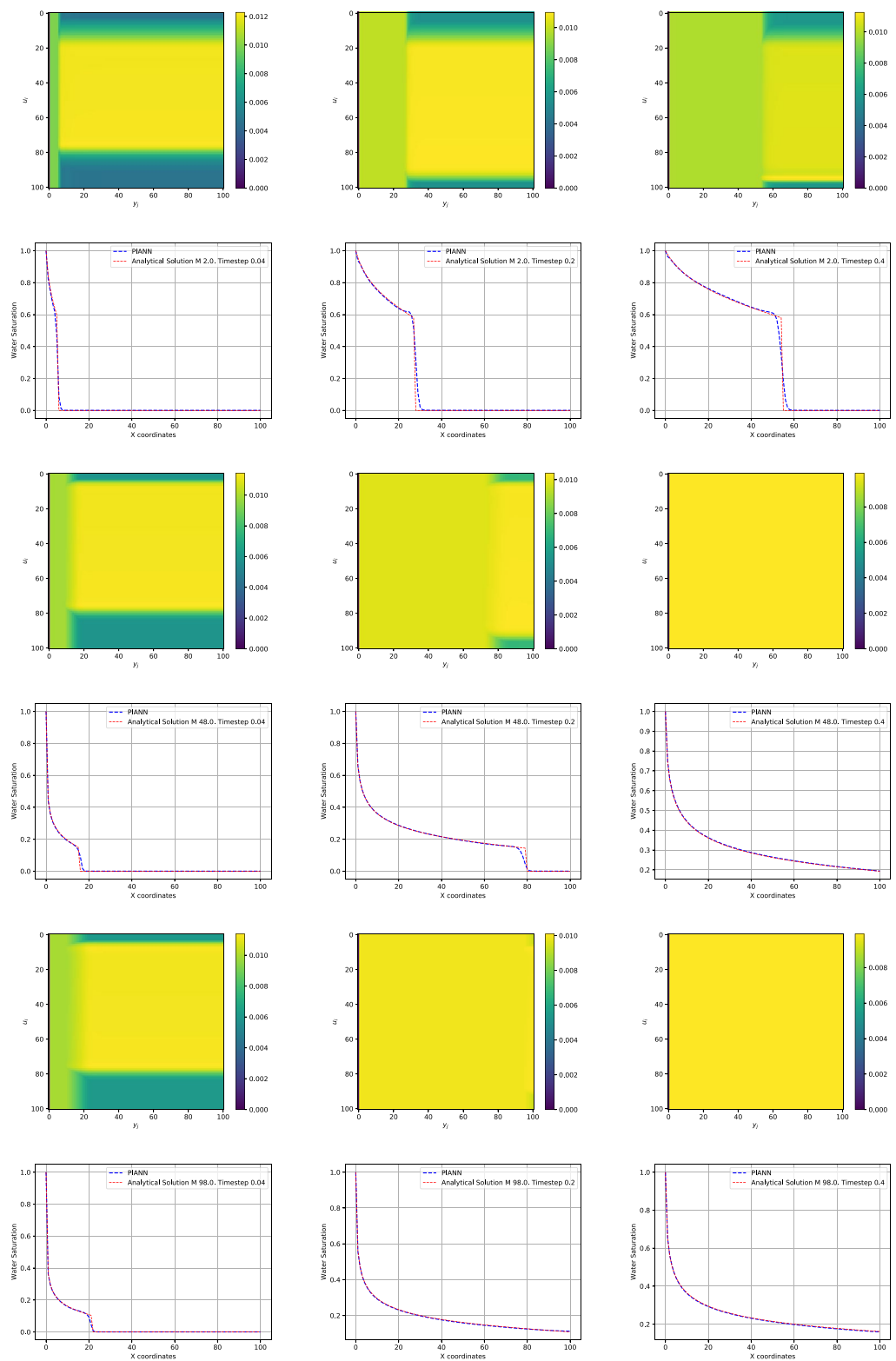
We test how the neural residual error progresses based on different  $\Delta t$  and  $\Delta x$  resolutions. Results are shown in Table 1, and demonstrate that our PIANN obtains smaller residuals when the resolution of the training set increases. However, we observe that changes in the residual are not highly significant. This is an advantage with respect to traditional numerical methods in computational fluid dynamics, where smaller values of  $\Delta t$  are necessary to capture the shock and guarantee convergence and stability.

Finally, we have compared the results with central and upwind finite difference schemes for the term of the vector of fluxes. The first-order upwind difference introduces a dissipation error when applied to the residual of the Buckley–Leverett equation, which is equivalent to regularizing the problem via artificial diffusion. Figure 7 shows that both approaches present similar results respect to the analytical solution. The fact that both central and upwind differences yield similar predictions is important, because it suggests that the proposed PIANN approach does not rely on artificial dissipation for shock capturing.

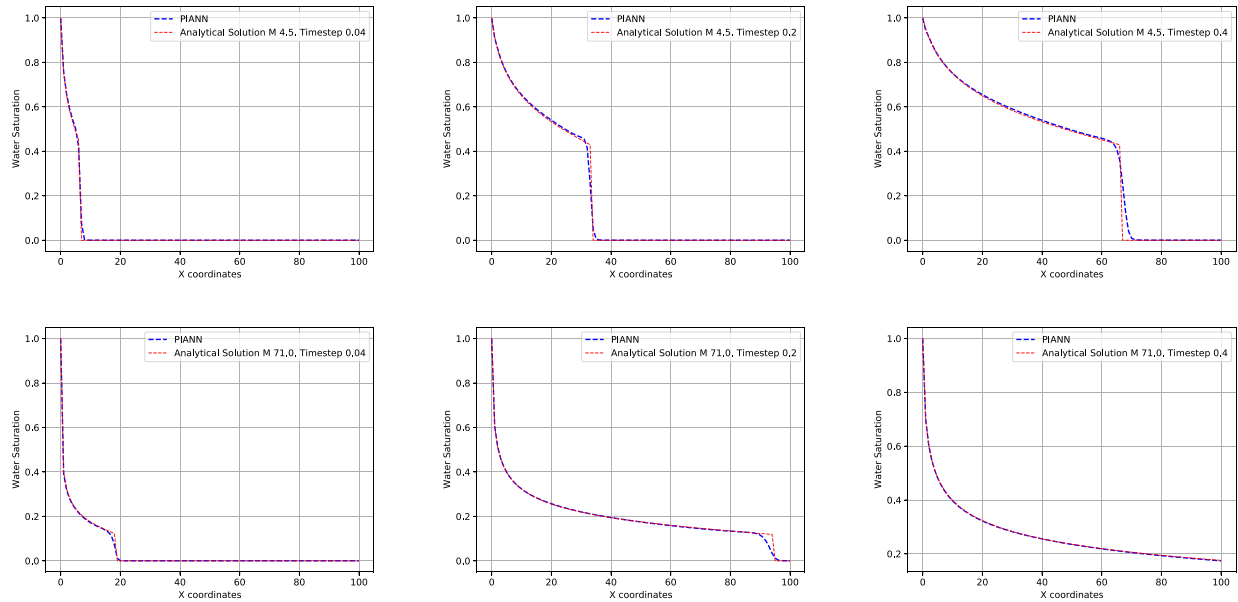
## Discussion

In this work, we have introduced a new method to solve hyperbolic PDEs. We propose a new perspective by focusing on network architectures rather than on residual regularization. We call our new architecture a physics informed attention neural network (PIANN).

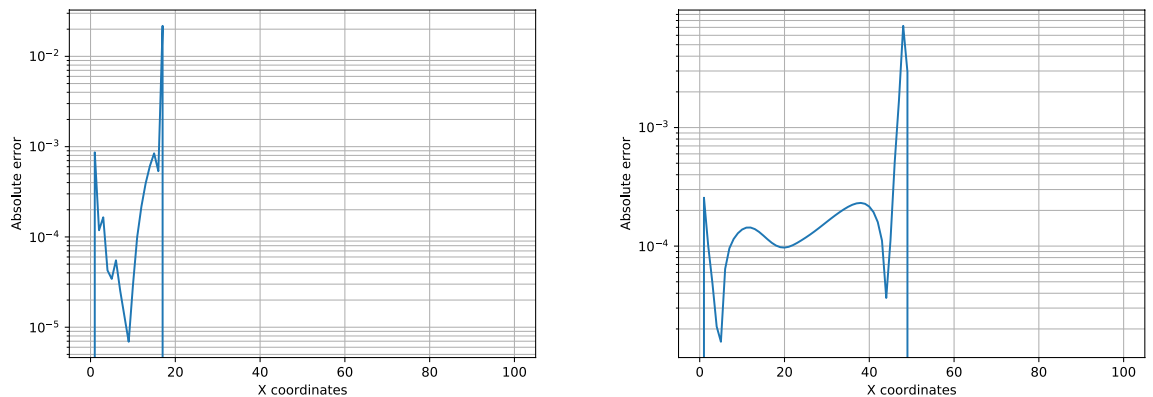




**Figure 3.** Top and bottom rows correspond to  $M = 2$  and  $M = 48$  and  $M = 98$  for attention weights map and comparison of the predicted by the neural network and the exact solutions of the PDE, respectively. The columns from left to right, correspond to different time steps  $t = 0.04$ ,  $t = 0.20$  and  $t = 0.40$ .



**Figure 4.** Top and bottom rows correspond to  $M = 4.5$  and  $M = 71$  for comparison of the predicted by the neural network and the exact solutions of the PDE, respectively. The columns from left to right, correspond to different time steps  $t = 0.04$ ,  $t = 0.20$  and  $t = 0.40$ .



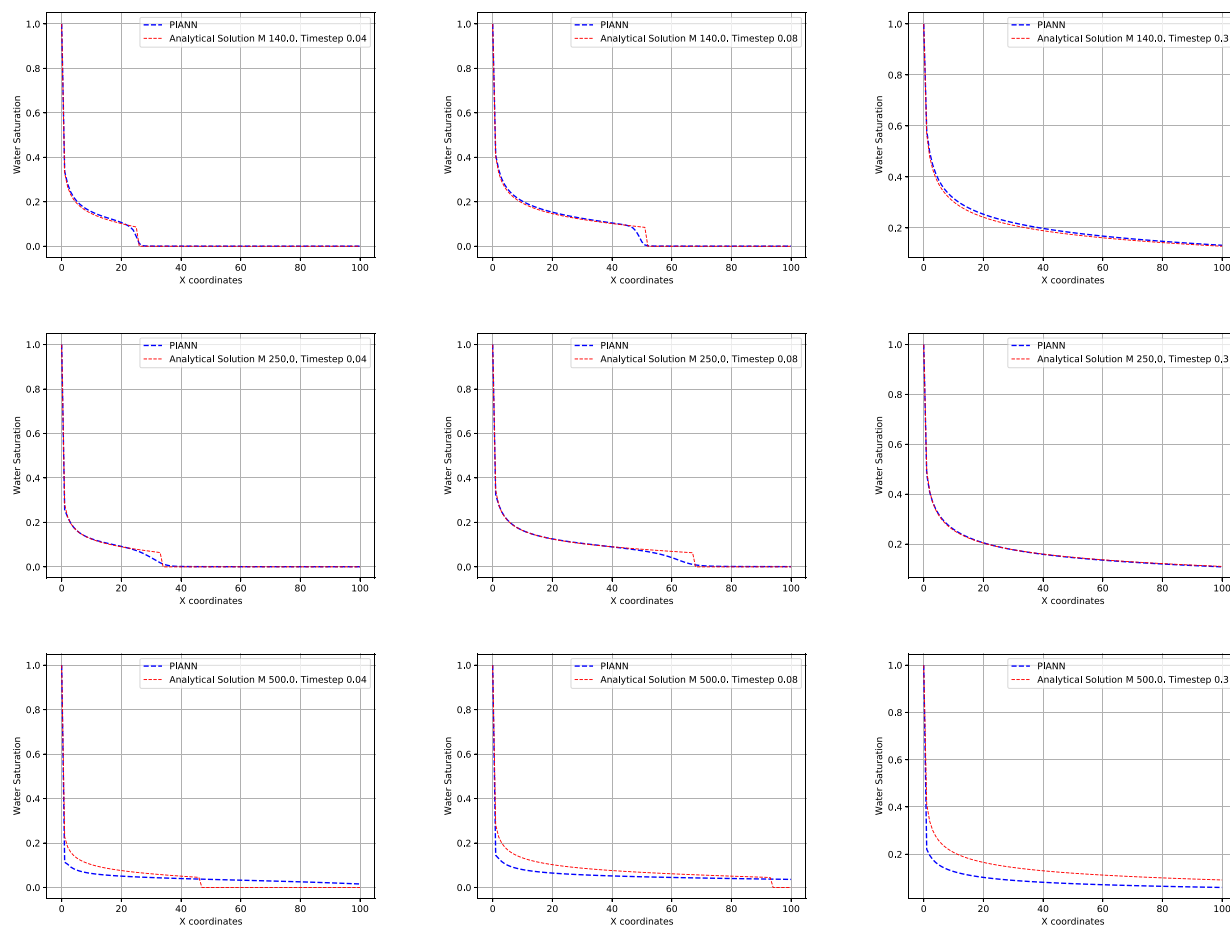
**Figure 5.** Absolute error for  $M = 4.5$  (left) and  $M = 71$  (right) for time step 0.10.

PIANN's novel architecture is based on two assumptions. First, correlations between values of the solution at all the spatial locations must be exploited, and second, the architecture has to be flexible enough to identify the shock and capture different behaviors of the solution at different regions of the domain. We have proposed an encoder-decoder GRU-based network to use the most relevant information of the fully encoded information, combined with the use of an attention mechanism. The attention mechanism is responsible for identifying the shock location and adapting the behavior of the PIANN model.

The loss function of PIANNs is based solely on the residuals of the PDE, and the initial and boundary conditions are introduced as hard constraints in the architecture. As a result, PIANN's training aims only at minimizing the residual of the PDE; no hyperparameters are needed to control the effect of initial and boundary conditions on the solution.

We have applied the proposed methodology to the non-concave flux Buckley–Leverett problem, which has hitherto been an open problem for PINNs. The experimental results support the validity of the proposed methodology and conclude that: (i) during training, the residuals of the equation decrease quickly to values smaller than  $10^{-4}$ , which means that our methodology is indeed solving the differential equation, (ii) the attention mechanism automatically detects shock waves of the solution and allows the PIANN to fit to the different behaviors of the analytical solution, and (iii) the PIANN is able to interpolate solutions for values of the mobility ratio  $M$  inside the range of training set, as well as to extrapolate when the value of  $M$  is outside the range. However, we observe that if  $M$  is too far away from range of the training set, the quality of the solution decreases. In that case, retraining of the network is recommended. (iv) We observe that the residuals decrease when the resolution of the training set increases. However, the change in the residuals is not highly significant. This is advantageous





**Figure 6.** Top and bottom rows correspond to  $M = 140$  and  $M = 250$  and  $M = 500$  comparison of the predicted by the neural network and the exact solutions of the PDE, respectively. The columns from left to right, correspond to different time steps  $t = 0.04$ ,  $t = 0.08$  and  $t = 0.30$ .

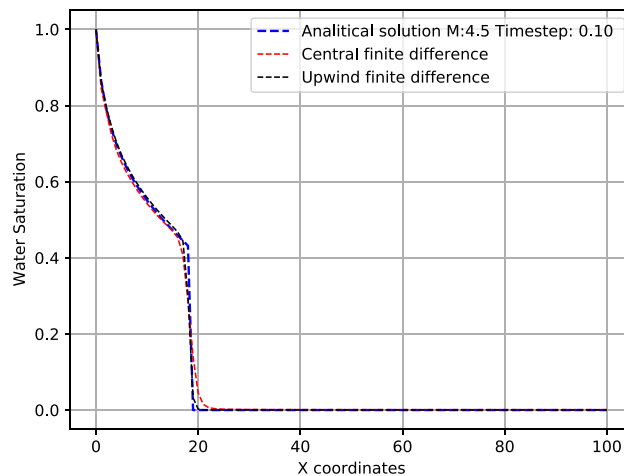
Resolution	Residual Error
$\Delta x = 1 * 10^{-2} \Delta t = 1 * 10^{-2}$	$1 * 10^{-4}$
$\Delta x = 5 * 10^{-3} \Delta t = 5 * 10^{-3}$	$9 * 10^{-5}$
$\Delta x = 1 * 10^{-3} \Delta t = 1 * 10^{-3}$	$8.7 * 10^{-5}$

**Table 1.** Residual calculation for different resolution of  $\Delta t$  and  $\Delta x$  for  $M = 4.5$ .

with respect to traditional numerical methods where small values of  $\Delta t$  are needed to capture the shock and guarantee convergence and stability.

## Conclusion

In conclusion, the proposed methodology transcends the current limitations of deep learning for solving hyperbolic PDEs with shock waves, and opens the door to applying these techniques to real-world problems, such as challenging reservoir simulations or carbon sequestration. However, the scaling up the proposed method for 3D problems will require to compute attention mechanism very intensive for memory, attention mechanism scale quadratically. A possible approach to overcoming this problem is to use a sparse representation of neural network, use attention mechanism to determine the most relevant time sequences or use simpler attention mechanism that has a linear dependency respect to number of cells. Finally, it is plausible that this method could be applied to model many processes in other domains which are described by non-linear PDEs with shock waves. The extent to which PIANNs could be applied in the more general space of differential equations is left to future research.



**Figure 7.** Comparison between solutions obtained with residuals evaluated using central and upwind finite differences, for  $M = 4.5$ .

### Data availability

The datasets generated and analysed during the current study are available in the google repository of OriGen AI [https://drive.google.com/drive/folders/16ZXRVSyTkie3s\\_mJyzRLej6MvoC33GIh](https://drive.google.com/drive/folders/16ZXRVSyTkie3s_mJyzRLej6MvoC33GIh).

Received: 12 December 2021; Accepted: 8 April 2022

Published online: 09 May 2022

### References

- Sejnowski, T. J. The unreasonable effectiveness of deep learning in artificial intelligence. *PNAS* **117**, 30033–30038 (2020).
- Cho, K. *et al.* Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint [arXiv:1406.1078](https://arxiv.org/abs/1406.1078) (2014).
- Sutskever, I., Vinyals, O. & Le, Q. V. Sequence to sequence learning with neural networks. arXiv preprint [arXiv:1409.3215](https://arxiv.org/abs/1409.3215) (2014).
- Hemanth, D. J. & Estrela, V. V. *Deep Learning for Image Processing Applications* Vol. 31 (IOS Press, 2017).
- Grigorescu, S., Trasnea, B., Cocias, T. & Macesanu, G. A survey of deep learning techniques for autonomous driving. *J. Field Robot.* **37**, 362–386 (2020).
- Johnson, P. A. *et al.* Laboratory earthquake forecasting: A machine learning competition. *PNAS* **118**, e2011362118 (2021).
- Kratzert, F. *et al.* Towards learning universal, regional, and local hydrological behaviors via machine learning applied to large-sample datasets. *Hydrol. Earth Syst. Sci.* **23**, 5089–5110 (2019).
- Nearing, G. S. *et al.* What role does hydrological science play in the age of machine learning?. *Water Resour. Res.* **57**, e2020WR028091 (2021).
- Justesen, N., Bontrager, P., Togelius, J. & Risi, S. Deep learning for video game playing. *IEEE Trans. Games* **12**, 1–20 (2019).
- Torrado, R. R., Bontrager, P., Togelius, J., Liu, J. & Perez-Liebana, D. Deep reinforcement learning for general video game ai. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, 1–8 (IEEE, 2018).
- Torrado, R. R. *et al.* Bootstrapping conditional gans for video game level generation. In *2020 IEEE Conference on Games (CoG)*, 41–48 (IEEE, 2020).
- Bar-Sinai, Y., Hoyer, S., Hickey, J. & Brenner, M. P. Learning data-driven discretizations for partial differential equations. *PNAS* **116**, 15344–15349 (2019).
- Haghighat, E., Raissi, M., Moure, A., Gomez, H. & Juanes, R. A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics. *Comput. Methods Appl. Mech. Eng.* **379**, 113741 (2021).
- Lu, L., Meng, X., Mao, Z. & Karniadakis, G. E. Deepxde: A deep learning library for solving differential equations. *SIAM Rev.* **63**, 208–228 (2021).
- Regazzoni, F., Dedé, L. & Quarteroni, A. Machine learning for fast and reliable solution of time-dependent differential equations. *J. Comput. Phys.* **397**, 108852 (2019).
- Samaniego, E. *et al.* An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementation and applications. *Comput. Methods Appl. Mech. Eng.* **362**, 112790 (2020).
- Han, J., Jentzen, A. & Weinan, E. Solving high-dimensional partial differential equations using deep learning. *PNAS* **115**, 8505–8510 (2018).
- Beck, C., Hutzenthaler, M., Jentzen, A. & Kuckuck, B. An overview on deep learning-based approximation methods for partial differential equations. arXiv preprint [arXiv:2012.12348](https://arxiv.org/abs/2012.12348) (2020).
- Raissi, M., Perdikaris, P. & Karniadakis, G. E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* **378**, 686–707 (2019).
- Raissi, M., Perdikaris, P. & Karniadakis, G. E. Physics informed deep learning (part I): Data-driven solutions of nonlinear partial differential equations. arXiv preprint [arXiv:1711.10561](https://arxiv.org/abs/1711.10561) (2017).
- Karniadakis, G. E. *et al.* Physics-informed machine learning. *Nat. Rev. Phys.* **3**, 422–440 (2021).
- Raissi, M., Yazdani, A. & Karniadakis, G. E. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science* **367**, 1026–1030 (2020).
- Brunton, S. L., Noack, B. R. & Koumoutsakos, P. Machine learning for fluid mechanics. *Annu. Rev. Fluid Mech.* **52**, 477–508 (2020).
- Kadeethum, T., Jørgensen, T. M. & Nick, H. M. Physics-informed neural networks for solving nonlinear diffusivity and Biot's equations. *PLoS ONE* **15**, e0232683 (2020).
- Mishra, S. & Molinaro, R. Physics informed neural networks for simulating radiative transfer. *J. Quant. Spectrosc. Radiat. Transfer* **270**, 107705 (2021).

26. De Florio, M., Schiassi, E., Furfaro, R., Ganapol, B. D. & Mostacci, D. Solutions of chandrasekhar's basic problem in radiative transfer via theory of functional connections. *J. Quant. Spectrosc. Radiat. Transfer* **259**, 107384 (2021).
27. Lou, Q., Meng, X. & Karniadakis, G. E. Physics-informed neural networks for solving forward and inverse flow problems via the Boltzmann-BGK formulation. *J. Comput. Phys.* **447**, 110676 (2021).
28. De Florio, M., Schiassi, E., Ganapol, B. D. & Furfaro, R. Physics-informed neural networks for rarefied-gas dynamics: Thermal creep flow in the bhatnagar-gross-krook approximation. *Phys. Fluids* **33**, 047110 (2021).
29. Jagtap, A. D., Mitsotakis, D. & Karniadakis, G. E. Deep learning of inverse water waves problems using multi-fidelity data: Application to Serre-Green-Naghdi equations. <https://doi.org/10.48550/ARXIV.2202.02899> (2022).
30. Jagtap, A. D., Kharazmi, E. & Karniadakis, G. E. Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. *Comput. Methods Appl. Mech. Eng.* **365**, 113028. <https://doi.org/10.1016/j.cma.2020.113028> (2020).
31. Jagtap, A. D., Mao, Z., Adams, N. & Karniadakis, G. E. Physics-informed neural networks for inverse problems in supersonic flows. <https://doi.org/10.48550/ARXIV.2202.11821> (2022).
32. De Florio, M., Schiassi, E., D'Ambrosio, A., Mortari, D. & Furfaro, R. Theory of functional connections applied to linear odes subject to integral constraints and linear ordinary integro-differential equations. *Math. Comput. Appl.* **26**, 65 (2021).
33. Ji, W., Qiu, W., Shi, Z., Pan, S. & Deng, S. Stiff-pinn: Physics-informed neural network for stiff chemical kinetics. *J. Phys. Chem. A* **125**, 8098–8106 (2021).
34. Kim, S., Ji, W., Deng, S., Ma, Y. & Rackauckas, C. Chaos: Stiff neural ordinary differential equations. *Interdiscip. J. Nonlinear Sci.* **31**, 093122 (2021).
35. D'Ambrosio, A., Schiassi, E., Curti, F. & Furfaro, R. Pontryagin neural networks with functional interpolation for optimal intercept problems. *Mathematics* **9**, 996 (2021).
36. Schiassi, E. *et al.* Physics-informed extreme theory of functional connections applied to optimal orbit transfer. In *Proceedings of the AAS/AIAA Astrodynamics Specialist Conference, Lake Tahoe, CA, USA*, 9–13 (2020).
37. Jin, X., Cai, S., Li, H. & Karniadakis, G. E. NSFnets (Navier–Stokes flow nets): Physics-informed neural networks for the incompressible Navier–Stokes equations. *J. Comput. Phys.* **426**, 109951 (2021).
38. Yang, L., Meng, X. & Karniadakis, G. E. B-pinns: Bayesian physics-informed neural networks for forward and inverse pde problems with noisy data. *J. Comput. Phys.* **425**, 109913 (2021).
39. Mao, Z., Jagtap, A. D. & Karniadakis, G. E. Physics-informed neural networks for high-speed flows. *Comput. Methods Appl. Mech. Eng.* **360**, 112789 (2020).
40. Jagtap, A. D., Kharazmi, E. & Karniadakis, G. E. Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. *Comput. Methods Appl. Mech. Eng.* **365**, 113028 (2020).
41. Zhang, D., Lu, L., Guo, L. & Karniadakis, G. E. Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems. *J. Comput. Phys.* **397**, 108850 (2019).
42. Tipireddy, R., Barajas-Solano, D. A. & Tartakovsky, A. M. Conditional Karhunen–Loève expansion for uncertainty quantification and active learning in partial differential equation models. *J. Comput. Phys.* **418**, 109604 (2020).
43. Dafermos, C. M. *Hyperbolic Conservation Laws in Continuum Physics* (Springer, 2000).
44. Leveque, R. J. *Numerical Methods for Conservation Laws (2. ed.)*. Lectures in Mathematics: ETH Zurich (Birkhäuser, 1992).
45. Fuks, O. & Tchelepi, H. A. Limitations of physics informed machine learning for nonlinear two-phase transport in porous media. *J. Mach. Learn. Model. Comput.* **1**, 10 (2020).
46. Fraces, C. G., Papaioannou, A. & Tchelepi, H. Physics informed deep learning for transport in porous media. Buckley Leverett problem. arXiv preprint [arXiv:2001.05172](https://arxiv.org/abs/2001.05172) (2020).
47. Fraces, C. G. & Tchelepi, H. Physics informed deep learning for flow and transport in porous media. arXiv preprint [arXiv:2104.02629](https://arxiv.org/abs/2104.02629) (2021).
48. Michoski, C., Milosavljevic, M., Oliver, T. & Hatch, D. R. Solving differential equations using deep neural networks. *Neurocomputing* **399**, 193–212 (2020).
49. Jagtap, A. D., Kawaguchi, K. & Karniadakis, G. E. Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. *J. Comput. Phys.* **404**, 109136 (2020).
50. Jagtap, A. D., Shin, Y., Kawaguchi, K. & Karniadakis, G. E. Deep kronecker neural networks: A general framework for neural networks with adaptive activation functions. *Neurocomputing* **468**, 165–180 (2022).
51. Bahdanau, D., Cho, K. & Bengio, Y. Neural machine translation by jointly learning to align and translate. arXiv preprint [arXiv:1409.0473](https://arxiv.org/abs/1409.0473) (2014).
52. Vaswani, A. *et al.* Attention is all you need. arXiv preprint [arXiv:1706.03762](https://arxiv.org/abs/1706.03762) (2017).
53. Sun, L., Gao, H., Pan, S. & Wang, J.-X. Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Comput. Methods Appl. Mech. Eng.* **361**, 112732 (2020).
54. Lagaris, I., Likas, A. & Fotiadis, D. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Trans. Neural Netw.* **9**, 987–1000 (1998).
55. Schiassi, E. *et al.* Extreme theory of functional connections: A fast physics-informed neural network method for solving ordinary and partial differential equations. *Neurocomputing* **457**, 334–356. <https://doi.org/10.1016/j.neucom.2021.06.015> (2021).
56. Schiassi, E., De Florio, M., D'Ambrosio, A., Mortari, D. & Furfaro, R. Physics-informed neural networks and functional interpolation for data-driven parameters discovery of epidemiological compartmental models. *Mathematics* **9**, 10 (2021).
57. Mortari, D. The theory of connections: Connecting points. *Mathematics* **5**, 10 (2017).
58. Buckley, S. & Leverett, M. Mechanism of fluid displacement in sands. *Trans. AIME* **241**, 107–116 (1942).
59. Gasmí, C. F. & Tchelepi, H. Physics informed deep learning for flow and transport in porous media (2021).
60. Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. arXiv preprint [arXiv:1412.6980](https://arxiv.org/abs/1412.6980) (2014).
61. Dwivedi, V. & Srinivasan, B. Physics informed extreme learning machine (pielm)—a rapid method for the numerical solution of partial differential equations. *Neurocomputing* **391**, 96–118. <https://doi.org/10.1016/j.neucom.2019.12.099> (2020).
62. Calabrò, F., Fabiani, G. & Siettos, C. Extreme learning machine collocation for the numerical solution of elliptic pdes with sharp gradients. *Comput. Methods Appl. Mech. Eng.* **387**, 114188 (2021).

## Author contributions

R.R.T. and P.R. developed methodology; L.C.F. provided data; R.R.T. implementation and experiments; L.C.F., S.M. and J.T. state of the art; R.R.T., P.R., L.C.F., S.M. and J.T. wrote the paper; M.C.G., T.F. and J.T. reviewed language. Competing interest statement: R.R.T. and P.R. have filed a patent application based on the research presented in this paper.

## Competing interests

R.R.T. and P.R. have filed a patent application based on the research presented in this paper.

### Additional information

**Correspondence** and requests for materials should be addressed to R.R.-T.

**Reprints and permissions information** is available at [www.nature.com/reprints](http://www.nature.com/reprints).

**Publisher's note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

© The Author(s) 2022