Can Physics Informed Neural Operators Self Improve?

Ritam Majumdar
TCS Research
ritam.majumdar@tcs.com

Amey Varhade
IIT Guwahati
varhade@alumni.iitg.ac.in

Shirish Karande
TCS Research
shirish.karande@tcs.com

Lovekesh Vig
TCS Research
lovekesh.vig@tcs.com

Abstract

Self-training techniques have shown remarkable value across many deep learning models and tasks. However, such techniques remain largely unexplored when considered in the context of learning fast solvers for systems of partial differential equations (Eg: Neural Operators). In this work, we explore the use of self-training for Fourier Neural Operators (FNO). Neural Operators emerged as a data driven technique, however, data from experiments or traditional solvers is not always readily available. Physics Informed Neural Operators (PINO) overcome this constraint by utilizing a physics loss for the training, however the accuracy of PINO trained without data does not match the performance obtained by training with data. In this work we show that self-training can be used to close this gap in performance. We examine canonical examples, namely the 1D-Burgers and 2D-Darcy PDEs, to showcase the efficacy of self-training. Specifically, FNOs, when trained exclusively with physics loss through self-training, approach $1.07\times$ for Burgers and $1.02\times$ for Darcy, compared to FNOs trained with both data and physics loss. Furthermore, we discover that pseudo-labels can be used for self-training without necessarily training to convergence in each iteration. A consequence of this is that we are able to discover self-training schedules that improve upon the baseline performance of PINO in terms of accuracy as well as time.

1 Introduction

In recent years, the development of rapid solvers for Partial Differential Equations (PDEs) has garnered substantial interest. Various Neural Operators [1–4] have notably been harnessed to expedite PDE solvers, among which Fourier Neural Operators (FNO) [1] have gained significant popularity. These have found applications in a myriad of fields including fluid flow, climate change [5], and material modeling [6]. However, Neural Operators, predominantly trained via data-driven methodologies, often confront challenges related to data availability. Physics-Informed Neural Operators (PINO) [7], address this by utilizing an physics-informed loss, based on governing PDEs. Nevertheless, a noticeable disparity in performance is evident when comparing PINOs trained with supervisory loss based on ground truth labels to those trained without.

Self-training, a semi-supervised learning approach, initiates by utilizing a model to generate labels for unlabeled instances. Subsequent to predictions, these instances are amalgamated with the labeled dataset, facilitating further model training. Through repeated iterations, this cycle aims to improve the model's predictive capabilities. Self-training has found utility in varied applications including sentiment analysis [8], text classification [9], Natural Language Processing [10], object detection [11], medical image classification [12], human action recognition [13], facial expression identification [14], Speech Recognition [15, 16], Anomaly Detection [17], as well as Genomics and proteomics

[18]. Intriguingly, self-training approaches have scarcely been explored to expedite PDE solvers. The closest work which resembles using self-training for PDE solvers is [19], wherein the authors use self-training for Physics-informed Neural Networks. More specifically, the authors add the most confident collocation points as pseudo-labels to improve the performance of surrogate PINN for single PDE instance. In this work however, we are performing self-training to generalize over entire function-spaces of PDE tasks rather than a single instance of a PDE, and use PINOs instead of PINNs.

In this work, we make two important contributions: (1) We show that self-training can be used along with PINO to close the gap in performance between PINO trained with just physics loss v/s the performance obtained by including physics + data loss. We illustrate the efficacy of self-training on the 1D-Burgers and 2D-Darcy PDEs. The performance ratio in terms of the $L2$ error of PINOs trained in term of just physics loss against PINOs trained with physics + data loss is 39.48 for 1D-Burgers and 1.16 for 2D Darcy. We show that self-training improves this ratio to 1.07 and 1.02. (2) We also explore the utility of self-training and pseudo-labels in reducing the training time of PINOs. In particular we observed that one can follow an early stopping schedule while training a PINO with physics loss and subsequently with physics + pseudo-labels loss . We observe that some schedules are able to obtain an accuracy greater than PINO while utilizing time lesser than that required to train a PINO to convergence on the basis of physics loss.

The remainder of the paper is structured as follows: Section 2 outlines the Methodology used for self-training, the results and observations are discussed in Section 3 and we highlight the key conclusions in Section 4.

2 Methodology

Algorithm 1 Methodology for Self-training of PINO

Input: Parametric PDE N, Domain Discretization, Train tasks T_{train} (A)
Output: Trained Fourier Neural Operator $G_{\theta_{\text{final}}}$
Method:
 1: Initialize Fourier Neural Operator G_{θ_1}
 2: Train G_{θ_1} using Physics-loss over T_{train} until $StoppingCriteria$
 3: **for** i (self-train iterations) = 1 to MAX **do**
 4:     Use G_{θ_i} to generate data labels for T_{train} as D_{labels}
 5:     Initialize a new Neural Operator $G_{\theta_{i+1}}$ with G_{θ_i}
 6:     Train $G_{\theta_{i+1}}$ using Physics-loss and Supervised-loss over D_{labels} until $StoppingCriteria$
 7:     **if** $|Error(G_{\theta_{i+1}}) - Error(G_{\theta_i})| < \epsilon$ **then**
 8:        **break**
 9:     **end if**
10: **end for**
11: **return** $G_{\theta_{\text{final}}}$

Fourier Neural Operators G_θ aim to learn a mapping between two Banach spaces $G_\theta : A \rightarrow U$, where A typically refers to parameterized initial conditions, boundary conditions, or coefficients of the PDE, while U refers to the final solution of the PDE. Algorithm 2 describes the methodology for self-training a physics-informed Neural Operator (PINO). We initialize a Fourier Neural Operator, denoted as G_{θ_1} and train it on input functions A exclusively using physics-loss. Here, parameterized input functions A refer to our train-tasks T_{train}. The physics-loss is defined using the governing equations of the Partial Differential Equation N. Leveraging the trained FNO from each prior iteration, we deduce pseudo-labels for our training tasks. These pseudo-labels subsequently function as annotated data points, facilitating supervised fine-tuning in the following iterations. In every iteration, we initialize $G_{\theta_{i+1}}$ using its predecessor G_{θ_i}, where i is the current self-train iteration. This FNO is then trained on T_{train}, drawing upon both the physics-loss and a supervised-loss on the aforegenerated pseudo-labels. This iterative loop of pseudo-label inference and operator enhancement is sustained until the absolute difference of two consecutive iterations of FNOs $G_{\theta_{i+1}}$ and G_{θ_i} falls below a threshold ϵ or maximum number of self-train iterations is completed. Stopping Criteria for training a FNO is met when there is no improvement in the train-loss for 100 consecutive epochs or pre-defined number of epochs are completed.

		Self-train Iterations							Benchmarks	
		0 (B0)	1	2	3	4	5	6	B1	B2
1D-Burger's	Data $(1e^{-4})$	74.22	3.28	2.75	2.19	2.11	2.03	2.02	1.88	16.26
	Phy. $(1e^{-6})$	8.22	6.96	5.42	4.84	4.00	3.66	3.83	2.65	547
2D-Darcy	Data $(1e^{-2})$	1.15	1.09	1.07	1.03	1.02	1.01	1.01	0.99	2.36
	Phy. $(1e^{-2})$	3.22	1.72	1.45	1.22	1.15	0.98	0.97	0.85	107

Table 1: Self-training experiments. Benchmark B1 refers to training PINOs using ground-truth data, while Benchmark B2 refers to training FNOs solely on ground-truth data without physics-loss. Benchmark B0 refers to the FNO trained with just physics-informed loss. Quantities in parenthesis denote the order of the errors. Data refers to test-task L2 errors, while Phy. refers to test-task PDE Mean-squared errors.

3 Results and Observations

3.1 Self-training with PINOs trained to convergence

Let us consider three baselines. B0 (or PINO), FNOs trained with just physics-informed loss, B1, FNOs trained with both physics-informed and actual ground-truth data loss, and B2, FNOs trained solely on ground-truth labels. Table 1 has the performance results for all these baselines and significant performance gap can be observed between training a PINO with ground-truth data and PINO without ground-truth data. For example, in case of 1D Burger's B0 has an L2 error of $7.42e^{-3}$, while B1 has an error of $1.88e^{-4}$. Infact, the performance of PINO is even worse than the conventional FNO in case of 1D-Burger's i.e. B2, $1.63e^{-3}$. Meanwhile, in case of 2D-Darcy, even though the accuracy of PINO (B0) is greater than that of FNO ($1.15e^{-2}$ v/s $2.36e^{-2}$), it pales significantly in comparison of PINO with data loss ($9.91e^{-3}$).

Table 1 demonstrates the result of self-training a PINO. In this experiment the convergence is not defined in terms of epochs, rather in terms of saturation of reduction in error. We notice that even single iteration of self training can provide significant gains. A self-trained PINO is not only able to easily beat the performance of an FNO but can almost match the performance of PINO trained with data. Our results show that the self-training approach, despite lacking access to true ground-truth, significantly outperforms B2 in both data (Burger's: 8.61, Darcy: 2.33) and physics-loss (Burger's: 206, Darcy: 125). Compared to B1, it only exhibits slightly higher errors, specifically [(1.07×,1.02×) for data L2 error and (1.45×,1.15×) for physics-loss MSE for Burger's and Darcy respectively.

3.2 Self-training with PINOs not-trained to convergence

Why and when does self-training improve performance? Will models that are not trained to convergence improve more or less with self-training? Poor training will induce noisy pseudo-labels possibly leading to error amplification, however, if the training is just good enough then we might save some epochs of training, without any recognizable loss in inducing pseudo-labels, thus possibly reducing the overall training time. We sought to investigate this possibility. Therefore we trained the models for a fixed number of epochs in each iteration. Indeed a strong self-improvement trend was observed in most cases, until we reach threshold where the models could start diverging. One can observe Figure 1 closely. In the Burger's equation, PINO without labels has a data L2-error of $7.42e^{-3}$, while taking 18:37 (18 mins and 37 sec) to converge. We observe, self-training for 250 epochs gives us a performance improvement ($2.5e^{-3}$) in the 2nd iteration itself, with a time advantage (14:12), while self-training for 100 epochs on 3rd iteration itself gives us a performance ($5.2e^{-3}$) and time improvement (8:36). Similarly, for Darcy flow, PINO without labels has a data L2-error of $1.15e^{-2}$, while taking (3:10:00) 3 hrs 10 mins to converge. Self-training for 500 epochs for 5 iterations gives us a performance $1.11e^{-2}$ and time improvement (1:58:22), while self-training for 250 epochs at 6th iteration gives us a performance $1.14e^{-2}$ and time improvement (1:05:13). In Figure 1, the highlighted point (in red) represents the benchmark B0. All points belonging to the third quadrant with benchmark B0 indicate Pareto-points wherein self-training iterations are both better in performance and time, compared to the benchmark B0. We further provide detailed numerical tabulations of these experiments in the Appendix 2,3. Thus, we infer that using number of epochs as a hyper-parameter can outperform traditional PINO in terms of error while reducing training time.
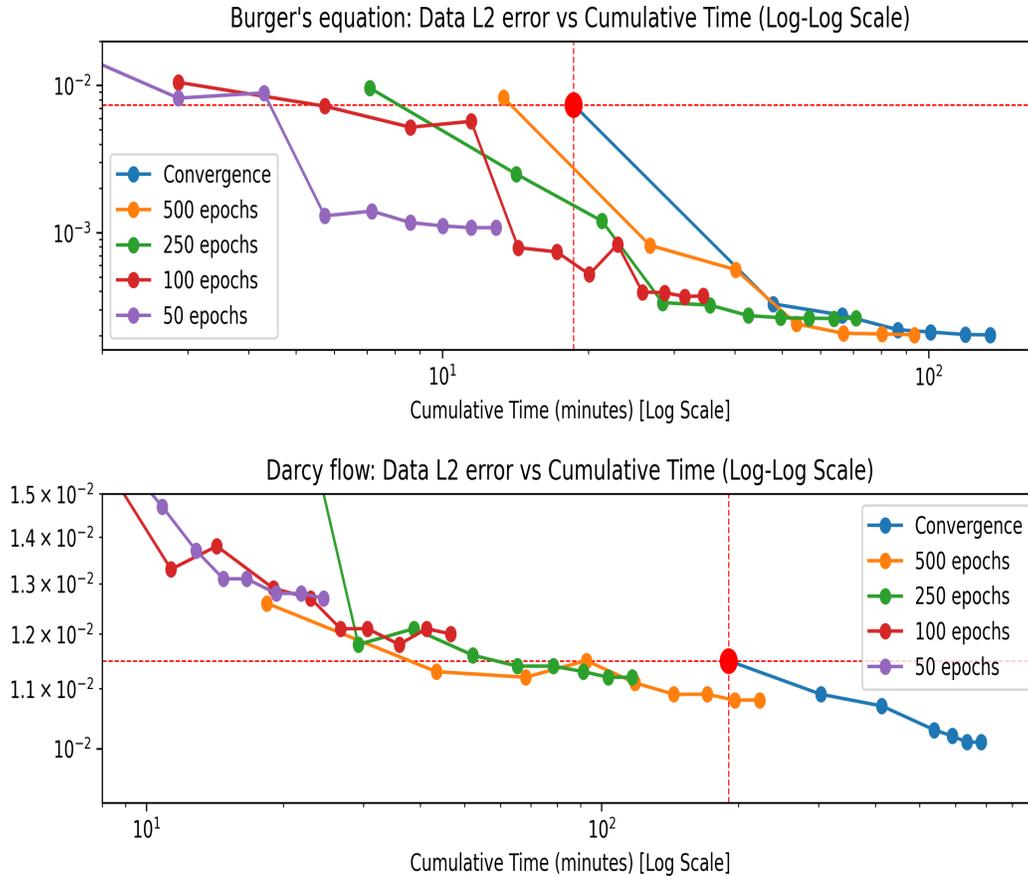
Figure 1: Self-training experiments with PINOs trained prematurely. Top: Burger's equation, Bottom: Darcy flow. We study the trend of Data L2-error against cumulative time on a log-log scale. The highlighted point (in red) represents the benchmark B0, i.e. Training a PINO without data-labels.

4   Conclusion

Self-training has been largely unexplored while employing machine learning in numerical methods. In this paper we have successfully shown that self-training with pseudo-label and physics loss can substantially improve the performance of FNOs without the need of any training data. Furthermore we have demonstrated that one can observe the phenomenon of self-improvement through self-training even with noisier FNOs. We have observed that one can stop training early in each iteration of self-training to obtain improvement in accuracy without exceeding the time-budget consumed for training a PINO. We believe that this issue needs further investigation in future.

5   Broader Impact

Creation of training data for solvers can consume a lot of compute. We believe this has been one of the reasons that a lot of data for solved systems does not exist in public domain. This in turn has meant that the current trend of foundational models which can be used for a broad number of tasks, has not extended to emergence of instruction tuned foundational models for fast solvers. We believe that if we can train models through self-improvement, even when they are initially noisy, we can potentially scale the breadth of tasks on which a single model gets trained. This has the potential to pave a path towards building a foundational model for a broad set of PDE tasks.

References

[1] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations, 2021.

[2] Tapas Tripura and Souvik Chakraborty. Wavelet neural operator: a neural operator for parametric partial differential equations. *arXiv preprint arXiv:2205.02191*, 2022.

[3] Ritam Majumdar, Shirish Karande, and Lovekesh Vig. How important are specialized transforms in neural operators? *arXiv preprint arXiv:2308.09293*, 2023.

[4] Gaurav Gupta, Xiongye Xiao, and Paul Bogdan. Multiwavelet-based operator learning for differential equations. *Advances in neural information processing systems*, 34:24048–24062, 2021.

[5] Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, et al. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022.

[6] Huaiqian You, Quinn Zhang, Colton J Ross, Chung-Hao Lee, and Yue Yu. Learning deep implicit fourier neural operators (ifnos) with applications to heterogeneous material modeling. *Computer Methods in Applied Mechanics and Engineering*, 398:115296, 2022.

[7] Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *arXiv preprint arXiv:2111.03794*, 2021.

[8] Akshat Gupta, Sargam Menghani, Sai Krishna Rallabandi, and Alan W Black. Unsupervised self-training for sentiment analysis of code-switched data, 2021.

[9] Yu Meng, Yunyi Zhang, Jiaxin Huang, Chenyan Xiong, Heng Ji, Chao Zhang, and Jiawei Han. Text classification using label names only: A language model self-training approach. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9006–9017, Online, November 2020. Association for Computational Linguistics.

[10] Seniz Demir, Jan Raab, Nils Reiter, Marketa Lopatkova, and Tomek Strzalkowski, editors. *Proceedings of the ACL 2010 Student Research Workshop*, Uppsala, Sweden, July 2010. Association for Computational Linguistics.

[11] Gang Li, Xiang Li, Yujie Wang, Yichao Wu, Ding Liang, and Shanshan Zhang. Dtg-ssod: Dense teacher guidance for semi-supervised object detection, 2022.

[12] Zhen Peng, Dezhi Zhang, Shengwei Tian, Weidong Wu, Long Yu, Shaofeng Zhou, and Shanhang Huang. Faxmatch: Multi-curriculum pseudo-labeling for semi-supervised medical image classification. *Medical Physics*, 50(5):3210–3222, 2023.

[13] Chenxi Wang, Jingzhou Luo, Xing Luo, Haoran Qi, and Zhi Jin. V-dixmatch: A semi-supervised learning method for human action recognition in night video sensing. *IEEE Sensors Journal*, pages 1–1, 2023.

[14] Nazir Shabbir and Ranjeet Kumar Rout. Fgbcnn: A unified bilinear architecture for learning a fine-grained feature representation in facial expression recognition. *Image and Vision Computing*, 137:104770, 2023.

[15] Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations, 2020.

[16] Mozhdeh Gheini, Tatiana Likhomanenko, Matthias Sperber, and Hendra Setiawan. Joint speech transcription and translation: Pseudo-labeling with out-of-distribution data. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 7637–7650, Toronto, Canada, July 2023. Association for Computational Linguistics.

[17] Jia-Chang Feng, Fa-Ting Hong, and Wei-Shi Zheng. Mist: Multiple instance self-training framework for video anomaly detection, 2021.

[18] Zhuohan Yu, Yanchi Su, Yifu Lu, Yuning Yang, Fuzhou Wang, Shixiong Zhang, Yi Chang, Ka-Chun Wong, and Xiangtao Li. Topological identification and interpretation for single-cell gene regulation elucidation across multiple platforms using scmgca. *Nature communications.*, 14(1), 2023-12-25.

[19] Junjun Yan, Xinhai Chen, Zhichao Wang, Enqiang Zhoui, and Jie Liu. St-pinn: A self-training physics-informed neural network for partial differential equations, 2023.

A Appendix

A.1 PDE information

A.1.1 Burgers' Equation

We consider the 1D Burgers' equation, a non-linear PDE with periodic boundary conditions. The governing equations are defined as follows:

$$\partial_t u(x,t) + \partial_x \left(u^2(x,t)/2\right) = \nu \partial_{xx} u(x,t), \quad x \in (0,1), t \in (0,1]$$
$$u(x,0) = u_0(x), \qquad x \in (0,1)$$

Here u_0 is the initial condition and $\nu = 1e^{-2}$ is the viscosity coefficient. We aim to learn the operator mapping the initial condition u_0 to the entire solution at $u|_{[0,1]}$.

A.1.2 2D Darcy Flow

2D Darcy Flow is a linear steady-state second-order elliptic PDE. We consider a flow in a unit box, whose governing equations are given by:

$$-\nabla \cdot (a(x)\nabla u(x)) = f(x), \quad x \in (0,1)^2$$
$$u(x) = 0, \qquad x \in \partial (0,1)^2$$

Here a is the piecewise constant diffusion coefficient and $f = 1$ is the forcing function. The objective is to learn the operator mapping the diffusion coefficient $a(x)$ to the solution $u(x)$.

A.2 Training and Hyperparameter Details

As our base architecture, we consider a FNO stacked with 4 layers with 15,12,9,9 fourier modes in every layer, using GeLU as our activation function to model non-linearities. We consider 800 train and 200 test examples in Burger's equation, while we consider 1000 train and 500 test examples in Darcy equation respectively. The prediction gridsize of $[x,t] \to [128, 101]$ for Burger's equation, while for Darcy equation it's $[x,y] \to [61, 61]$. All experiments were conducted on Nvidia P100 GPU with 16 GB GPU Memory and 1.32 GHz GPU Memory clock using Pytorch framework. In our benchmark experiments B1, B2 and for self-training experiments trained till convergence, we start with a learning rate of $1e^{-3}$, with a patience of 100 and tolerance of $1e^{-5}$ for the total train error, until the minimum learning rate of $1e^{-6}$ is obtained. In the self-training experiments wherein the Neural Operators are trained for 250, 100 and 50 epochs (not till convergence): We start with a base learning rate of $1e^{-3}$, and reduce the learning rate by a factor of 0.1 when there is no-improvement in the test error at start of every iteration. The physics-informed loss is evaluated using Finite Difference Method in the spectral domain. In benchmark experiments B0 and B1, the dataset is generated using Finite-Difference Method in the spectral domain with input conditions generated using Gaussian Random Fields.

A.3 Tabulated Results of Self-training experiments

Table 2: 1D Burger's equation results

Iteration		Conv.	500 epochs	250 epochs	100 epochs	50 epochs
	Data L2-error	$7.42e^{-3}$	$8.26e^{-3}$	$9.65e^{-3}$	$1.05e^{-2}$	$2.23e^{-2}$
0	Physics MSE	$8.22e^{-6}$	$9.81e^{-6}$	$5.33e^{-5}$	$7.48e^{-5}$	$6.58e^{-4}$
	Cum. time	18:37	13:22	7:06	2:52	1:26
	Data L2-error	$3.28e^{-4}$	$8.25e^{-4}$	$2.56e^{-3}$	$7.24e^{-3}$	$8.26e^{-3}$
1	Physics MSE	$6.96e^{-6}$	$1.17e^{-5}$	$1.06e^{-5}$	$3.06e^{-5}$	$8.01e^{-4}$
	Cum. time	47:53	26:44	14:12	5:44	2:52
	Data L2-error	$2.75e^{-4}$	$5.69e^{-4}$	$1.28e^{-3}$	$5.21e^{-3}$	$8.91e^{-3}$
2	Physics MSE	$5.42e^{-6}$	$7.62e^{-6}$	$6.97e^{-6}$	$3.57e^{-5}$	$7.29e^{-4}$
	Cum. time	1:06:30	40:06	21:18	8:36	4:18
	Data L2-error	$2.19e^{-4}$	$2.45e^{-4}$	$3.34e^{-4}$	$5.73e^{-4}$	$1.32e^{-3}$
3	Physics MSE	$4.84e^{-6}$	$4.66e^{-6}$	$5.01e^{-6}$	$3.85e^{-5}$	$4.10e^{-4}$
	Cum. time	1:26:24	53:28	28:24	11:28	5:44
	Data L2-error	$2.11e^{-4}$	$2.07e^{-4}$	$3.22e^{-4}$	$7.92e^{-4}$	$1.42e^{-3}$
4	Physics MSE	$4.00e^{-6}$	$3.90e^{-6}$	$4.43e^{-6}$	$1.05e^{-5}$	$4.10e^{-4}$
	Cum. time	1:41:06	1:06:50	35:30	14:20	7:10
	Data L2-error	$2.03e^{-4}$	$2.05e^{-4}$	$2.73e^{-4}$	$7.42e^{-4}$	$1.17e^{-3}$
5	Physics MSE	$3.66e^{-6}$	$3.85e^{-6}$	$4.16e^{-6}$	$8.95e^{-6}$	$4.05e^{-4}$
	Cum. time	1:59:00	1:20:12	42:35	17:12	8:36
	Data L2-error	$2.02e^{-4}$	$2.02e^{-4}$	$2.64e^{-4}$	$5.25e^{-4}$	$1.15e^{-3}$
6	Physics MSE	$3.83e^{-6}$	$3.83e^{-6}$	$3.91e^{-6}$	$8.15e^{-6}$	$3.97e^{-4}$
	Cum. time	2:13:49	1:33:34	49:40	20:04	10:02
	Data L2-error			$2.62e^{-4}$	$8.36e^{-4}$	$1.08e^{-3}$
7	Physics MSE			$3.84e^{-6}$	$7.55e^{-6}$	$3.97e^{-4}$
	Cum. time			56:45	22:56	11:28
	Data L2-error			$2.61e^{-4}$	$3.95e^{-4}$	$1.08e^{-3}$
8	Physics MSE			$3.82e^{-6}$	$7.39e^{-6}$	$3.96e^{-4}$
	Cum. time			1:03:50	25:48	12:54
	Data L2-error			$2.62e^{-4}$	$3.91e^{-4}$	
9	Physics MSE			$3.81e^{-6}$	$7.39e^{-6}$	
	Cum. time			1:10:55	28:40	
	Data L2-error				$3.68e^{-4}$	
10	Physics MSE				$7.39e^{-6}$	
	Cum. time				31:32	
	Data L2-error				$3.72e^{-4}$	
11	Physics MSE				$7.38e^{-6}$	
	Cum. time				34:24	

Table 3: 2D Darcy flow results

Iteration		Conv.	500 epochs	250 epochs	100 epochs	50 epochs
	Data L2-error	$1.15e^{-2}$	$1.26e^{-2}$	$1.63e^{-2}$	$3.22e^{-2}$	$6.96e^{-2}$
0	Physics MSE	$3.22e^{-2}$	$5.46e^{-2}$	$8.71e^{-2}$	$1.21e^{-1}$	$1.72e^{-1}$
	Cum. time	3:10:00	18:24	9:41	4:02	2:13
	Data L2-error	$1.09e^{-2}$	$1.13e^{-2}$	$2.19e^{-2}$	$1.61e^{-2}$	$2.92e^{-2}$
1	Physics MSE	$1.72e^{-2}$	$2.20e^{-2}$	$7.78e^{-2}$	$1.02e^{-1}$	$1.35e^{-1}$
	Cum. time	5:03:37	45:22	18:38	7:46	4:08
	Data L2-error	$1.07e^{-2}$	$1.12e^{-2}$	$1.18e^{-2}$	$1.33e^{-2}$	$3.18e^{-2}$
2	Physics MSE	$1.45e^{-2}$	$1.84e^{-2}$	$3.41e^{-2}$	$7.98e^{-2}$	$1.21e^{-1}$
	Cum. time	6:52:59	1:10:06	29:18	11:21	6:01
	Data L2-error	$1.03e^{-2}$	$1.15e^{-2}$	$1.21e^{-2}$	$1.38e^{-2}$	$1.64e^{-2}$
3	Physics MSE	$1.22e^{-2}$	$1.60e^{-2}$	$2.91e^{-2}$	$7.76e^{-2}$	$5.51e^{-2}$
	Cum. time	8:57:20	1:34:37	38:46	14:18	8:39
	Data L2-error	$1.02e^{-2}$	$1.11e^{-2}$	$1.16e^{-2}$	$1.29e^{-2}$	$1.47e^{-2}$
4	Physics MSE	$1.15e^{-2}$	$1.44e^{-2}$	$1.88e^{-2}$	$3.36e^{-2}$	$4.72e^{-2}$
	Cum. time	9:48:42	2:00:22	52:13	18:42	10:49
	Data L2-error	$1.01e^{-2}$	$1.09e^{-2}$	$1.14e^{-2}$	$1.27e^{-2}$	$1.37e^{-2}$
5	Physics MSE	$9.81e^{-3}$	$1.32e^{-2}$	$1.72e^{-2}$	$3.26e^{-2}$	$4.63e^{-2}$
	Cum. time	10:33:56	2:25:56	1:05:23	22:38	12:51
	Data L2-error	$1.01e^{-2}$	$1.09e^{-2}$	$1.14e^{-2}$	$1.21e^{-2}$	$1.31e^{-2}$
6	Physics MSE	$9.79e^{-3}$	$1.30e^{-2}$	$1.63e^{-2}$	$3.15e^{-2}$	$4.85e^{-2}$
	Cum. time	11:22:17	2:52:25	1:18:21	26:23	14:45
	Data L2-error		$1.08e^{-2}$	$1.13e^{-2}$	$1.21e^{-2}$	$1.31e^{-2}$
7	Physics MSE		$1.28e^{-2}$	$1.56e^{-2}$	$2.93e^{-2}$	$4.46e^{-2}$
	Cum. time		3:18:54	1:31:18	30:28	16:35
	Data L2-error		$1.08e^{-2}$	$1.12e^{-2}$	$1.18e^{-2}$	$1.28e^{-2}$
8	Physics MSE		$1.27e^{-2}$	$1.51e^{-2}$	$2.14e^{-2}$	$3.27e^{-2}$
	Cum. time		3:45:16	1:43:31	35:47	19:17
	Data L2-error			$1.12e^{-2}$	$1.21e^{-2}$	$1.28e^{-2}$
9	Physics MSE			$1.36e^{-2}$	$1.99e^{-2}$	$3.12e^{-2}$
	Cum. time			1:56:34	41:06	21:51
	Data L2-error				$1.23e^{-2}$	$1.27e^{-2}$
10	Physics MSE				$1.89e^{-2}$	$3.09e^{-2}$
	Cum. time				46:25	24:30