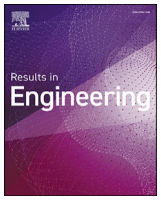
Training physics-informed neural networks: One learning to rule them all?

Results in Engineering 18 (2023) 101023

Contents lists available at ScienceDirect

Results in Engineering

journal homepage: www.sciencedirect.com/journal/results-in-engineering



Research paper

Training physics-informed neural networks: One learning to rule them all?

Simone Monaco *, Daniele Apiletti

Department of Control and Computer Engineering, Politecnico di Torino, Torino, 10129, Italy

ARTICLE INFO

Keywords:
Deep learning
Partial differential equations
Computational physics

ABSTRACT

Physics-informed neural networks (PINNs) are gaining popularity as powerful tools for solving nonlinear Partial Differential Equations (PDEs) through Deep Learning. PINNs are trained by incorporating physical laws as soft constraints in the loss function. Such an approach is effective for trivial equations, but fails in solving various classes of more complex dynamical systems.

In this work, we put on the test bench three state-of-the-art PINN training methods for solving three popular Partial Differential Equations (PDEs) of increasing complexity, besides the additional application of the Fourier Feature Embedding (FFE), and the introduction of a novel implementation of Curriculum regularization.

Experiments evaluate the convergence of the trained PINN and its prediction error rate for different training sizes and training lengths (i.e., number of epochs). To provide an overview of the behaviour of each learning method, we introduce a new metric, named overall score.

Our experiments show that a given approach can either be the best in all situations or not converge at all. The same PDE can be solved with different learning methods, which in turn give the best results, depending on the training size or the use of FFE. From our experiments we conclude that there is no learning method to train them all, yet we extract useful patterns that can drive future works in this growing area of research.

All code and data of this manuscript are publicly available on GitHub.

1. Introduction

Deep learning techniques have revolutionized scientific research in recent years [1–3]. The availability of data and the increasing plethora of methods to extract information from them are changing our understanding of the physical world and the way we dive into it. However, complete data acquisition for many complex real-world phenomena remains untractable. Faced with these scenarios, purely data-driven approaches present difficulties when the data is scarce or insufficient for the complexity of the problem. Moreover, many scientific problems also have to satisfy certain physical principles (e.g., conservation of mass, momentum, etc.), which is not guaranteed in traditional machine learning techniques. On the other side, many physical or engineering systems can also be modelled, or at least approximated, by means of equations describing their behaviour. The recent paradigm of blending these two worlds to take advantage of both data analytics and scientific discovery (by modelling equations and physical constraints) has been referred to as Theory-Guided Data Science or Physics-Informed Machine Learning [4–7].

An emerging direction in this field is provided by Physics-Informed Neural Networks (PINNs) [8–10]. PINNs are neural networks that take advantage of machine learning methodologies to solve scientific problems by constraining the networks to follow known physical laws [11–13]. This is generally done using the residuals of the underlying differential equations as extra loss function terms in the learning stage. This approach leads to the rise of a new class of function approximators, able to encode these laws or infer them from unknown data [14,15]. A naive application of the PINN framework is known to suffer from many difficulties in solving classes of problems that exhibit highly nonlinear, chaotic, or multiscale behaviour [5,16]. To cope with this issue, many works in the past few years tried to extend the original implementation to improve robustness and generalizability [17–20]. Similarly, many interpretations have been proposed on why PINNs fail in these problems [16]. However, the currently proposed approaches tend to address specific issues raised in some contexts, ignoring how

* Corresponding author.
E-mail address: simone.monaco@polito.it (S. Monaco).

https://doi.org/10.1016/j.rineng.2023.101023
Received 11 November 2022; Received in revised form 5 March 2023; Accepted 13 March 2023
Available online 17 March 2023

these strategies perform in other use cases, and making this process hard to generalize.

The aim of this work is to investigate the ability of PINNs to learn in different settings with variable difficulty and evaluate the limitations of different state-of-the-art PINN training strategies. To the best of the author's knowledge, the current literature is missing a multi-faceted experimental comparison on PINN's learning strategies across different use cases.

The contributions of the paper can be summarised as follows.

1. We compare 3 PINN learning approaches among the most relevant in the state of the art, by applying them in 3 benchmark use cases with partial differential equations of increasing complexity.
2. We analyze the impact of different parameters, such as training size and the number of epochs, on the results evaluated in terms of convergence rate and on the final capability of the PINNs to model the ground truth in terms of Mean Squared prediction Error (MSE).
3. We introduce a new version of curriculum regularization, which is included in the experimental comparison and which yields better results than previous state-of-the-art solutions in specific conditions.
4. We propose an overall score metric to evaluate the capability of a learning method to provide high-performance models in terms of convergence and prediction error rate.
5. We extract useful patterns out the variable behaviour of the different training methods and PDEs, which can drive future works and suggest research directions in this growing field.

The paper is organised as follows. Section 2 introduces the basic concept of the PINN algorithm in its vanilla implementation. Section 3 presents the related works with the most relevant strategies to improve the PINN learning. Section 4 describes the methods included in our experimental evaluation, whose design is provided in Section 5. Section 6 shows the experimental results of our analysis, measuring the impact of each method. Finally, Section 7 summarises the results on a broader view, and Section 8 draws conclusions and presents future works.

All code and data of this manuscript are publicly available at https://github.com/simone7monaco/PINNTrainingStrategies.

2. Physics-informed neural networks

The use of neural networks for the solution of parametrised Partial Differential Equations (PDEs) has a long history in the literature [21–23]. The PINN framework [24] addresses this task in a natural way by taking advantage of automatic differentiation. Suppose that we want to solve a dynamical problem which is following a known law within its domain Ω, namely:

$$\frac{\partial u}{\partial t} = \mathcal{N}[u] \quad x \in \Omega, \quad t \in [0, T], \tag{1}$$

where $\mathcal{N}[\cdot]$ stays for a generic differential operator and $u(x,t)$ is the solution of the system, that is, the target of the neural network. To completely define the problem, we will add initial conditions and boundary ones (e.g. periodic, fixed, etc.) in the form:

$$\begin{aligned} &\mathcal{B}[u] = g(x,t) \quad x \in \partial\Omega, \quad t \in [0, T], \\ &u(x,0) = h(x), \quad x \in \bar{\Omega}, \end{aligned} \tag{2}$$

where B can be another differential operator defined on the boundaries of Ω, and $\bar{\Omega}$ denotes the interiors of the domain in which the initial condition is provided.

Within this context, the PINN can be seen as the approximator $F_{NN}(x_i, t_i) := \hat{u}_i$ can be trained by enforcing together each of these constraints. The regular/vanilla implementation proposed by Raissi et al. [24] is configured as a sum of the residuals from the elements introduced previously:

$$\mathcal{L} = \mathcal{L}_{init} + \mathcal{L}_{bound} + \mathcal{L}_f, \tag{3}$$

where

$$\begin{aligned} \mathcal{L}_{init} &= \frac{1}{N_{init}} \sum_i |h(x_i) - \hat{u}_i|^2, \\ \mathcal{L}_{bound} &= \frac{1}{N_{bound}} \sum_i |g(x_i, t_i) - \mathcal{B}\left[\hat{u}_i\right]|^2, \\ \mathcal{L}_f &= \frac{1}{N_f} \sum_i |f(x_i, t_i)|^2. \end{aligned} \tag{4}$$

Each normalised sum in Eq. (4) spans all the sampled points from the respective domains. We also defined $f(x,t)$ as the residuals of the network evaluating Eq. (1):

$$f := \frac{\partial \hat{u}}{\partial t} - \mathcal{N}[\hat{u}]. \tag{5}$$

Finally, an additional data-driven term can extend this strategy to consider also experimental measures in various scientific problems.

As introduced in Section 1, literature shows that this framework fails to converge to the right solution in many cases. In Section 3, we collect some of the most relevant implementations that address these limitations.

3. Related works

Many works in the literature addressed the PINN application to solve different linear and nonlinear PDEs [24,8,5], although recent studies have also shown that PINNs can produce highly inaccurate approximations or even be unable to reach convergence when facing other classes of "harder" problems. One of the greatest issues leading to these results lies in the gradient descent applied to the PINN multi-objective loss function.

We can observe that the elements in Eq. (3) may have significantly different cardinalities, leading the weight update procedure of the neural network to pay more attention to some components than others. The general approach to overcome this problem is to set weighting coefficients for each loss component. When focussing their work on dissipative equations (i.e. not reversible), Wight et al. [25] assumed the initial condition term as the most important one to reach convergence. Then they add a fixed hyper-parameter $C_0 \gg 1$ as the coefficient of the $\mathcal{L}_{init}$ term. This choice comes from the intuition that a PINN failure at some time t would more likely lead to errors in the predictions at subsequent times. To automatically obtain the optimal value for this coefficient, Wang et al. [18] suggested making it variable during training, and selecting it from the statistics on the loss function gradients. The authors then applied this strategy, which is configured as a learning rate annealing, to balance all the components of Eq. (3). Another contribution to this approach is provided by [26], deriving the Neural Tangent Kernel matrix for the neural network and using its eigenvectors to define the loss coefficients.

From another perspective, many works tried to improve the vanilla PINN implementation by focusing on its tendency to propagate errors as the time step increases. This problem has been addressed by a time-adaptive sampling strategy [25], in which the authors defined in the training phase a gradually increasing upper bound in time t_i and sample points in the time range $[0, t_i]$. This procedure is repeated for N steps, having $t_N = T$, expecting a solution learnt throughout the domain at the end of the training.

An analogous approach is the sequence-to-sequence learning proposed in [27], in which the authors still separate the entire time domain into regions of size Δt and train the network on one of them at a time, having the next time-step initial conditions propagated from the previous one. The fact that these approaches rely on a reasonable convergence at one step to pass to the following makes them difficult to configure.

With the same motivation but with the intention of automatically letting the network learn which time step to consider based on its pre-

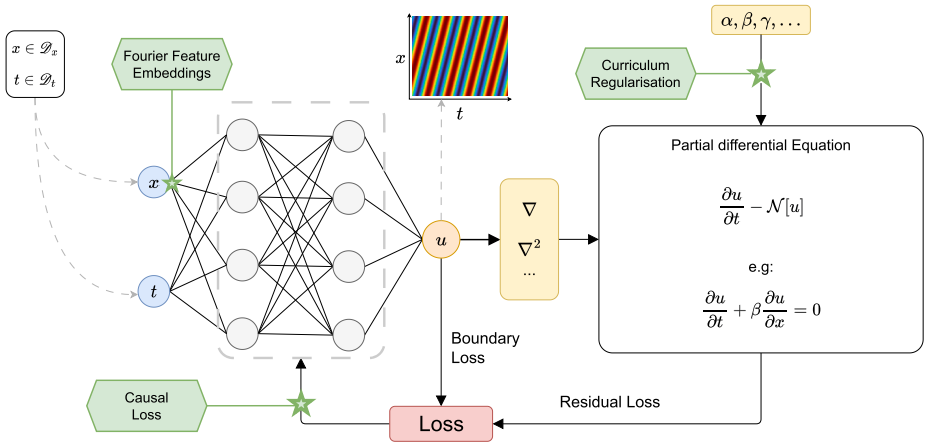


Fig. 1. Workflow diagram with the Vanilla training pipeline and the analysed enhanced strategies (the green hexagons).

diction, Wang et al. [16] proposed a new definition of the PINN loss function called the *causal training* strategy. In particular, they considered the $\mathcal{L}_f$ term of Equation (3) as a composition of multiple contributions for each time step in $[0, T]$ and weighted each of them based on how much the network has learnt in the earlier steps. This approach is effective in overcoming the time-dependency issue. Then, experimental evaluations of the authors prove that it leads to sensible results with no need for additional normalisation strategies on the other components of the loss function. We provide more details on this strategy and its implementation in section 4.2.

From a broader point of view, PINN failures occur because the loss function profile makes convergence harder than usual supervised tasks. In other words, the network is highly likely to get stuck at a local minimum as the complexity of the optimisation problem increases. A curriculum regularisation strategy [27] allows the system to find a good initialisation point by training the network to learn problems of increasing difficulty in the same training phase. We will discuss this method in detail in Section 4.1.

Among this collection of enhanced training strategies, it is rather difficult to find which is the best for each use case and/or if one single strategy can effectively apport benefits to the solution of a general physical system. Therefore, in the following of the paper we present the results of an experimental comparison among some of these strategies, which, to the best of our knowledge, are the most promising and the most reproducible thanks to the public resources made available by the original authors (papers, source code, datasets).

4. Learning techniques

This section will introduce the advanced training strategies we will evaluate further. Fig. 1 shows the Vanilla PINN training, in which the neural network weights are updated via optimization of the differential equation's residual loss and boundary/initial losses. Green hexagons represent alternative strategies that can eventually be introduced to optimize the training.

4.1. Curriculum regularisation

Curriculum learning was first introduced in the context of PINNs by Krishnapriyan et al. [27]. The basic idea is that PINN convergence fails when the parametrised differential equation residuals make the optimisation problem hard to solve. Contextually, the same equation with different parameters could lead to a simpler problem. One example of this behaviour is the convection equation (more details in Section 6.1), which is parametrized by a constant β. Fig. 2 shows the exact solution of the equation by increasing the parameter β. The authors proved that vanilla PINNs get lower and lower performance results as this value increases, suggesting that the problem is actually getting more complex. Within this context, one can start training the network on the easy-problem parameter value and make it vary during the training till reaching the desired value. In this way, at the end of each transition step, the network is ideally closer to the minima of the next step, which is a good initialisation point for the new (harder) problem.

One of the main limitations of this method is that it requires prior knowledge of the system's behaviour on the variation of its parameter. This is even more critical when the parameters are more than one. Furthermore, the final behaviour of a PINN is not straightforward; then the design of the training strategy requires some intuition about its actual performance.

In our implementation, once a target value *hard* is defined for a system parameter, we define a fixed number of steps to achieve it, starting from an *easy* initial value. The baseline implementation of this strategy assigns a fixed number of epochs at each of these steps. In this way, the network is trained on the actual problem only in the last step, assuming that the preliminary training drives the network sufficiently close to the final minimum.

From a slightly different perspective, we can instead think of curriculum regularisation just as an *initialization procedure* to put the network on the right way to the desired solution. Then, the length of the steps before the last one should be minimal to prepare the learning for the subsequent task. Calculating the best point at which to stop each step is not straightforward, but to give a measure of the effect of this change, we designed a novel strategy, depicted in Fig. 3 as Curriculum v2. Here we want to give "more time" to the network to focus on the target problem. To this aim, we simply split the total training time into one more curriculum step, letting the network train for more epochs on the final values of the parameters. This is arguably just one of the many alternatives to modify the curriculum strategy by emphasizing mostly one perspective or another. In Section 5, the two curriculum learning strategies, named v1 (the original) and v2 (our newly proposed one) are experimentally compared.

4.2. Causal training

The causal training strategy [16] considers the problem of PINNs degrading their prediction performance over time as if they did not respect physical causality within the dynamical system. The authors prove this behaviour by rewriting the residual component of Equation (4) to emphasise that the elements of the summations related to time t_i are based on the predictions at time t_{i-1}, which are possibly incorrect. Therefore, they propose to pay less attention to the next time steps as long as the network makes significant errors in all the previous time steps. To this aim, they rewrite the residual term as a weighted sum of the residual losses evaluated at a fixed time step, namely:

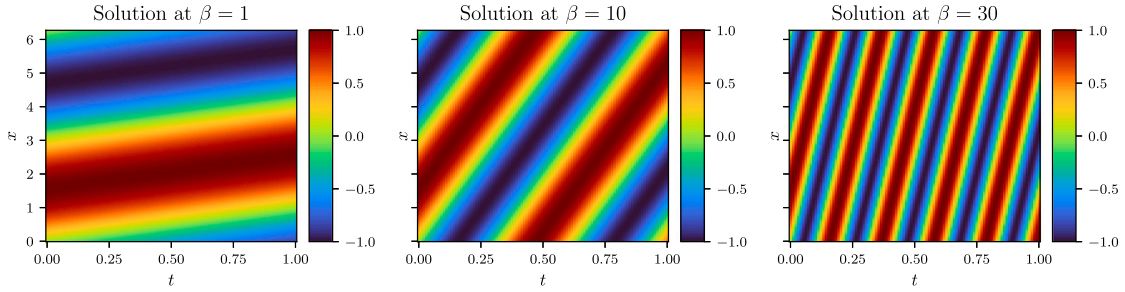


Fig. 2. Convection equation exact solutions at different values of β.

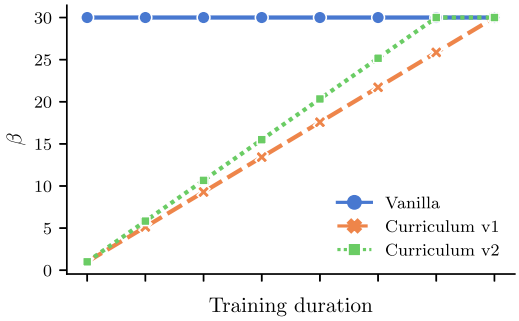


Fig. 3. Example for curriculum regularization strategy in version 1 and 2.

$$\mathcal{L}_f = \frac{1}{N_t}\sum_i w_i \mathcal{L}_f(t_i)$$
$$w_i = \exp\left(-\epsilon \sum_{k=0}^{i-1} \mathcal{L}_f(t_k)\right), \tag{6}$$

where ϵ is a *temperature-like* parameter to define the slope of the temporal weights and N_t is the number of time steps in the domain grid.

In this paper implementation, we used fixed parameters ϵ for each system. The basic implementation from the original paper only applies to PINNs trained on a complete mesh grid. In the following, we will discuss whether the extension to a limited random sampling over these points can be effective as well.

4.3. Fourier feature embeddings

An additional strategy included in our experimental comparison is a way to enforce exact periodic boundary conditions in the network output as hard constraints on the architecture [28]. The idea of the method is to rewrite the input space coordinates as Fourier components enforcing the desired periodicity of the final solution. Namely, fixing $\omega = \frac{2\pi}{L}$ and a non-negative integer M, it can be proven that the transformation

$$v(x) = (1, cos(\omega x), sin(\omega x), cos(2\omega x), ..., cos(M\omega x), sin(M\omega x)) \tag{7}$$

used as input to a function produces a result respecting periodicity on L by design. Then, any PINN prediction obtained on top of this embedding strategy, i.e., any output of $\hat{u}([v(x), t])$, will exactly respect the desired periodicity. It can be shown that such a strategy can simplify PINN training and reduce convergence time [16] in some benchmark problems. In this paper we name this strategy as Fourier Feature Embedding (FFE). We evaluate its benefits when applied to the previously introduced training strategies.

5. Experimental design

This section describes the methods included in the experimental analysis. All the methods have been applied to a standard MultiLayer Perceptron (MLP) with 4 hidden layers of 50 units each and an Adam optimizer with a fixed learning rate of $5 \cdot 10^{-4}$. For each use case, we define a mesh $\mathcal{D} \subset \Omega \times [0, T]$ in the space and time domains. Initial and boundary conditions components of the loss function are evaluated on all the related points of the mesh. The PDE residuals are calculated on a random sample, which is a subset of fixed dimension N_f, $\mathcal{D}_f \subset \mathcal{D}$, of the inner domain (i.e., excluding initial and boundary points). These samples are passed to the network as a single batch at each epoch. We define a fixed number of training epochs for each use case, at which all the methods yield no significant further improvements. As an additional hyper-parameter, we then added the number of hidden random samples N_f in the inner mesh. We tested the performance of all the methods by also varying this parameter.

We experimentally observed that the error rate of the neural networks for the baseline solution tends to reach a plateau, either close to zero or significantly far from it, reaching a local non-optimal minimum. Within the number of epochs we fixed, we observed that the models' results could have (i) reached convergence with a reasonably low error rate, (ii) converged to a wrong solution, or (iii) approximated the right solution, but still far from it, i.e., with a large error. On the basis of this intuition, from empirical evidence, we fixed a convergence threshold at 0.01, considered as the "reasonably low" error rate. Such threshold allows us to binary evaluate the converge, i.e., to determine whether a model reached convergence or not. We empirically noticed that most of the runs tend to be significantly higher or lower than the threshold value, making our results quite stable with respect to this choice.

We evaluate each learning method according to the following metrics.

Convergence rate. This is the ratio of experiments that reach convergence (i.e., error rate below the threshold) with the same configuration for different random initialisations. This metric can be thought of as the robustness of the method with respect to the initial conditions. We refer to it as σ_{conv} in the following.

Prediction error. This is computed as the MSE of each model, for all the points of the mesh $\mathcal{D}$, as

$$\epsilon_{\text{pred}} = \mathbb{E}\left[|\hat{u} - U_{base}|^2\right] \tag{8}$$

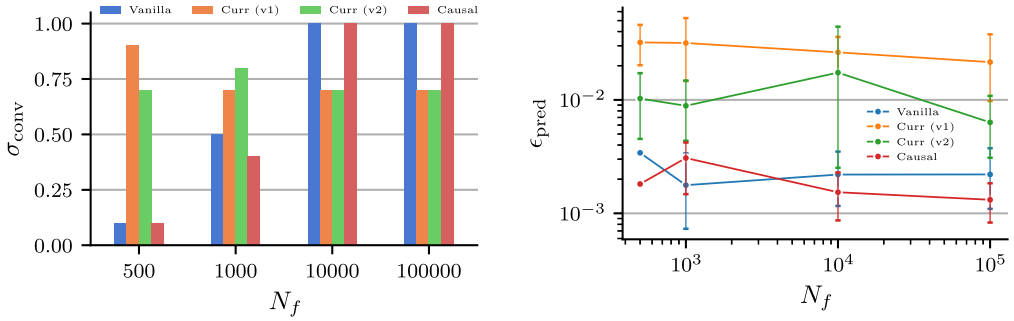
where $\mathbb{E}[\cdot]$ denotes the average over the space $\mathcal{D}$, and U_{base} is the baseline solution.

The above-mentioned methods are applied to 3 use cases represented by a different PDE each. The PDEs have been chosen among the most popular in the PINN literature, and with increasing complexity. They are the convection equation (Section 6.1), the reaction-diffusion equation (Section 6.2), and the Allen-Cahn equation (Section 6.3).

6. Experimental results

6.1. Convection equation

The first benchmark use case we consider is the convection equation. This system is generally used to model heat transfer or transport phenomena. Having β as the convection coefficient, we defined the problem with periodic boundary conditions as:



(a) Convergence rate over training size.

(b) Prediction error over training size.

Fig. 4. Metrics evaluation for the convection system.

$$\begin{aligned} &\frac{\partial u}{\partial t} + \beta \frac{\partial u}{\partial x} = 0, \quad x \in [0, 2\pi],\ t \in [0, 1] \\ &u(x, 0) = \sin(x) \end{aligned} \tag{9}$$

The parameter β is set to 30 to get a *complex* setting (as discussed in Section 4.1).

Since our experiments reported that the encoding input strategy (Fourier Feature Embedding FFE) was completely ineffective, i.e., none of the experiments reached convergence, we discarded experimental results with FFE for the rest of the discussion on this use case.

We trained each model for 27,500 epochs, with curriculum learning performed in 10 steps. This training duration has been fixed on the basis of the performance reached by the Vanilla method with $N_f = 1000$.

Fig. 4a reports the convergence rate for each learning strategy and for different numbers of training samples N_f. Vanilla shows to be quite unstable for different values of N_f. Half of the experiments with Vanilla PINN reached convergence with $N_f = 1000$. This result decreases abruptly (-80%) when the training size is reduced only by 50% ($N_f = 500$). Instead, all the runs reached convergence by increasing N_f by one order of magnitude ($N_f = 10000$).

Curriculum learning (in both versions v1 and v2) is, instead, significantly more stable in response to variations in training size, achieving scores between 70% and 90%. Causal training is comparable to Vanilla PINN, providing no improvements in this use case.

Fig. 4b shows the prediction error for each learning strategy and for different N_f values, by considering only models that reached convergence at the end of the training. Vanilla and Causal methods confirm to be comparable over all the training sizes N_f. Hence, in this use case, causal learning provides no significant improvements. Even if Curriculum learning achieved higher convergence rates for low N_f (see Fig. 4a), it performs worse than both Vanilla and Causal in terms of prediction error, for all values of N_f, with the newly introduced version v2 being generally better than the state-of-the-art version v1.

In summary, curriculum regularisation experiments (mainly v1) reach good stability across different values of training size, but pay in terms of prediction error. Our newly introduced Curriculum version v2 presents a sort of intermediate behaviour between Vanilla and Curriculum version v1 learning, showing how such a small change in the learning can have a significant impact on the final PINN performance.

The fact that Curriculum strategies present a convergence rate that does not increase with greater N_f values, together with an overall worse prediction error, suggests that the defined training duration might be insufficient to reach convergence with the evaluated training samples. One possible explanation is that increasing the number of training data points could make the convergence more epoch-demanding. Then, any curriculum step would conclude slightly further from the optimal minimum, resulting in significant performance degradation at the end of the training. To investigate this intuition, we report the worst-case prediction (in terms of final MSE) in Fig. 5. Vanilla and Causal reached a plateau, hence extending the training to more epochs would not lead to improvements. On the contrary, both Curriculum versions present no plateau, hence further training with more epochs might help them in reaching better results, i.e., a lower prediction error rate.

6.1.1. Longer learning

Our experimental design sets the number of epochs to a fixed value for all strategies. As reported in Fig. 5, for Curriculum learning approaches, this choice might limit their performance. In this section we investigate whether Curriculum approaches could be improved with a longer training phase. Fig. 6 shows the resulting prediction error when training each model for 40,000 epochs, with both versions of Curriculum regularisation. The grey curve shows the baseline, i.e., the average result of the Vanilla and Causal models. All the experiments reached convergence with all the training sizes, but without getting significant improvements with respect to the previous best solutions. Curriculum version v2 is again better than version v1, however we exclude that the Curriculum regularization learning can benefit from longer training, contrary to our initial intuition.

6.2. Reaction-diffusion equation

The reaction-diffusion equation is another system in which vanilla PINNs generally fail [27]. Such an equation can be designed to map the behaviour of density variations of chemical systems, population growth, etc. Our formulation is reported in Equation (10), where $\mathcal{N}(\pi, \pi/4)$ is a gaussian with mean π and standard deviation $\pi/4$ and the parameters have been set to $\nu = 5,\ \rho = 5$.

$$\begin{aligned} &\frac{\partial u}{\partial t} - \nu \frac{\partial^2 u}{\partial x^2} - \rho u(1 - u) = 0, \quad x \in [0, 2\pi],\ t \in [0, 1] \\ &u(x, 0) = \mathcal{N}(\pi, \pi/4) \end{aligned} \tag{10}$$

The two parameters ν and ρ are associated, respectively, with the diffusion and reaction components of the equation, in the sense that setting ν to zero would lead to fully reactive dynamics, while $\rho = 0$ would imply a fully diffusive one. The presence of two parameters makes the curriculum learning strategy not directly applicable, but the semantic separability of the two behaviours suggested to keep fixed one of the two parameters while letting the other approach zero, and contextually measuring the vanilla PINN performance. We found out that the regular model was more effective when addressing a problem with a lower reactive contribution with respect to the contrary (i.e., lower diffusive contribution). Then we set ρ as the curriculum variable parameter, letting it increase to the desired value with the running epochs.

For this use case, we fixed the training epochs to 50,000, which we experimentally found as a good point in which most of the models reached a plateau.

Fig. 7a shows that all Curriculum and Causal learning experiments were able to reach convergence, while none of the Vanilla models were below our convergence threshold, with the only exception of an experiment trained at $N_f = 10^5$. Regarding FFE, we can see that it yields to

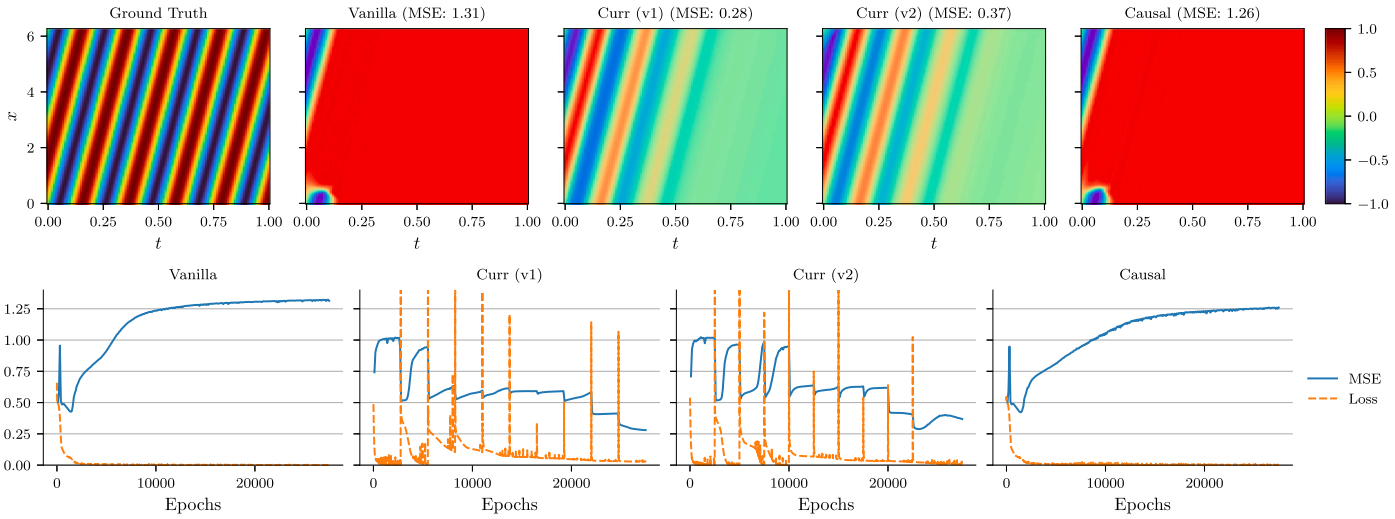


Fig. 5. Worst case prediction for each method. The upper row shows the output of the trained network, the lower row reports the total loss and the mean squared error over the training epochs.

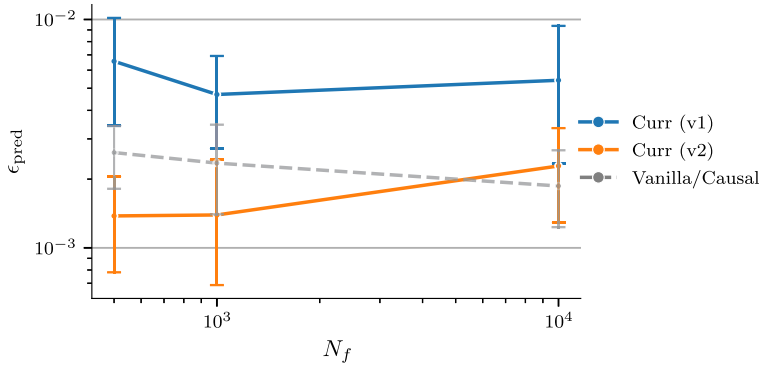
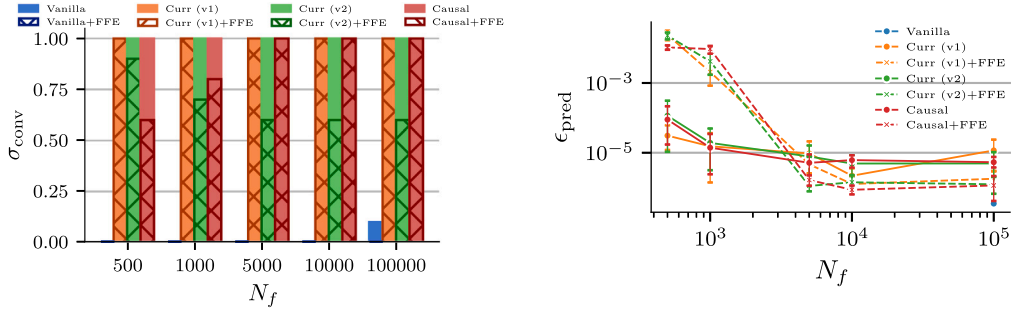


Fig. 6. Prediction error for curriculum alternatives trained for 40000 epochs.



(a) Convergence rate over training size.

(b) Prediction error over training size.

Fig. 7. Metrics evaluation for the reaction-diffusion system.

lower values of convergence rate for most of the training sizes with respect to models without FFE.

Fig. 7b reports the prediction error rates. All methods improve their performance for $N_f = 5000$ with respect to lower N_f values, largely for FFE methods, and slightly for non-FFE methods. FFE yields to much poorer performance for $N_f = 1000$ and below, with respect to non-FFE methods. On the contrary, for higher N_f values, from 10^4 and above, the FFE methods yield better performance, with lower prediction error rates than non-FFE ones.

Both Curriculum strategies (v1 and v2), and Causal learning present comparable trends. Hence, in this use case, the learning approach is not a crucial choice. The most relevant choice is instead the usage of FFE, which is detrimental for low N_f values (1000 and below), and beneficial for high N_f values (5000 and above). Hence, the best solutions to the reaction-diffusion equation are those without FFE for small training sizes, but those exploiting FFE for large training sizes, with the latter yielding the absolute best performance.

To integrate the quantitative experiments, we report in Fig. 8 a sample qualitative error map for each configuration.

6.3. Allen-Cahn equation

One popular use case for PINN benchmarking is the Allen-Cahn equation. This PDE is another form of reaction-diffusion equation, and

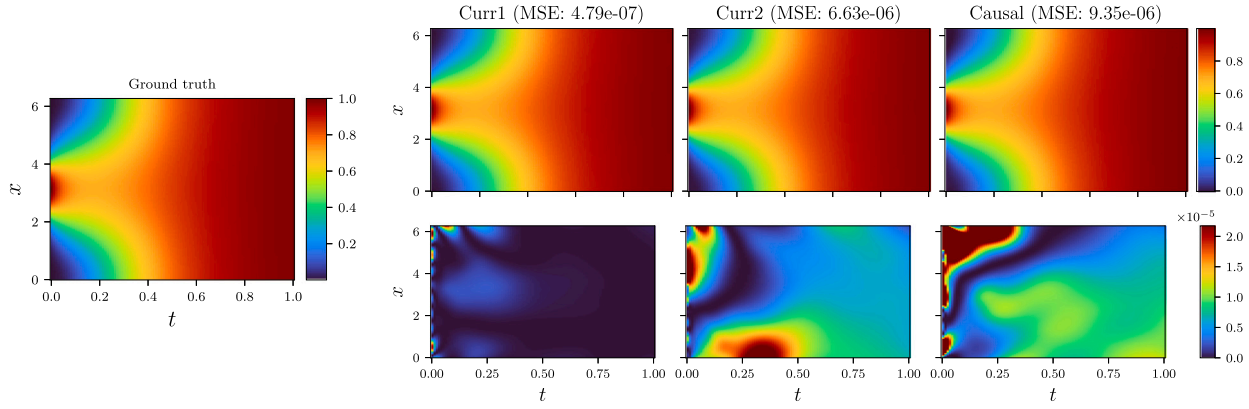
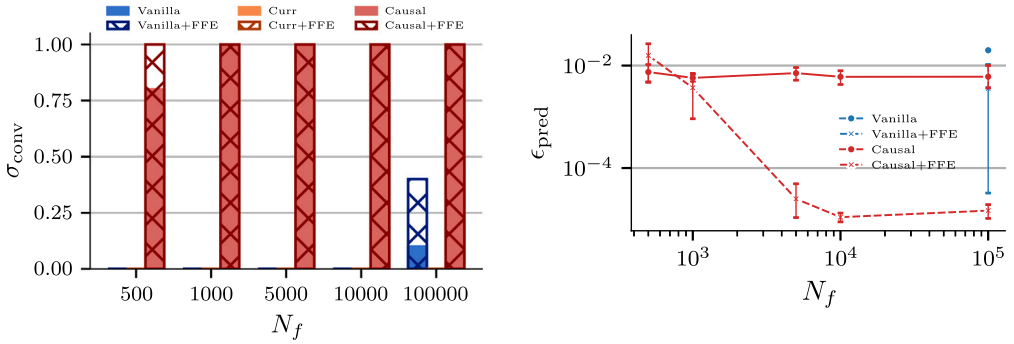


Fig. 8. Prediction error maps for each experiment.



(a) Convergence rate over training size.

(b) Prediction error over training size.

Fig. 9. Metrics evaluation for the Allen-Cahn system.

describes the phase separation in multi-component alloy systems, as reported in the following.

$$\frac{\partial u}{\partial t} + \rho u(u^2 - 1) - \nu \frac{\partial^2 u}{\partial x^2} = 0, \quad x \in [-1, 1], t \in [0, 1]$$
$$u(x, 0) = x^2 \cos(\pi x). \tag{11}$$

We solve this equation for $\nu = 0.0001$ and $\rho = 5$. This equation use case proves to be harder to address by PINNs than the previous ones. The first difference from the reaction-diffusion use case is that the Vanilla PINN cannot solve the Allen-Cahn equation even with a low value of ν. Furthermore, we could not find a valuable starting point for Curriculum learning with any combination of parameters obtained by fixing ν and varying ρ or vice versa. Nevertheless, for the sake of comparison, we follow a curriculum regularization on the parameter ρ as in the reaction-diffusion use case.

We performed the experiments for 150,000 training epochs, which is much larger than all previous use cases. The number of epochs is based on the convergence point of the Causal training models. Due to the complexity of the problem, we perform our analyses at higher values of N_f with respect to the previous use cases.

Fig. 9a reports the results in terms of convergence rate, for all the methods, also including the FFE option. Causal training always reaches convergence, for any N_f value, both with and without FFE. Instead, only for the largest value of $N_f = 10^5$ the Vanilla PINN reached a convergence rate higher than zero. Vanilla also benefits from FFE, which more than doubles its convergence rate, but its performance is still much worser than Causal training models: 40% of Vanilla vs 100% of Causal ($N_f = 10^5$). Curriculum learning, in both versions, is not able to address this problem, and indeed its convergence rate values are almost invisible in the plot, being zero in all cases.

The benefits of applying FFE to this system are also reflected in Fig. 9b, which shows a large increase in performance with much lower prediction error rates, starting from $N_f > 10^3$, with respect to non-FFE models. In terms of prediction error rates, plain causal learning models yield the same performance independently from the training set size, whereas exploiting FFE improves their performance by more than two orders of magnitude for $N_f = 5000$ and higher.

To provide an insight into these behaviours, Fig. 10 shows an example in which both Vanilla and Causal learning models reach convergence. Vanilla PINN's solution presents broader error regions, starting at early time steps, whereas the Causal learning model clearly provides better predictions, that are very similar to the ground truth.

7. Discussion

In Section 6, we observed that no single model can provide the best results in all the use cases. In some use cases, a specific approach might be the best in all settings. As an example, we mention the Allen-Cahn equation, for which the Causal learning is always the best performer, and the FFE is (almost) always beneficial, often by a large factor. Indeed the Causal learning is also the only method to reach convergence for the Allen-Cahn equation.

However, in most use cases, different learning methods provide the best results depending on the training size, or the usage of FFE, even within the same use case.

To provide an overview of the best learning strategies for each experimental setting, we define an *overall score* by considering both convergence rate and average prediction error. Our *overall score* is given by the product of the former and the negated logarithm of the latter, as reported in Equation (12).

$$SCORE = -\log(\epsilon_{\text{pred}}) \cdot \sigma_{\text{conv}}. \tag{12}$$

Table 1 summarizes all the *overall scores* for each experiment and training size, grouped into small ($N_f < 10^3$), medium, and large ($N_f >$

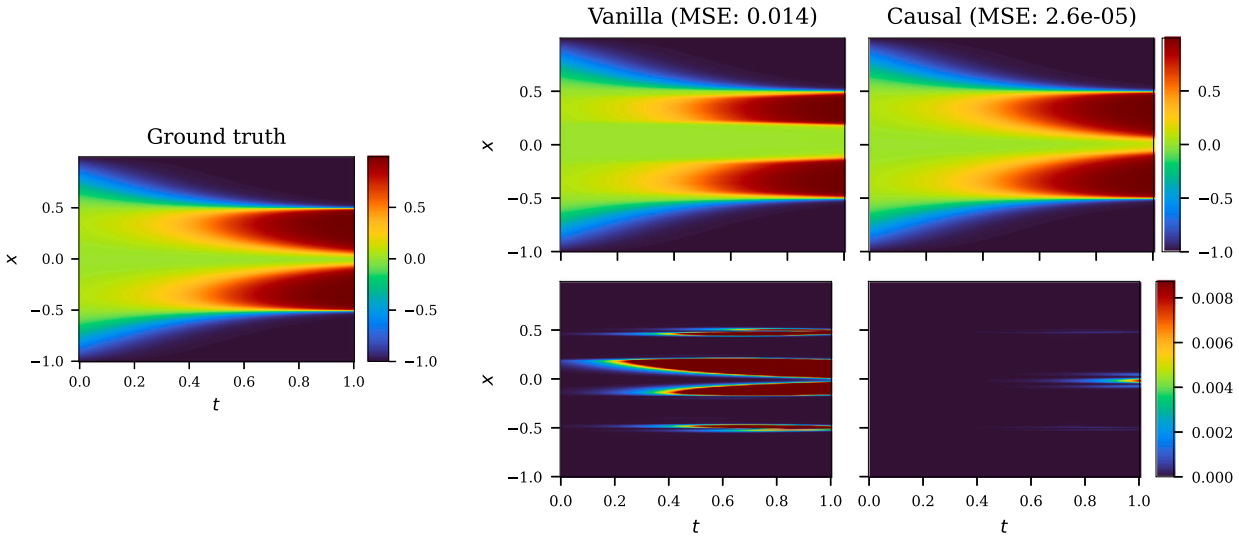


Fig. 10. Prediction error maps for each experiment (without applying FFE) for the Allen-Cahn system.

Table 1
Overall score results for all use cases, learning strategies, and training sizes. Training sizes have been grouped into small ($N_f < 10^3$), medium, and large ($N_f > 10^5$).

System	Convection			Reaction-diffusion			Allen-Cahn		
Training size	Small	Medium	Large	Small	Medium	Large	Small	Medium	Large
Vanilla	0.25	**2.02**	2.66	-	-	0.65	-	-	0.17
Curr (v1)	**1.34**	1.08	1.17	**4.51**	**5.05**	4.93	-	-	-
Curr (v2)	**1.39**	1.42	1.54	3.94	**4.97**	5.31	-	-	-
Causal	0.27	1.89	**2.88**	4.05	**5.07**	5.27	1.70	2.20	2.22
Vanilla + FFE	-	-	-	-	-	-	-	-	0.98
Curr (v1) + FFE	-	-	-	1.62	3.16	5.75	-	-	-
Curr (v2) + FFE	-	-	-	1.48	1.78	3.54	-	-	-
Causal + FFE	-	-	-	1.19	2.38	**5.94**	**1.80**	**3.98**	**4.83**

Table 2
Best learning strategy for each use case and training size based on the *overall score*.

	Small ($N_f < 10^3$)	Medium	Large ($N_f > 10^4$)
Convection	Curr (v2)	Vanilla	Causal
Reaction-diffusion	Curr (v1)	Curr/Causal	Causal + FFE
Allen-Cahn	Causal + FFE	Causal + FFE	Causal + FFE

10^5). For each use case (in the columns), we highlight in bold the best results. In case of minor differences, all strategies with the highest scores are highlighted.

While reporting the best learning strategies for each configuration in Table 2, we notice that the outcome is significantly variegated:

- FFE tends yields very large improvements for complex problems (Allen-Cahn equation), but it completely fails for the convection equation, and its benefits are very limited for the reaction-diffusion one.
- Causal learning reaches a high score in most cases, but not for small training sizes, and only for large training sizes in the convection equation.
- The Causal learning strategy seems to become more and more suitable as the complexity of the equation increases, such as in the reaction-diffusion and in the Allen-Chan use cases.
- Curriculum regularisations are promising, but in some use cases they did not reach convergence at all (e.g., Allen-Cahn).
- Curriculum regularizations seem to be more suitable for low-complexity problems, such as the convection equation, and small training sizes.

At the current state of the art, training PINNs requires many intuitions to be effectively applied to each use case, and the application of a specific learning strategy is not straightforward. All in all, there is no clear winner, no learning to rule them all.

8. Conclusions

This paper shows how PINNs generally face many difficulties in finding the correct solution for different tasks. Even the best state-of-the-art strategies, such as Curriculum regularisation and Causal training, present pitfalls and can yield from best to worst results by simply changing the equation under analysis or by adding the usage of FFE. At the moment, no learning strategy can be safely applied to all use cases. The results of our experiments underscored the need for a broader approach to address these limitations.

Moreover, we introduced a new design of the Curriculum learning algorithm, revealing how a small variation in its formulation can significantly change the performance depending on the task. This emphasizes how any possible variation on the baseline could in principle apport benefits on different tasks.

As future work, we plan to design and develop an automatic approach to determine which Curriculum steps are needed and their length, and possibly join this strategy with Causal learning to take advantage of both.

CRediT authorship contribution statement

Simone Monaco: Conceptualization, Formal analysis, Investigation, Methodology, Software, Visualization; **Daniele Apiletti:** Funding acqui-

sition, Project administration, Resources, Supervision, Validation; **Both authors:** Roles/Writing – original draft; Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgement

This work is part of the project NODES, funded by the Italian MUR (Ministry of University and Research) under M4C2 1.5 of the PNRR (National Plan for Recovery and Resilience) with grant agreement no. ECS00000036. This work has been also partially funded by the Smart-Data@PoliTO research centre of Politecnico di Torino, Italy.

References

[1] B. Alipanahi, A. Delong, M.T. Weirauch, B.J. Frey, Predicting the sequence specificities of dna- and rna-binding proteins by deep learning, Nat. Biotechnol. 33 (8) (2015) 831–838.

[2] P. Baldi, P. Sadowski, D. Whiteson, Searching for exotic particles in high-energy physics with deep learning, Nat. Commun. 5 (1) (2014) 1–9.

[3] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al., Mastering the game of go without human knowledge, Nature 550 (7676) (2017) 354–359.

[4] A. Karpatne, G. Atluri, J. Faghmous, M. Steinbach, A. Banerjee, A. Ganguly, S. Shekhar, N. Samatova, V. Kumar, Theory-guided data science: a new paradigm for scientific discovery from data, IEEE Trans. Knowl. Data Eng. 29 (10) (2017) 2318–2331.

[5] G.E. Karniadakis, I.G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, Nat. Rev. Phys. 3 (6) (2021) 422–440, https://doi.org/10.1038/s42254-021-00314-5, https://www.nature.com/articles/s42254-021-00314-5, number: 6, Publisher: Nature Publishing Group.

[6] L. Von Rueden, S. Mayer, K. Beckh, B. Georgiev, S. Giesselbach, R. Heese, B. Kirsch, J. Pfrommer, A. Pick, R. Ramamurthy, M. Walczak, J. Garcke, C. Bauckhage, J. Schuecker, Informed machine learning – a taxonomy and survey of integrating knowledge into learning systems, IEEE Trans. Knowl. Data Eng. (2021) 1, arXiv:1903.12394.

[7] C. Rackauckas, Y. Ma, J. Martensen, C. Warner, K. Zubov, R. Supekar, D. Skinner, A. Ramadhan, A. Edelman, Universal differential equations for scientific machine learning, arXiv:2001.04385 [cs, math, q-bio, stat], https://doi.org/10.48550/arXiv.2001.04385, http://arxiv.org/abs/2001.04385, Nov. 2021.

[8] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707, https://doi.org/10.1016/j.jcp.2018.10.045, https://www.sciencedirect.com/science/article/pii/S0021999118307125.

[9] Y. Yang, P. Perdikaris, Adversarial uncertainty quantification in physics-informed neural networks, arXiv:1811.04026 [physics, stat], https://doi.org/10.1016/j.jcp.2019.05.027, http://arxiv.org/abs/1811.04026, Nov. 2018.

[10] E. Kharazmi, Z. Zhang, G.E.M. Karniadakis, hp-VPINNs: variational physics-informed neural networks with domain decomposition, Comput. Methods Appl. Mech. Eng. 374 (2021) 113547, https://doi.org/10.1016/j.cma.2020.113547, https://www.sciencedirect.com/science/article/pii/S0045782520307325.

[11] G. Kissas, Y. Yang, E. Hwuang, W.R. Witschey, J.A. Detre, P. Perdikaris, Machine learning in cardiovascular flows modeling: predicting arterial blood pressure from non-invasive 4d flow mri data using physics-informed neural networks, Comput. Methods Appl. Mech. Eng. 358 (2020) 112623.

[12] M. Raissi, A. Yazdani, G.E. Karniadakis, Hidden fluid mechanics: learning velocity and pressure fields from flow visualizations, Science 367 (6481) (2020) 1026–1030.

[13] Y. Chen, L. Lu, G.E. Karniadakis, L. Dal Negro, Physics-informed neural networks for inverse problems in nano-optics and metamaterials, Opt. Express 28 (8) (2020) 11618–11633.

[14] S. Cai, Z. Mao, Z. Wang, M. Yin, G.E. Karniadakis, Physics-informed neural networks (pinns) for fluid mechanics: a review, Acta Mech. Sin. (2022) 1–12.

[15] X. Chen, Y. Tian, Learning to perform local rewriting for combinatorial optimization, Adv. Neural Inf. Process. Syst. 32 (2019) 00112, https://proceedings.neurips.cc/paper/2019/hash/131f383b434fdf48079bff1e44e2d9a5-Abstract.html.

[16] S. Wang, S. Sankaran, P. Perdikaris, Respecting causality is all you need for training physics-informed neural networks, arXiv:2203.07404 [nlin, physics:physics, stat], http://arxiv.org/abs/2203.07404, Mar. 2022.

[17] L. McClenny, U. Braga-Neto, Self-adaptive physics-informed neural networks using a soft attention mechanism, arXiv:2009.04544 [cs, stat], http://arxiv.org/abs/2009.04544, Apr. 2022.

[18] S. Wang, Y. Teng, P. Perdikaris, Understanding and mitigating gradient flow pathologies in physics-informed neural networks, SIAM J. Sci. Comput. 43 (5) (2021) A3055–A3081, https://doi.org/10.1137/20M1318043, https://epubs.siam.org/doi/10.1137/20M1318043.

[19] A.D. Jagtap, K. Kawaguchi, G.E. Karniadakis, Adaptive activation functions accelerate convergence in deep and physics-informed neural networks, J. Comput. Phys. 404 (2020) 109136.

[20] A.D. Jagtap, G.E. Karniadakis, Extended physics-informed neural networks (XPINNs): a generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations, Commun. Comput. Phys. 28 (5) (2020) 2002–2041.

[21] I. Lagaris, A. Likas, D. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE Trans. Neural Netw. 9 (5) (1998) 987–1000, https://doi.org/10.1109/72.712178, conference name: IEEE Transactions on Neural Networks.

[22] L.P. Aarts, P. Van Der Veer, Neural network method for solving partial differential equations, Neural Process. Lett. 14 (3) (2001) 261–271.

[23] M. Kumar, N. Yadav, Multilayer perceptrons and radial basis function neural network methods for the solution of differential equations: a survey, Comput. Math. Appl. 62 (10) (2011) 3796–3811.

[24] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics informed deep learning (part I): data-driven solutions of nonlinear partial differential equations, arXiv:1711.10561 [cs, math, stat], version: 1, http://arxiv.org/abs/1711.10561, Nov. 2017.

[25] C.L. Wight, J. Zhao, Solving Allen-Cahn and Cahn-Hilliard equations using the adaptive physics informed neural networks, arXiv preprint, arXiv:2007.04542, 2020.

[26] S. Wang, X. Yu, P. Perdikaris, When and why PINNs fail to train: a neural tangent kernel perspective, arXiv:2007.14527, http://arxiv.org/abs/2007.14527, Jul. 2020.

[27] A. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby, M.W. Mahoney, Characterizing possible failure modes in physics-informed neural networks, in: Advances in Neural Information Processing Systems, vol. 34, Curran Associates, Inc., 2021, pp. 26548–26560, https://proceedings.neurips.cc/paper/2021/hash/df438e5206f31600e6ae4af72f2725f1-Abstract.html.

[28] S. Dong, N. Ni, A method for representing periodic functions and enforcing exactly periodic boundary conditions with deep neural networks, J. Comput. Phys. 435 (2021) 110242.