Incremental learning for physics-informed neural networks

Aleksandr Dekhovich
Delft University of Technology

Marcel H.F. Sluiter
Delft University of Technology

David M.J. Tax
Delft University of Technology

Miguel A. Bessa *
Brown University

Abstract

This work proposes an incremental learning algorithm for physics-informed neural networks (PINNs), which have recently become a powerful tool for solving partial differential equations (PDEs). As demonstrated herein, by developing incremental PINNs (iPINNs) we can effectively mitigate training challenges associated with PINNs loss landscape optimization and learn multiple tasks (equations) sequentially without additional parameters for new tasks. Interestingly, we show that this also improves performance for every equation in the sequence. The approach is based on creating its own subnetwork for each PDE and allowing each subnetwork to overlap with previously learned subnetworks. We also show that iPINNs achieve lower prediction error than regular PINNs for two different scenarios: (1) learning a family of equations (e.g., 1-D convection PDE); and (2) learning PDEs resulting from a combination of processes (e.g., 1-D reaction-diffusion PDE). The code implementation of the work is available at `https://github.com/adekhovich/incremental_PINNs`.

1 Introduction

Deep neural networks (DNNs) play a central role in scientific machine learning (SciML). Recent advances in neural networks find applications in real-life problems in physics [1, 2, 3, 4], medicine [5, 6, 7], finance [8, 9, 10, 11], and engineering [12, 13, 14, 15]. In particular, they are also applied to solve Partial Differential Equations (PDEs) [16, 17, 18, 19]. Consider the following PDE,

$$\mathcal{F}[u(\mathbf{x},t)] = f(\mathbf{x}), \quad \mathbf{x} \in \Omega,\ t \in (t_0, T], \tag{1}$$
$$\mathcal{B}[u(\mathbf{x},t)] = b(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega, \tag{2}$$
$$u(\mathbf{x},t_0) = h(\mathbf{x}), \quad \mathbf{x} \in \Omega, \tag{3}$$

where $\mathcal{F}$ is a differential operator, $\mathcal{B}$ is a boundary condition operator, $h(\mathbf{x})$ is an initial condition, and Ω is a bounded domain. Physics-informed neural networks (PINNs)[19] incorporate initial and boundary conditions as soft constraints into the loss function of a DNN. Let us denote the output of the network $\mathcal{N}$ with learnable parameters θ as $\hat{u}(\theta, \mathbf{x}, t) = \mathcal{N}(\theta; \mathbf{x}, t)$. Then sampling the set of collocation points, i.e. a set of points in the domain, $\mathcal{CP} = \{(x^i, t^i) : x^i \in \text{int}\ \Omega,\ t^i \in (t_0, T],\ i = 1, 2, \dots N_{\mathcal{F}}\}$, the set of initial points $\mathcal{IP} = \{(x^j, t_0) : x^j \in \partial\Omega,\ j = 1, 2, \dots, N_{u_0}\}$ and the set of boundary points $\mathcal{BP} = \{(x^k, t^k) : x^k \in \partial\Omega,\ t^k \in (t_0, T],\ k = 1, 2, \dots, N_b\}$ one can write the optimization problem and loss function arising from PINNs as follows:

*Corresponding author, e-mail: `miguel_bessa@brown.edu`

Machine Learning and the Physical Sciences Workshop, NeurIPS 2023.

$$\mathcal{L}(\theta) = \mathcal{L}_{\mathcal{F}}(\theta) + \mathcal{L}_{u_0}(\theta) + \mathcal{L}_b(\theta) \to \min_{\theta}, \tag{4}$$

$$\mathcal{L}_{\mathcal{F}}(\theta) = \frac{1}{N_{\mathcal{F}}} \sum_{i=1}^{N_{\mathcal{F}}} \left|\left|\mathcal{F}[\hat{u}(\theta, x^i, t^i)] - f(x^i)\right|\right|^2, \quad (x^i, t^i) \in \mathcal{CP}, \tag{5}$$

$$\mathcal{L}_{u_0}(\theta) = \frac{1}{N_{u_0}} \sum_{j=1}^{N_{u_0}} \left|\left|\hat{u}(\theta, x^j, t_0) - h(x^j)\right|\right|^2, \quad (x^j, t_0) \in \mathcal{IP}, \tag{6}$$

$$\mathcal{L}_b(\theta) = \frac{1}{N_b} \sum_{k=1}^{N_b} \left|\left|\mathcal{B}[\hat{u}(\theta, x^k, t^k)] - b(x^k)\right|\right|^2, \quad (x^k, t^k) \in \mathcal{BP}. \tag{7}$$

However, sometimes PINNs struggle to learn the ODE/PDE dynamics [20, 21, 22, 23]. Wight & Zhao [24] proposed several techniques to improve the optimization process compared to the original formulation: mini-batch optimization and adaptive sampling of collocation points. Krishnapriyan et al. [21] proposed the *seq2seq* approach that splits the domain into smaller subdomains in time and learns the solution on each of the subdomains with a separate network. Thus, both adaptive sampling in time and *seq2seq* are based on the idea of splitting the domain into multiple subdomains, on which solutions can be learned more easily. As explained in [22], improving PINN's solutions by considering small subdomains is possible because the loss residuals ($\mathcal{L}_{\mathcal{F}}$ term) can be trivially minimized in the vicinity of fixed points, despite corresponding to nonphysical system dynamics that do not satisfy the initial conditions. Therefore, the reduction of the domain improves the convergence of the optimization problem (4) and helps to escape nonphysical solutions.

Despite the popularity of PINNs in physical sciences [25, 26, 27], there is no incremental learning procedure for PINNs. Thus, we propose *incremental PINNs* (iPINNs) – a strategy for training PINNs incrementally, creating task-specific subnetwork for every PDE. Each subnetwork $\mathcal{N}_i$ has its own set of parameters $\theta_i \subset \theta$, and the model is trained sequentially on different tasks. A subnetwork for a new task can overlap with all previous subnetworks, which helps to assimilate the new task. As a result, the network consists of overlapping subnetworks, while the free parameters can be used for future tasks. The network uses pretrained parts (subnetworks) to find and train the next (possibly overlapping) subnetwork that cannot be learned well with regular PINN. To the best of our knowledge, this is the first example where one network can sequentially learn multiple equations without extending its architecture, with the added benefit that performance is significantly improved.

2 Problem formulation

We focus on two scenarios: (1) incremental PINNs learning, where the network sequentially learns several equations from the same family; and (2) learning a combination of multiple equations that create another physical process. To illustrate these cases, we consider one-dimensional convection and reaction-diffusion problems with periodic boundary conditions.

Scenario 1: 1-D convection equation

$$\begin{aligned} \frac{\partial u}{\partial t} + \beta_k \frac{\partial u}{\partial x} &= 0, \qquad \text{(P1)} \\ u(x, 0) &= \sin x, \\ u(0, t) &= u(2\pi, t), \end{aligned}$$

where $t \in (0, 1],\ x \in [0, 2\pi],\ \beta_k \in \mathcal{B} \subset \mathbb{R}$.

Scenario 2: 1-D reaction-diffusion equation

$$\begin{aligned} \frac{\partial u}{\partial t} - \nu \frac{\partial^2 u}{\partial x^2} - \rho u(1-u) &= 0, \qquad \text{(P2)} \\ u(x, 0) &= e^{-\frac{(x-\pi)^2}{2(\pi/4)^2}}, \\ u(0, t) &= u(2\pi, t), \end{aligned}$$

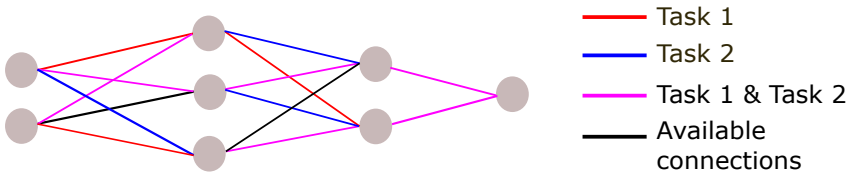
where $t \in (0, 1],\ x \in [0, 2\pi],\ \nu, \rho > 0$. This process consists of two parts: reaction term ($\nu = 0$): $-\rho u(1-u)$ and diffusion term ($\rho = 0$): $-\nu \frac{\partial^2 u}{\partial x^2}$.

In the second scenario, we construct one task as the reaction, another one as the diffusion, and the final one as the reaction-diffusion. We can change the order of the reaction tasks and diffusion tasks to show the robustness of incremental learning. Considering these two problems, we want to show that

better generalization can be achieved by pretraining the network with simpler related problems. In the following section, we show how one network can incrementally learn different equations without catastrophic forgetting.

3 Proposed method

An incremental learning method needs to be applicable to both types of problems P1 and P2. However, these problems cannot be solved by one network with the same output head for every different task, since $\mathcal{F}_i[u(x,t)] \neq \mathcal{F}_j[u(x,t)]$ for $i \neq j$ and $x \in \Omega,\ t \in [t_0, T]$. Therefore, we propose iPINNs – an incremental learning algorithm that focuses on learning task-specific subnetworks $\mathcal{N}_1, \mathcal{N}_2, ..., \mathcal{N}_k, ...$ for each task k. The algorithm starts by creating the above-mentioned subnetworks. Here we use an iterative pruning algorithm that is called NNrelief [28]. We select NNrelief because it achieves the highest number of pruned parameters (connections) reported to date for different state-of-the-art neural network architectures trained on MNIST, CIFAR-10/100 and Tiny-ImageNet.



$$\mathcal{L}(\theta; \mathcal{D}_1 \cup \mathcal{D}_2) = \mathcal{L}_1(\theta_1; \mathcal{D}_1) + \mathcal{L}_2(\theta_2; \mathcal{D}_2)$$

Figure 1: An example of iPINNs with two PDEs.

The total loss and its gradient with respect to a parameter w can be written as:

$$\mathcal{L} = \sum_{j=1}^{k} \mathcal{L}_j, \tag{8}$$

$$\frac{\partial \mathcal{L}}{\partial w} = \sum_{j=1}^{k} \frac{\partial \mathcal{L}_j}{\partial w} = \sum_{j:\ w \in \mathcal{N}_j} \frac{\partial \mathcal{L}_j}{\partial w}, \tag{9}$$

because if $w \notin \mathcal{N}_j$, then $\frac{\partial \mathcal{L}_j}{\partial w} = 0$.

Algorithm 1 PINN incremental learning: adding new task k

Require: neural network $\mathcal{N}$, training datasets $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_{k-1}$ and $\mathcal{D}_k$, training hyperparameters, pruning hyperparameters (num_iters).

1: $\mathcal{N}_k \leftarrow \mathcal{N}$ ▷ set full network as a subnetwork
2: Train $\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_k$ on tasks $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_k$ using Eq. 9. ▷ training step
3: **for** $it = 1, 2, \ldots, num\_iters$ **do** ▷ repeat pruning
4: $\quad \mathcal{N}_k \leftarrow Pruning(\mathcal{N}_k, \mathcal{D}_k)$ ▷ pruning step
5: $\quad$ Retrain subnetworks $\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_k$ on tasks $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_k$ using Eq. 9. ▷ retraining step
6: **end for**

An important concept in the proposed iPINN strategy is that the pruning method is used to train task-specific subnetworks, but allowing the subnetworks to *naturally* overlap on some connections (see Figure 1). This way the method provides knowledge sharing between the subnetworks. These overlaps are updated with respect to all tasks that are assigned to a particular connection. Thus, for every new task k that enters the network, we first find a corresponding subnetwork $\mathcal{N}_k$ with NNrelief (line 4 of the Algorithm 1), then adapt the overlaps between previous subnetworks $\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_{k-1}$ and a new one $\mathcal{N}_k$ (line 5 of the Algorithm 1). The main advantage of the proposed approach is that a neural network learns *all* tasks that were given during training and not only the last one. This is achieved by constantly replaying old data. Data for previous tasks is easily available by sampling collocation points, which eliminates all issues of data replaying for continual learning problems

in computer vision and natural language processing tasks and makes the algorithm well-suited in the context of PINNs. We want to emphasize that iPINN does not need to know how many tasks will be given overall, and it has access only to those that were given up to task k inclusive, which distinguishes it from multi-task learning. In the next section, we experimentally show that pretrained parts of the network help to improve the convergence process.

4 Numerical experiments

Experiments setup. We use a four-layer neural network with 50 neurons per layer and select 1000 randomly collocation points on every time interval between 0 and 1 for $\mathcal{L}_{\mathcal{F}}$. The Adam optimizer [29] with a learning rate of 0.01 and 20000 epochs are used to train the model. We divide the learning rate by 3 every 500 epochs in which the loss does not decrease. We repeat our experiments multiple times with different random initializations of the network parameters and show the average values of error. In addition, following continual learning literature [30], we compare backward transfer metrics. Let us denote the test set as $\mathcal{D}^{test} = \{(x^i, t^i, l) : x^i \in [0, 2\pi],\ t^i \in [0, 1],\ l \text{ is the task-ID}\}$, the solution of the equation at the point (x^i, t^i, l) as $\mathbf{u}^i_{l,k} = u^i_{l,k}(x^i, t^i)$, and $\hat{\mathbf{u}}^i_{l,k}$ is a prediction of the model at point (x^i, t^i, l) after task $\mathcal{D}_k$ is learned. The relative error is denoted as $r_{l,k} = \frac{||\mathbf{u}_l - \hat{\mathbf{u}}_{l,k}||_2}{||\mathbf{u}_l||_2} \times 100\%$ as it is calculated for task l after task k is learned ($l \leq k$). Backward Transfer is defined as $\text{BWT} = \frac{1}{k-1} \sum_{l=1}^{k-1} r_{l,k} - r_{l,l}$.

Table 1: Final relative error and forgetting after all convection equations are learned.

	regular PINN	iPINN
$\beta = 1$	$\mathbf{0.042}\%$	0.074%
$\beta = 10$	0.222%	$\mathbf{0.087}\%$
$\beta = 20$	0.339%	$\mathbf{0.288}\%$
$\beta = 30$	3.957%	$\mathbf{0.246}\%$
$\beta = 40$	37.4%	$\mathbf{1.139}\%$
BWT	N/A	0.0280%

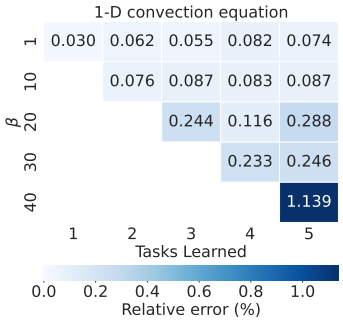


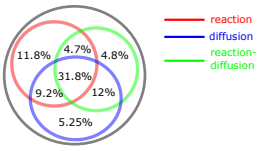
Figure 2: Relative error history for convection equations ($\alpha = 0.95$). Every row shows the error after a new task is learned.

Results. For the convection equation, we observe that by learning incrementally the sequence of convection equations, we achieve much lower absolute and relative errors for the equations that are more difficult to learn ($\beta = 30, 40$). In Table 1 we show final errors at the end of the training, and Figure 2 shows the absolute error history for each equation. In this case, we observe some level of forgetting, however, it is insignificant compared to the error values.

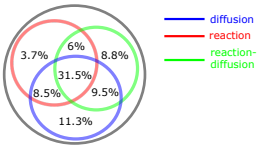
We also illustrate the effectiveness of the iPINN method by addressing problem P2. Results obtained when first learning the reaction part (or vice-versa, the diffusion part) are shown in Table 2. The main finding is that the network can learn every equation at least as well as when it is learned independently. In fact, for the reaction equation, the neural network improves significantly the prediction error. Another interesting observation is that the model learns the reaction-diffusion equation with almost the same error, regardless of the order of the tasks. This gives us a hint about the robustness of the algorithm to different task orders in terms of prediction error. In Figure 3, we present the portions of the subnetworks that are occupied by each task. We will illustrate this by considering both orders – when the model learns the reaction equation first (Figure 2a), and when diffusion comes first (Figure 2b). It is noteworthy that the percentage of parameters occupied by all tasks is very similar for both orderings (31.8% and 31.5% respectively of all network parameters). On the other hand, the percentages of used parameters for both cases are 79.5% and 79.3%. This means that the total number of trained parameters for the two incremental procedures is the same for both cases, which shows the

Table 2: Final relative error for $\nu = 4,\ \rho = 4$ and two different orders.

equation	regular PINN	iPINN
reaction	3.68%	**2.99%**
diffusion	**0.16%**	0.19%
reaction-diffusion	0.70%	**0.67%**
diffusion	0.16%	**0.12%**
reaction	3.68%	**1.97%**
reaction-diffusion	0.70%	**0.67%**



(a) reaction → diffusion → reaction-diffusion.



(b) diffusion → reaction → reaction-diffusion.

Figure 3: Percentage of parameters used for every equation with $\rho = 4,\ \nu = 4$.

robustness of the method. Moreover, the network has about 20% of free connections to learn new tasks.

5 Conclusion

In this work, we propose an incremental learning approach for PINNs where every task is presented as a new PDE. Our algorithm is based on task-related subnetworks for every task obtained by iterative pruning. To illustrate our idea, we consider two cases when incremental learning is applicable to a sequence of PDEs. In the first case, we consider the family of convection PDEs, learning them sequentially. In the second example, we consider the reaction-diffusion equation and learn firstly the components of the process, namely reaction and diffusion, and only then the reaction-diffusion equation. Our main goal is to show the possibility of incremental learning for PINNs without significantly forgetting previous tasks. From our numerical experiments, the proposed algorithm can learn all the given tasks, which is not possible with standard PINNs. Importantly, we also show that future tasks are learned better because they can share connections trained from previous tasks, leading to significantly better performance than if these tasks were learned independently. We demonstrate that this stems from the transfer of knowledge occurring between subnetworks that are associated with each task.

References

[1] Celia Fernández Madrazo, Ignacio Heredia, Lara Lloret, and Jesús Marco de Lucas. Application of a convolutional neural network for image classification for the analysis of collisions in high energy physics. In *EPJ Web of Conferences*, volume 214, page 06017. EDP Sciences, 2019.

[2] Alexander Bogatskiy, Brandon Anderson, Jan Offermann, Marwah Roussi, David Miller, and Risi Kondor. Lorentz group equivariant neural network for particle physics. In *International Conference on Machine Learning*, pages 992–1002. PMLR, 2020.

[3] Jonathan Shlomi, Peter Battaglia, and Jean-Roch Vlimant. Graph neural networks in particle physics. *Machine Learning: Science and Technology*, 2(2):021001, 2020.

[4] Omar Khatib, Simiao Ren, Jordan Malof, and Willie J Padilla. Learning the physics of all-dielectric metamaterials with deep lorentz neural networks. *Advanced Optical Materials*, page 2200097, 2022.

[5] Gonçalo Marques, Deevyankar Agarwal, and Isabel de la Torre Díez. Automated medical diagnosis of covid-19 through efficientnet convolutional neural network. *Applied soft computing*, 96:106691, 2020.

[6] Tapas Si, Jayri Bagchi, and Péricles BC Miranda. Artificial neural network training using metaheuristics for medical data classification: an experimental study. *Expert Systems with Applications*, 193:116423, 2022.

[7] DR Sarvamangala and Raghavendra V Kulkarni. Convolutional neural networks in medical image understanding: a survey. *Evolutionary intelligence*, 15(1):1–22, 2022.

[8] Tadaaki Hosaka. Bankruptcy prediction using imaged financial ratios and convolutional neural networks. *Expert systems with applications*, 117:287–299, 2019.

[9] Pengfei Yu and Xuesong Yan. Stock price prediction based on deep neural networks. *Neural Computing and Applications*, 32(6):1609–1628, 2020.

[10] Periklis Gogas and Theophilos Papadimitriou. Machine learning in economics and finance. *Computational Economics*, 57(1):1–4, 2021.

[11] Jun Wang and Xin Gan. Neurodynamics-driven portfolio optimization with targeted performance criteria. *Neural Networks*, 157:404–421, 2023.

[12] Miguel A Bessa, R Bostanabad, Zeliang Liu, A Hu, Daniel W Apley, C Brinson, Wei Chen, and Wing Kam Liu. A framework for data-driven analysis of materials under uncertainty: Countering the curse of dimensionality. *Computer Methods in Applied Mechanics and Engineering*, 320:633–667, 2017.

[13] Ivan Sosnovik and Ivan Oseledets. Neural networks for topology optimization. *Russian Journal of Numerical Analysis and Mathematical Modelling*, 34(4):215–223, 2019.

[14] Aaditya Chandrasekhar and Krishnan Suresh. Tounn: topology optimization using neural networks. *Structural and Multidisciplinary Optimization*, 63(3):1135–1149, 2021.

[15] Nerea Portillo Juan and Vicente Negro Valdecantos. Review of the application of artificial neural networks in ocean engineering. *Ocean Engineering*, 259:111947, 2022.

[16] Hyuk Lee and In Seok Kang. Neural algorithm for solving differential equations. *Journal of Computational Physics*, 91(1):110–131, 1990.

[17] Andrew J Meade Jr and Alvaro A Fernandez. Solution of nonlinear ordinary differential equations by feedforward neural networks. *Mathematical and Computer Modelling*, 20(9):19–44, 1994.

[18] R Yentis and ME Zaghloul. Vlsi implementation of locally connected neural network for solving partial differential equations. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 43(8):687–690, 1996.

[19] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

[20] Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 2021.

[21] Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems*, 34:26548–26560, 2021.

[22] Franz M Rohrhofer, Stefan Posch, Clemens Gößnitzer, and Bernhard C Geiger. Understanding the difficulty of training physics-informed neural networks on dynamical systems. *arXiv preprint arXiv:2203.13648*, 2022.

[23] Rambod Mojgani, Maciej Balajewicz, and Pedram Hassanzadeh. Lagrangian pinns: A causality-conforming solution to failure modes of physics-informed neural networks. *arXiv preprint arXiv:2205.02902*, 2022.

[24] Colby L Wight and Jia Zhao. Solving allen-cahn and cahn-hilliard equations using the adaptive physics informed neural networks. *arXiv preprint arXiv:2007.04542*, 2020.

[25] Souvik Chakraborty. Transfer learning based multi-fidelity physics informed deep neural network. *Journal of Computational Physics*, 426:109942, 2021.

[26] Ameya D Jagtap and George E Karniadakis. Extended physics-informed neural networks (xpinns): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. In *AAAI Spring Symposium: MLPS*, 2021.

[27] Peter R Wiecha, Arnaud Arbouet, Christian Girard, and Otto L Muskens. Deep learning in nano-photonics: inverse design and beyond. *Photonics Research*, 9(5):B182–B200, 2021.

[28] Aleksandr Dekhovich, David MJ Tax, Marcel HF Sluiter, and Miguel A Bessa. Neural network relief: a pruning algorithm based on neural activity. *arXiv preprint arXiv:2109.10795*, 2021.

[29] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[30] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.