Parameterized Physics-informed Neural Networks for Parameterized PDEs

Woojin Cho [1 2] **Minju Jo** [3] **Haksoo Lim** [1] **Kookjin Lee** [2] **Dongeun Lee** [4] **Sanghyun Hong** [5] **Noseong Park** [6]

Abstract

Complex physical systems are often described by partial differential equations (PDEs) that depend on parameters such as the Raynolds number in fluid mechanics. In applications such as design optimization or uncertainty quantification, solutions of those PDEs need to be evaluated at numerous points in the parameter space. While physics-informed neural networks (PINNs) have emerged as a new strong competitor as a surrogate, their usage in this scenario remains underexplored due to the inherent need for repetitive and time-consuming training. In this paper, we address this problem by proposing a novel extension, parameterized physics-informed neural networks (P^2INNs). P^2INNs enable modeling the solutions of parameterized PDEs via explicitly encoding a latent representation of PDE parameters. With the extensive empirical evaluation, we demonstrate that P^2INNs outperform the baselines both in accuracy and parameter efficiency on benchmark 1D and 2D parameterized PDEs and are also effective in overcoming the known "failure modes".

Table 1. **P^2INNs greatly improve the quality of CDR solutions.** We compare the average absolute (Abs.) and relative (Rel.) errors of PINN and P^2INN in six different CDR equations. IMP is the improvement ratio of P^2INN to PINN. (see Section 4.1 for details).

PDE type	PINN		P^2INN		IMP. (%)	
	Abs.	Rel.	Abs.	Rel.	Abs.	Rel.
Convection	0.0496	0.0871	0.0330	0.0241	33.43	72.37
Diffusion	0.3611	0.6939	0.1592	0.3190	55.91	54.03
Reaction	0.5825	0.6431	0.0041	0.0069	99.30	98.92
Conv.-Diff.	0.1493	0.2793	0.0532	0.1236	64.34	55.75
Reac.-Diff.	0.4744	0.5614	0.1319	0.2008	72.21	64.24
Conv.-Diff.-Reac.	0.4811	0.5315	0.0391	0.0759	91.88	85.72

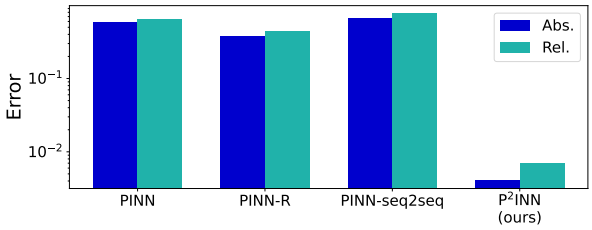


Figure 1. **P^2INNs outperform the baselines.** P^2INNs reduce the average L_2 absolute (Abs.) and relative (Rel.) errors by $100\times$ compared to the baselines. The results are for reaction equations – the most challenging problems for PINNs.

1. Introduction

Scientific machine learning (SML) (Baker et al., 2019) has been growing fast. Unlike traditional tasks in machine learning, e.g., image classification and object detection, SML requires exact satisfaction of important physical characteristics. Recent work has developed various deep-learning models that encode such physical characteristics, that are physically-consistent (e.g., by enforcing conservation laws (Raissi et al., 2019; Lee & Carlberg, 2021) or preserving structures (Greydanus et al., 2019; Toth et al., 2019; Lutter et al., 2018; Cranmer et al., 2020b; Lee et al., 2021)) and symmetrical (e.g., modeling invariance or equivariance by design (Battaglia et al., 2018; Satorras et al., 2021)). Among those methods, physics-informed neural networks (PINNs) (Raissi et al., 2019) are gaining traction in the research community because of their sound computational formalism to enforce governing physical laws to learn solutions. PINNs are also easy to implement by using automatic differentiation (Baydin et al., 2018) and gradient-based training algorithms that are readily available in any deep-learning frameworks, such as PyTorch (Paszke et al., 2019) or TensorFlow (Abadi et al., 2016).

PINNs parameterize the solution $u(x,t)$ of partial differential equations (PDEs) using a neural network $u_\Theta(x,t)$ that takes the spatial and temporal coordinates (x,t) as input and has Θ as the model parameters. During training, the neural network minimizes a *PDE residual loss* (cf. Eq. (12)) denoting the governing equation, at a set of collocation

[1]Yonsei University [2]Arizona State University [3]LG CNS [4]Texas A&M University-Commerce [5]Oregon State University [6]KAIST. Correspondence to: Noseong Park <noseong@kaist.ac.kr>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

points, and a *data matching loss* (cf. Eqs. (11) and (13)), which enforces an initial condition and a boundary condition, at another set of collocation points sampled from initial/boundary conditions. This computational formalism enables to infuse the physical laws, described by the governing equation $\mathcal{F}(x, t, u)$, into the solution model and, thus, is denoted as "physics informed." PINNs have shown to be effective in solving many different PDEs, such as Navier–Stokes equations (Shukla et al., 2021; Jagtap & Karniadakis, 2020; Jagtap et al., 2020). While powerful, PINNs suffer from several obvious weaknesses.

W1) PDE operators are highly nonlinear (making training extremely difficult);

W2) Repetitive trainings from scratch are needed when solutions to new PDEs are sought (even for new PDEs arising from new PDE parameters in parameterized PDEs).

There have been various efforts to mitigate each of these issues: (For addressing W1) curriculum-learning-type training algorithms that train PINNs from easy PDEs to hard PDEs[1] (Krishnapriyan et al., 2021), and (for addressing W2) meta-learning PINNs (Liu et al., 2022); or directly learning solutions of parameterized PDEs such that $u_\Theta(x, t; \boldsymbol{\mu})$, where $\boldsymbol{\mu}$ is a set of PDE parameters, e.g., $\boldsymbol{\mu} = [\beta, \nu, \rho]$ in convection-diffusion-reaction (CDR) equations. However, there has been a less focus on addressing both problems in a unified PINN framework.

To mitigate the both issues in W1 and W2 simultaneously, we propose a variant of PINNs for solving parameterized PDEs, called *parameterized physics-informed neural networks* (P^{2}INNs). P^{2}INNs approximate solutions as a neural network of a form $u_\Theta(x, t; \boldsymbol{\mu})$ (for resolving W2) and are capable of inferring approximate solutions with accuracy (cf. Table 1 and Figure 1) even for harder PDEs (for resolving W1). A novel modification proposed in our model is to explicitly extract a hidden representation of the PDE parameters by employing a separate encoder network, $\boldsymbol{h}_{\text{param}} = g_{\Theta_p}(\boldsymbol{\mu})$, and uses this hidden representation to parameterize the solution neural network, $u_\Theta(x, t; \boldsymbol{h}_{\text{param}})$. Rather than simply treating $\boldsymbol{\mu}$ as a coordinate in the parameter domain, we infer useful information of PDEs from the PDE parameters $\boldsymbol{\mu}$, constructing the latent manifold on which the hidden representation of each PDE lies.

To demonstrate the effectiveness of the proposed model, we demonstrate the performance of the proposed model with well-known benchmarks (Krishnapriyan et al., 2021), i.e., parameterized CDR equations. As studied in (Krishnapriyan et al., 2021), certain choices of the PDE parameters (e.g., high convective or reaction term) make training PINNs very challenging (i.e., harder PDEs), and our goal is to show that the proposed method is capable of producing approximate solutions with reasonable accuracy for those harder PDEs.

To sum up, our contributions are as follows:

- We design a novel neural network architecture for solving parameterized PDEs, P^{2}INNs, which significantly improves the performance of PINNs overcoming the well-known weaknesses (W1 and W2).
- We empirically demonstrate that explicitly encoding the PDE parameters into a hidden representation plays an important role in improving performance.
- We show that the proposed P^{2}INNs are able to learn all the experimented benchmark PDEs via a single training run and greatly outperforms existing PINN-based methods in terms of prediction accuracy.

2. Background and Motivation

We start by providing illustrative examples of parameterized PDEs and their solutions to motivate a development of a new efficient variant of PINNs for *multi-query* and *real-time* scenarios. Details on the PDEs can be found in Appendix.

2.1. Convection-Diffusion-Reaction Equations

As an example, we consider parameterized CDR equations:

$$\frac{\partial u}{\partial t} + \beta \frac{\partial u}{\partial x} - \nu \frac{\partial^2 u}{\partial x^2} - \rho u(1-u) = 0, \; x \in \Omega, \; t \in [0, T]. \quad (1)$$

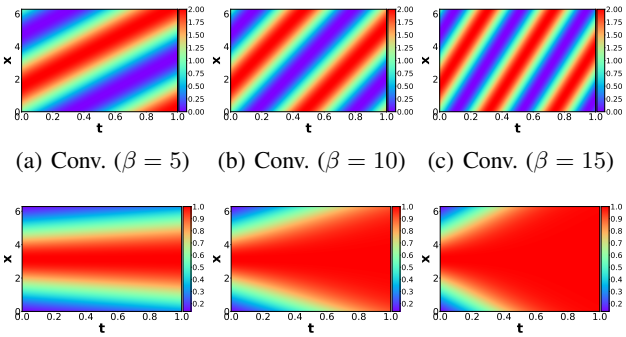
The equation describes how the state variable u changes over time with the existence of convective (the second term), diffusive (the third term), and reactive (the fourth term) phenomena. Here, β is a coefficient about how fast transportable the equation is, ν is a diffusivity for the diffusion phase, and ρ is a scaling parameter about spreading velocity. Note that we choose the well-known Fisher's form $\rho u(1-u)$, which was used in (Krishnapriyan et al., 2021), as our reaction term.

We note that we choose this class of PDEs due to many advantages: (1) solution characteristics are varying significantly based on PDE parameters, (2) some PDE parameter values introduce challenging situations for PINNs (a.k.a "failure modes"), and (3) analytical solutions exist. However, we also note that our proposed method is not specifically limited to this PDE class, but is applicable to general PDE classes (e.g., see Section 4.3 for 2D cases).

2.2. Motivation

Our goal is to develop a method to solve parameterized PDEs via the computational formalism of PINNs' overcoming W1 and W2. With this in mind, we first attempt to obtain

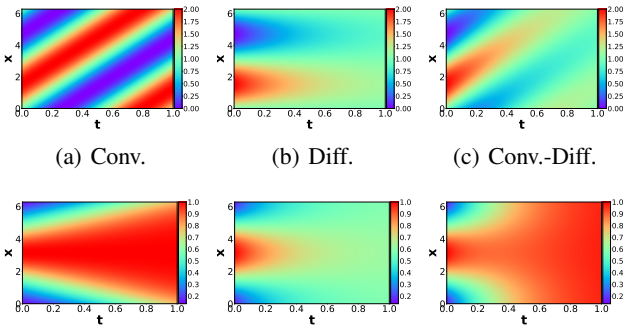
[1] Following the notational conventions in curriculum learning, we use the terms, "easy" and "hard," to indicate data that are easy or hard for neural networks to learn.



(a) Conv. ($\beta = 5$) (b) Conv. ($\beta = 10$) (c) Conv. ($\beta = 15$)

(d) Reac. ($\rho = 1$) (e) Reac. ($\rho = 4$) (f) Reac. ($\rho = 7$)

Figure 2. The ground-truth solutions of various convection equations with an initial condition of $1 + \sin(x)$ (Figure 2. (a)-(c)) and reaction equations with an initial condition of a Gaussian distribution $N(\pi, (\pi/2)^2)$ (Figure 2. (d)-(f)). We note that varied solutions are made (with similar architectures) depending on changes in coefficient.



(a) Conv. (b) Diff. (c) Conv.-Diff.

(d) Reac. (e) Diff. (f) Reac.-Diff.

Figure 3. The ground-truth solutions of various CDR equations with an initial condition of $1 + \sin(x)$ (Figure 3. (a)-(c)) or a Gaussian distribution $N(\pi, (\pi/2)^2)$ (Figure 3. (d)-(f)). We note that the solution in the last column reflects the first two columns' solutions. Therefore, there also exist similarities across different equation types.

intuitions from the visual inspection of solution snapshots displayed on the (x, t)-coordinate space (Figures 2 and 3).

The first set of the examples is shown in Figure 2: The ground-truth solutions of convection equations (top row) and reaction equations (bottom row) with varying parameters β and ρ, respectively. As we vary the PDE parameter, e.g., increasing β, we obtain gradually changing solutions (i.e., becoming more oscillatory, as we go left from Figure 2(a) to Figure 2(c)). This suggests that model parameters of PINNs for varying PDE parameters could have similar values and this can be leveraged in the training of PINNs.

This observation indeed has been investigated in (Krishnapriyan et al., 2021) to solve hard PDEs for PINNs. With a higher convective term (large β), the PDE becomes a hard problem for PINNs to solve due to the spectral bias (Rahaman et al., 2019) (i.e., the solution is highly oscillatory in time). Thus, (Krishnapriyan et al., 2021) proposed a curriculum-learning algorithm which starts to feed an easier PDE and gradually increases β until it reaches the target value. This approach, however, drops all the intermediate model parameters obtained in the course of training. Instead, in our approach, we utilize all PDE information to train a single model for the solutions of parameterized PDEs.

The second set of examples (Figure 3, the solutions of different types of PDEs) provide a similar observation as above: even for different classes of PDEs (e.g., convection, diffusion, and convection-diffusion equations), the solutions gradually change, which can be leveraged in training PINNs.

Motivation #1: a latent space of parameterized PDEs may exist. Since PDEs with similar parameter settings share common characteristics, we conjecture that solutions of parameterized PDEs can be embedded onto a latent space and reconstructed by using a shared decoder network.

Motivation #2: it will be more effective to solve similar problems simultaneously. Considering the similarities between solutions parameterized by similar PDE parameters, we conjecture that training can be improved by attempting to solve all those similar problems together — multi-task learning approaches are also based on the same intuition (Kendall et al., 2018; Ruder, 2017).

Motivated by the observations, we develop a new approach that alleviates the two known weaknesses W1 and W2.

3. P^{2}INNs: Parameterized PINNs

Now we introduce our parameterized physics-informed neural networks (P^{2}INNs). In essence, our goal is to design a neural network architecture that effectively emulates the action of the parameterized PDE solution function, $u(x, t; \boldsymbol{\mu})$.

3.1. Model Architecture

For P^{2}INNs, we propose a modularized design of the neural network $u_\Theta(x, t; \boldsymbol{\mu})$, which consists of three parts, i.e., two separate encoders g_{θ_p} and g_{θ_c}, and a manifold network g_{θ_g} such that

$$u_\Theta(x, t; \boldsymbol{\mu}) = g_{\theta_g}([g_{\theta_c}(x, t); g_{\theta_p}(\boldsymbol{\mu})]), \tag{2}$$

where $\Theta = \{\theta_c, \theta_p, \theta_g\}$ denotes the set of model parameters. The two encoders, g_{θ_c} and g_{θ_p}, take the spatiotemporal coordinate (x, t) and the PDE parameters $\boldsymbol{\mu}$ as inputs and extract hidden representations such that $\boldsymbol{h}_{\text{coord}} = g_{\theta_c}(x, t)$ and $\boldsymbol{h}_{\text{param}} = g_{\theta_p}(\boldsymbol{\mu})$. The two extracted hidden representations are then concatenated and fed into the manifold network to

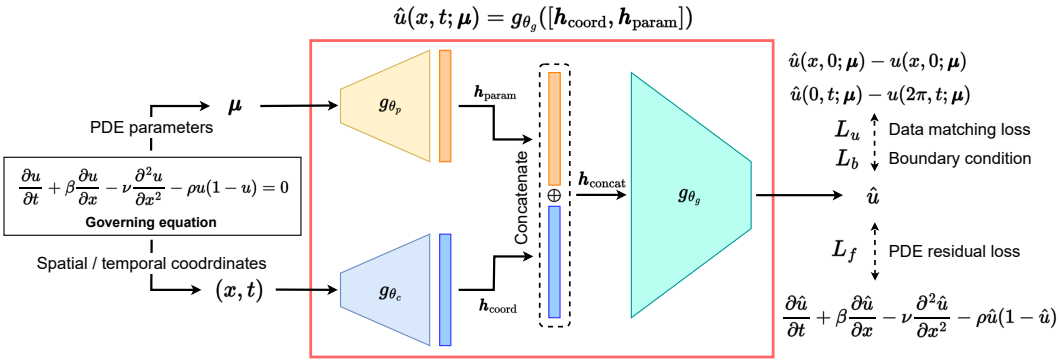


Figure 4. **P^{2}INNs architecture.** The two encoders g_{θ_p} and g_{θ_c} are added to generate better representations for the PDE parameter and the spatial/temporal coordinate. We also customize the manifold network g_{θ_g}. In this figure, we provide the CDR equation as an example.

infer the solution of of the PDE with the parameters $\boldsymbol{\mu}$ at the coordinate (x,t), i.e., $\hat{u}(x,t;\boldsymbol{\mu}) = g_{\theta_g}([\boldsymbol{h}_{\text{coord}}; \boldsymbol{h}_{\text{param}}])$. Figure 4 summarizes the P^{2}INNs architecture.

The important design choice here is that we explicitly encode the PDE parameters into a hidden representation as opposed to treating the PDE parameters merely as a coordinate in the parameter domain, e.g., $(x,t,\boldsymbol{\mu})$ is combined and directly fed into our ablation model, called PINN-P, for our ablation study in Section 4.2.3. With the abuse of notation, P^{2}INNs can be expressed as a function of (x,t), parameterized by the hidden representation: $u_\Theta(x,t;\boldsymbol{\mu}) = u_{\{\theta_c,\theta_g\}}(x,t;\boldsymbol{h}_{\text{param}})$. This expression emphasizes our intention that we explicitly utilize the PDE model parameters to characterize the behavior of the solution neural network.

3.1.1. Encoder for Equation Input

The equation encoder g_{θ_p} reads the PDE parameters, and generates a hidden representation of the equation, denoted as $\boldsymbol{h}_{\text{param}}$. We employ the following fully-connected (FC) structure for the encoder:

$$\boldsymbol{h}_{\text{param}} = \sigma(FC_{D_p} \cdots (\sigma(FC_2(\sigma(FC_1(\boldsymbol{\mu}))))))), \quad (3)$$

where σ denotes a non-linear activation, such as ReLU and tanh, and FC_i denotes the i-th FC layer of the encoder. D_p means the number of FC layers.

We note that $\boldsymbol{h}_{\text{param}}$ has a size larger than that of $\boldsymbol{\mu}$ in our design to encode the space and time-dependent characteristics of the parameterized PDE. Since highly non-linear PDEs show different characteristics at different spatial and temporal coordinates, we intentionally employ relatively high-dimensional encoding.

3.1.2. Encoder for Spatiotemporal Coordinate

The spatial and temporal coordinate encoder g_{θ_c} generates a hidden representation $\boldsymbol{h}_{\text{coord}}$ for (x,t). This encoder has the following FC layer structure:

$$\boldsymbol{h}_{\text{coord}} = \sigma(FC_{D_c} \cdots (\sigma(FC_2(\sigma(FC_1(x,t)))))), \quad (4)$$

where FC_i and D_c denote the i-th FC layer of this encoder and the number of FC layers, respectively.

3.1.3. Manifold Network

The manifold network g_{θ_g} reads the two hidden representations, $\boldsymbol{h}_{\text{param}}$ and $\boldsymbol{h}_{\text{coord}}$, and infer the input equation's solution at (x,t), denoted as $\hat{u}(x,t;\boldsymbol{\mu})$. With the inferred solution $\hat{u}$, we construct two losses, L_u and L_f. The manifold network can have various forms but we use the following form:

$$\hat{u}(x,t;\boldsymbol{\mu}) = \sigma(FC_{D_g} \cdots \sigma(FC_1(\boldsymbol{h}_{\text{concat}}))), \quad (5)$$

where $\boldsymbol{h}_{\text{concat}} = \boldsymbol{h}_{\text{coord}} \oplus \boldsymbol{h}_{\text{param}}$, and $\oplus$ is the concatenation of the two vectors; D_g denotes the number of FC layers.

3.2. Training

Model training is performed by minimizing the regular PINN loss. With the prediction $\hat{u}$ produced by P^{2}INNs, our basic loss function consists of three terms as follows:

$$L(\Theta) = w_1 L_u + w_2 L_f + w_3 L_b, , \quad (6)$$

where L_u, L_b, and L_f enforces initial, boundary conditions, and physical laws in PDEs, respectively, and $w_1, w_2, w_3 \in \mathbb{R}$ are hyperparameters. In general, the overall training method follows the training procedure of the original PINN (Raissi et al., 2019). The only exception is that the PDE residual loss associated with multiple PDEs is minimized in a mini-batch whereas in the original PINN, the residual of only one PDE is minimized. To be more specific, in each iteration, we create a mini-batch of $\{\boldsymbol{\mu}_i, (x_i,t_i)\}_{i=1}^B$, where B is a mini-batch size. We randomly sample the collocation points and, thus, there can be multiple different PDEs, identified by $\boldsymbol{\mu}_i$, in a single mini-batch.

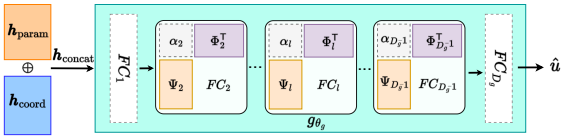


Figure 5. **P^{2}INNs with SVD modulation.** From the pre-trained decoder layer of P^{2}INN, we obtain the bases Φ_l, Ψ_l for parameterized PDEs through SVD (cf. Eq. (7)). Note that only the diagonal matrices α_l are used for fine-tuning. (The dotted lines represent learnable parameters.)

3.3. Fast Fine-tuning

As the ultimate goal in this study is to deploy trained models to a set of specific PDE parameters of our interest, we devise a method to fine-tune the trained models to improve the solution accuracy at those query PDE parameters. To this end, we adopt an idea of SVD-PINNs (Gao et al., 2022), which shows that extracting basis through singular value decomposition (SVD) from the weights of a PINN trained on a single PDE equation is effective in transferring information. Extending this insight, we introduce an *SVD modulation* by obtaining bases through applying SVD to the weights of the decoder layers of P^{2}INNs. Specifically, only the manifold network g_{θ_g} is transformed to a form that can be modulated (cf. Figure 5); each layer, excluding the first and last layers, is decomposed as follows:

$$FC_l = \Psi_l \alpha_l \Phi_l^\top, \quad l = 2, 3, \ldots, D_g - 1. \tag{7}$$

Then, during fine-turning, we set $\{\alpha_l\}_{l=2}^{D_g-1}$ to be learnable, while keeping all other parameters in the network fixed. It is an option to fix the parameters of FC_1 and FC_{D_g}.

In the field of implicit neural representations, where the study on learning coordinate-based continuous neural function is conducted, the *shift modulation* (Dupont et al., 2022) has been one of leading architectural choices. This involves adding a shift term to the bias of each layer in the model, to better represent various data with a small number of learnable parameter. However, we found from empirical experiments that modulating by shift in PINNs does not lead to significant performance improvement. We discuss this further in Section 4.2.4.

4. Evaluation

In this section, we test the performance of P^{2}INNs on the benchmark PDE problems: 1D CDR equations and 2D Helmholtz equations, both of which are known to suffer from the failure modes. We first layout our experimental setup and show that P^{2}INNs outperform the baselines with an extensive evaluation. We further analyze how P^{2}INNs address the failures shown in Section 2. Due to space reasons, detailed experimental setups and results are in Appendix.

4.1. Experimental Setup

Datasets. For simplicity but without loss of generality, we assume the parameterized 1D CDR equations and 2D Helmholtz equations (cf. Eqs. (1) and (8)). To generate the ground-truth data, we use either analytic or numerical solutions. In case of 1D CDR equations, we analyze the target equations with three types of initial conditions $u(x, 0)$: two Gaussian distributions of $N(\pi, (\pi/2)^2)$ and $N(\pi, (\pi/4)^2)$, and a sinusoidal function of $1 + \sin(x)$. To solve the equation, we use the Strang splitting method (Strang, 1968). For 2D Helmholtz equations, we obtain the exact solution by calculating it directly.

Baseline and Ablation Methods. We compare P^{2}INNs with three baselines. PINN is the original design based on fully-connected layers with non-linear activations in (Raissi et al., 2019), and PINN-R is its enhancement by using residual connections, which was used in (Kim et al., 2021). PINN-seq2seq (Krishnapriyan et al., 2021) is a model that applies the seq2seq learning method to the PINN model, sequentially learning data over time. We divided the entire time into 10 steps. In addition, we define one ablation model for our method, called PINN-P, which has the same structure as original PINN, but the PDE parameters $\boldsymbol{\mu}$ is treated as a coordinate in the parameter space, i.e., $(x, t, \boldsymbol{\mu})$.

Methodology. We train PINN and PINN-R for each parameter configuration in each equation type, following the standard PINN training method — in other words, there are as many models as the number of PDE parameter configurations for an equation type. To train P^{2}INNs, however, we train it with all the initial conditions and collocation points of the multiple parameter configurations in each equation type, following the training method in Section 3.2. Therefore, we have only one model in each equation type.

Metrics. To evaluate the performance of the model, we measure the L_2 relative and absolute errors between the solution predicted by the model and the analytic solution. The relative error and the absolute error of the i-th equation are defined as the averages of $\|\hat{\boldsymbol{u}}_i - \boldsymbol{u}_i\|_2 \,/\, \|\boldsymbol{u}_i\|_2$ and $\|\hat{\boldsymbol{u}}_i - \boldsymbol{u}_i\|_2$, where $i \in \{1, ..., N_e\}$ and N_e is the number of equations used for the task. At this time, the errors are measured for each test points and the average value is used. In addition, we use max error and explained variance score for further analysis (cf. Table 12). We test with 3 seed numbers and report their mean.

4.2. 1D CDR Equations

In the experiments, we employ 6 different equation types stemmed from CDR equations (cf. Section 2.1) with the varying parameters as listed in Table 5, and the experimental results are summarized in Table 2. Whereas existing

Table 2. The relative and absolute L_2 errors over all the equations. Our P^{2}INNs surpass baselines in all but one cases, even without fine-tuning. IMP. denotes the rate of improvement of our model over the best baseline.

	PDE type	Coefficient range	Metric	PINN	PINN-R	PINN-seq2seq	P^{2}INN	IMP. (%)
Class 1	Convection	1∼5	Abs. err.	0.0183	0.0222	0.1281	**0.0039**	78.44
			Rel. err.	0.0327	0.0381	0.2160	**0.0079**	75.82
		1∼10	Abs. err.	0.0164	0.0666	0.1924	**0.0093**	43.62
			Rel. err.	0.0307	0.1195	0.3276	**0.0179**	41.78
		1∼20	Abs. err.	0.1140	0.1624	0.2252	**0.0198**	82.64
			Rel. err.	0.1978	0.2779	0.3819	**0.0464**	76.55
	Diffusion	1∼5	Abs. err.	0.1335	0.1665	0.1987	**0.1322**	0.97
			Rel. err.	0.2733	0.3462	0.4050	**0.2710**	0.84
		1∼10	Abs. err.	0.2716	0.3175	0.3149	**0.1539**	43.34
			Rel. err.	0.5259	0.6206	0.6174	**0.3116**	40.75
		1∼20	Abs. err.	0.6782	0.7054	0.3346	**0.1916**	42.74
			Rel. err.	1.2825	1.3401	0.6442	**0.3745**	41.87
	Reaction	1∼5	Abs. err.	0.3341	0.3336	0.4714	**0.0015**	99.54
			Rel. err.	0.3907	0.3907	0.5907	**0.0027**	99.31
		1∼10	Abs. err.	0.6232	0.3619	0.6924	**0.0065**	98.19
			Rel. err.	0.6926	0.4190	0.7931	**0.0089**	97.88
		1∼20	Abs. err.	0.7902	0.4320	0.8246	**0.0042**	99.02
			Rel. err.	0.8460	0.4932	0.8960	**0.0092**	98.14
Class 2	Conv.-Diff.	1∼5	Abs. err.	0.0610	0.0654	0.0979	**0.0399**	34.61
			Rel. err.	0.1175	0.1289	0.1950	**0.0892**	24.05
		1∼10	Abs. err.	0.1133	0.1313	0.0917	**0.0576**	37.25
			Rel. err.	0.2098	0.2510	0.1959	**0.1320**	32.62
		1∼20	Abs. err.	0.2735	0.2118	0.0645	**0.0622**	3.51
			Rel. err.	0.5106	0.4154	0.1504	**0.1485**	1.28
	Reac.-Diff.	1∼5	Abs. err.	0.1900	0.1876	0.4201	**0.1225**	34.70
			Rel. err.	0.2702	0.2777	0.5346	**0.1856**	31.31
		1∼10	Abs. err.	0.5166	0.3809	0.6288	**0.1833**	51.88
			Rel. err.	0.6141	0.4790	0.7274	**0.2756**	42.46
		1∼20	Abs. err.	0.7167	0.7210	0.7663	**0.0898**	81.03
			Rel. err.	0.7998	0.8105	0.8337	**0.1411**	74.68
Class 3	Conv.-Diff.-Reac.	1∼5	Abs. err.	0.1663	0.0865	0.4943	**0.0311**	64.02
			Rel. err.	0.2057	0.1415	0.6104	**0.0525**	62.88
		1∼10	Abs. err.	0.5321	0.3170	0.7051	**0.0508**	83.98
			Rel. err.	0.5928	0.3772	0.8027	**0.0939**	75.10
		1∼20	Abs. err.	0.7450	0.4080	0.7136	**0.0353**	91.94
			Rel. err.	0.7960	0.4645	0.8100	**0.0812**	82.88

baselines show fluctuating performance, our P^{2}INNs show stable performance for all the 6 different equation types. The most notable accuracy differences are made for the diffusion, the reaction, the reaction-diffusion, and the convection-diffusion-reaction equations.

For instance, PINN-R marks an absolute error of 0.4320 whereas P^{2}INN achieves an error of 0.0042 for the reaction equations with the coefficient range of 1 to 20, i.e., 102 time smaller error. The smallest accuracy differences happen for the diffusion equations with the coefficient range of 1 to 5. While PINN and P^{2}INN show similar performance, our method much better predict reference solution for the range of 1 to 20, i.e., an error or 0.6782 by PINN vs. 0.1916 by P^{2}INN. Since large coefficients incur equations difficult to solve, all existing baselines commonly fail in the range. In all cases, our method outperforms PINNs, depending on the equation type, by 33% to 99% as reported in Table 1. For the reaction equations, the improvement ratio by our method is significant.

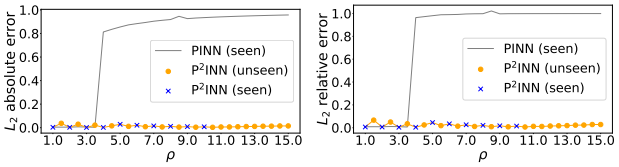
4.2.1. Inferring Solutions of Unseen PDE Parameters

We further evaluate our P^{2}INNs in more challenging situations: testing trained models on PDE parameters that are unseen during training, which can be considered as *real-time multi-query* scenarios.

For reaction equations, we train P^{2}INNs on $\rho \in [1, 10]$ with interval 1 and conduct interpolation on $\rho \in [1.5, 9.5]$ with interval 1 and extrapolation on $\rho \in [10.5, 15]$ with interval 0.5. As shown in Figure 6, PINNs' failure for $\rho > 4$ contrasts P^{2}INNs' exceptional performance, demonstrating its resilience in extrapolation, not limited good performance only for learned or closely aligned parameters.

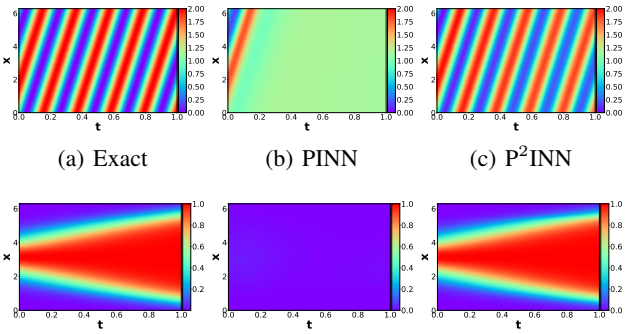
4.2.2. P^{2}INNs in PINN's Failure Modes

It is well known that PINNs have several failure cases. In particular, CDR equations with large coefficients are no-



(a) L_2 absolute error (b) L_2 relative error

Figure 6. [Reaction equation] Interpolation and extrapolation results for unseen ρ.



(a) Exact (b) PINN (c) P^{2}INN

(d) Exact (e) PINN (f) P^{2}INN

Figure 7. Failure modes in the convection equation of $\beta = 30$ (a-c), and the reaction equation of $\rho = 5$ (d-f). P^{2}INNs much more accurately predict reference solutions.

toriously hard to solve with PINNs (Krishnapriyan et al., 2021). Among the reported failure cases of PINN, the convection equation with $\beta = 30$ and the reaction equation with $\rho = 5$ are two representative ones — in particular, $\beta = 30$ corresponds to an extrapolation task after being trained for up to $\beta = 20$, which is considered as one of the most challenging task. These two equations generate signals sharply fluctuating over time. As shown in Figure 7 and Table 9, therefore, PINNs fail to predict reference solutions whereas our method almost exactly reproduce them (cf. Appendix F).

4.2.3. Ablation Studies

As an ablation study, we do not separately encode $(x, t, \boldsymbol{\mu})$ but directly feed it into our ablation model, PINN-P (i.e., employing a single encoder network $g(x, t, \boldsymbol{\mu})$ without explicitly having an encoder for PDE parameters, $g_{\theta_p}(\mu)$). We test on reaction equations, which are hard for PINN baselines to solve, and summarize the results in Table 3. As shown in Table 3, especially in a wide coefficient range, P^{2}INNs outperforms the ablation model, justifying our model design to separately encode the PDE parameters and the spatial/temporal coordinate. More details of the experiments and other ablation studies are in Appendix.

Table 3. Ablation study on the reaction equations.

Coefficient range	PINN-P		P^{2}INN	
	Abs. err.	Rel. err.	Abs. err.	Rel. err.
1∼5	0.0083	0.0113	0.0015	0.0027
1∼20	0.8975	0.9908	0.0042	0.0092

Table 4. Experimental results of modulations with an initial condition of a Gaussian distribution $N(\pi, (\pi/2)^2)$.

PDE type	Metric	PINN (best)	P^{2}INN	Modulation All	Modulation Shift	Modulation SVD
Convection	Abs. err.	0.0183	0.0174	0.1959	0.0139	**0.0138**
	Rel. err.	0.0327	0.0316	0.3319	0.0248	**0.0246**
Reaction	Abs. err.	0.3336	0.0126	0.0713	0.0095	**0.0089**
	Rel. err.	0.3907	0.0229	0.1211	0.0198	**0.0184**
Conv.-Diff.-Reac.	Abs. err.	0.0865	0.0315	0.0463	0.0321	**0.0303**
	Rel. err.	0.1415	0.0508	0.0690	0.0521	**0.0486**

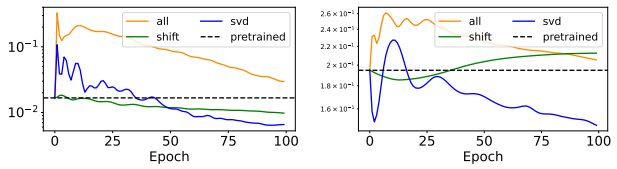
4.2.4. P^{2}INN Learned Various Equation Types

Now, we test the proposed model on a more challenging case, i.e., learning a single solution network for all six different types of CDR equations (cf. Section 2.1) at the same time. We compare the performance of our proposed PINN-specific modulation method (cf. Section 3.3) against updating all the parameters of the network (denoted as "All") and shift modulation (Dupont et al., 2022) (denoted as "shift"). That is, using P^{2}INN trained with a range of 1 to 5 as a pretrained model, we test how the each method fine-tunes the pretrained model. For the experiment, we fine-tune only for 15 epochs on each type of CDR equations and summarize the results of convection, reaction, and convection-diffusion-reaction equations in Table 4. The full experimental results are in Appendix F.3.

As shown in Table 4, P^{2}INN accurately approximates and distinguishes the solution for each PDE type even in these challenging scenarios. Moreover, our SVD-based modulation outperforms all other baselines, including shift modulation and the pretrained model. Additionally, Figure 8, shows how each model infers solutions of seen and unseen PDE parameters at first 100 epochs. For both seen and unseen PDE parameters, SVD-based methods show the lowest L_2 absolute errors, proving its generalizability and robustness. In other words, we demonstrate that through our SVD modulation, P^{2}INN can be adapted to various PDEs with small number of trainable parameters and only a few epochs.

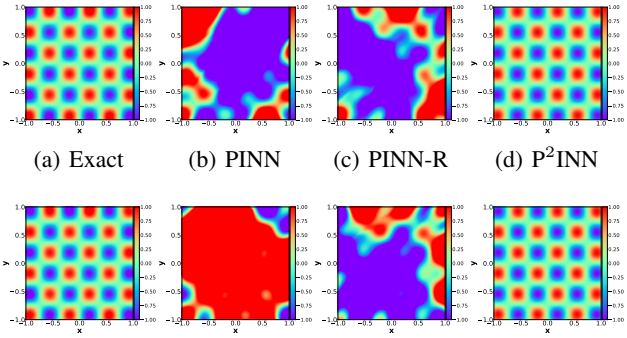
4.3. 2D Helmholtz Equations

For 2D Helmholtz equations, we train models with $a \in [2.5, 3.0]$ with interval 0.1, and then test them with interval 0.05. Notably, as shown in Figure 9, P^{2}INNs consistently shows good performance in both cases where a is a seen parameter($a = 2.7$) and an unseen($a = 2.75$) parameter.



(a) $\beta = 3$ (seen) (b) $\beta = 8$ (unseen)

Figure 8. [Convection equation] Comparison of L_2 absolute errors based on modulation methods with an initial condition of a Gaussian distribution $N(\pi, (\pi/2)^2)$



(a) Exact (b) PINN (c) PINN-R (d) P^{2}INN

(e) Exact (f) PINN (g) PINN-R (h) P^{2}INN

Figure 9. [2D-Helmholtz equation] Exact solutions and results of baselines and P^{2}INN for $a = 2.7$ (seen) (a-d) and $a = 2.75$ (unseen) (e-h)

However, PINN and PINN-R struggle, despite the fact that both of these values are within the seen(trained) parameter range for two models.

These outcomes underscore our method's capacity to deliver robust solutions, a characteristic that extends to unexplored parameter spaces. Therefore, when solving equations in some coefficient range, P^{2}INNs offer exceptional efficiency as they only require testing within the learned latent space, eliminating the need for additional training. On top of that, the experiment on 2D PDEs reaffirms the robustness and efficacy of our proposed P^{2}INN approach in higher-dimensional settings, where there are non-trivial boundary conditions. More results are listed in Appendix G.

5. Related Work

Machine Learning Methods for Solving Partial Differential Equations. Traditional numerical methods such as finite element methods and finite difference methods have clear pros and cons (Patidar, 2016; Li & Bettess, 1997; Srirekha et al., 2010). The more accurate the results, the more expensive the calculation of numerically approximated formulas. It means that to earn more accurate solutions, it needs to use finer grids, which implies more cost. To alleviate those cons, researchers were attracted to machine learning approaches (Karniadakis et al., 2021; Cuomo et al., 2022). After various trials like using the Galerkin or Ritz method (Rudd & Ferrari, 2015), PINNs proposed a transformative way of using deep learning for solving general governing PDEs in a physically sound and easy-to-formulate computational formalism (Raissi et al., 2019). As elaborated above, PINNs, however, possess weaknesses which must be addressed (Krishnapriyan et al., 2021): (1) there are classes of PDEs that it is difficult for PINNs to learn (e.g., PDEs exhibiting high oscillation or sharp transitions in spatial and/or temporal domains) and (2) gradient-based training often converges to a local optimum of models. Another line of research for solving PDEs is to analyze operator learning for differential equation or deep Ritz methods (Yu et al., 2018; Li et al., 2020; Gupta et al., 2021) but PINNs still have its potential for mainly focusing on governing equations which describe physical phenomena.

Physics as Inductive Biases. There have been various strategies to impose physical constraints on neural networks (Cranmer et al., 2020a; Rudd & Ferrari, 2015; Lee et al., 2021). Most of them focus on imposing constraints for outputs or injecting specific physical conditions into neural networks. As a simple but effective solution, PINNs directly impose physical conditions into neural networks by using a governing equation itself as a loss (Raissi et al., 2019). This loss function is called L_f. In this way, PINNs can learn the residual error of the governing equation. If initial conditions are given, we can add an initial error loss term L_u. Furthermore, if there are specific boundary conditions, we can specify boundary conditions in L_b.

Recent Developments in PINNs. In the recent literature, PINNs have evolved in many different ways to resolve issues inherent with the vanilla PINNs. Some architectural enhancements have been made in (Cho et al., 2024b) (a low-rank extension PINNs for model efficiency and a hypernetwork for effective training) and in (Cho et al., 2024a) (a separable design of model parameters for efficient training). A systematic assessment for PINNs and a new sample strategie have been investigated in PINNACLE (Lau et al., 2023). There have been some effort to combine PINNs with symbolic regression in universal PINNs (Podina et al., 2023) and to devise a preconditioner for PINNs from an PDE operator preconditioning perspective (De Ryck et al., 2023). Lastly, novel optimizers for effective training of PINNs have been proposed in (Yao et al., 2023) (MultiAdam) and (Müller & Zeinhofer, 2023) (based on natural gradient descent).

6. Conclusions

PINN is a highly applicable and promising technology for many engineering and scientific domains. In particular, it

has the strength that training is possible only with a PDE to be solved, without any additional data. However, due to the highly nonlinear characteristic of PDEs, PINNs show very poor performance in certain parameterized PDE problems. In addition, there is a weakness that the model must be re-trained from scratch to analyze a new PDE. To solve these chronic issues, we propose parameterized physics-informed neural networks (P^2INNs), which can learn similar parameterized PDEs simultaneously. Through this approach, it is possible to overcome the failure situation of PINNs that could not be solved in previous studies. To ensure that it is effective in general cases, we use more than thousands of CDR equations and show that P^2INNs outperform baselines in almost all cases of the benchmark PDEs.

Impact Statement

Our method can significantly reduce the energy consumption in training PINNs for many equations. When solving for an one trivial equation only, however, existing PINN methods show better efficiency.

Acknowledgements

This work was partly supported by Samsung Electronics Co., Ltd. (No. G01240136, KAIST Semiconductor Research Fund (2nd)), the Korea Advanced Institute of Science and Technology (KAIST) grant funded by the Korea government (MSIT) (No. G04240001, Physics-inspired Deep Learning), and Institute for Information & Communications Technology Planning & Evaluation (IITP) grants funded by the Korea government (MSIT) (No. RS-2020-II201361, Artificial Intelligence Graduate School Program (Yonsei University)). K. Lee acknowledges support from the U.S. National Science Foundation under grant IIS 2338909.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. {TensorFlow}: A system for {Large-Scale} machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pp. 265–283, 2016.

Baker, N., Alexander, F., Bremer, T., Hagberg, A., Kevrekidis, Y., Najm, H., Parashar, M., Patra, A., Sethian, J., Wild, S., et al. Workshop report on basic research needs for scientific machine learning: Core technologies for artificial intelligence. Technical report, USDOE Office of Science (SC), Washington, DC (United States), 2019.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Baydin, A. G., Pearlmutter, B. A., Radul, A. A., and Siskind, J. M. Automatic differentiation in machine learning: a survey. *Journal of Marchine Learning Research*, 18:1–43, 2018.

Cho, J., Nam, S., Yang, H., Yun, S.-B., Hong, Y., and Park, E. Separable physics-informed neural networks. *Advances in Neural Information Processing Systems*, 36, 2024a.

Cho, W., Lee, K., Rim, D., and Park, N. Hypernetwork-based meta-learning for low-rank physics-informed neural networks. *Advances in Neural Information Processing Systems*, 36, 2024b.

Cranmer, M., Greydanus, S., Hoyer, S., Battaglia, P., Spergel, D., and Ho, S. Lagrangian neural networks, 2020a. URL https://arxiv.org/abs/2003.04630.

Cranmer, M., Greydanus, S., Hoyer, S., Battaglia, P., Spergel, D., and Ho, S. Lagrangian neural networks. In *ICLR 2020 Workshop*, 2020b.

Cuomo, S., di Cola, V. S., Giampaolo, F., Rozza, G., Raissi, M., and Piccialli, F. Scientific machine learning through physics-informed neural networks: Where we are and what's next, 2022. URL https://arxiv.org/abs/2201.05624.

De Ryck, T., Bonnet, F., Mishra, S., and de Bezenac, E. An operator preconditioning perspective on training in physics-informed machine learning. In *The Twelfth International Conference on Learning Representations*, 2023.

Dupont, E., Kim, H., Eslami, S. M. A., Rezende, D. J., and Rosenbaum, D. From data to functa: Your data point is a function and you can treat it like one. In *39th International Conference on Machine Learning (ICML)*, 2022.

Gao, Y., Cheung, K. C., and Ng, M. K. Svd-pinns: Transfer learning of physics-informed neural networks via singular value decomposition. In *2022 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, December 2022. doi: 10.1109/ssci51031.2022.10022281. URL http://dx.doi.org/10.1109/SSCI51031.2022.10022281.

Greydanus, S., Dzamba, M., and Yosinski, J. Hamiltonian neural networks. *Advances in Neural Information Processing Systems*, 32:15379–15389, 2019.

Gupta, G., Xiao, X., and Bogdan, P. Multiwavelet-based operator learning for differential equations. *Advances in Neural Information Processing Systems*, 34, 2021.

Jagtap, A. D. and Karniadakis, G. E. Extended physics-informed neural networks (xpinns): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. *Communications in Computational Physics*, 28(5):2002–2041, 2020.

Jagtap, A. D., Kharazmi, E., and Karniadakis, G. E. Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 365:113028, 2020.

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., and Yang, L. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

Kendall, A., Gal, Y., and Cipolla, R. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7482–7491, 2018.

Kim, J., Lee, K., Lee, D., Jhin, S. Y., and Park, N. DPM: A novel training method for physics-informed neural networks in extrapolation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 8146–8154, 2021.

Krishnapriyan, A., Gholami, A., Zhe, S., Kirby, R., and Mahoney, M. W. Characterizing possible failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems*, 34, 2021.

Lau, G. K. R., Hemachandra, A., Ng, S.-K., and Low, B. K. H. PINNACLE: Pinn adaptive collocation and experimental points selection. In *The Twelfth International Conference on Learning Representations*, 2023.

Lee, K. and Carlberg, K. T. Deep conservation: A latent-dynamics model for exact satisfaction of physical conservation laws. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 277–285, 2021.

Lee, K., Trask, N., and Stinis, P. Machine learning structure preserving brackets for forecasting irreversible processes. *Advances in Neural Information Processing Systems*, 34, 2021.

Li, L.-y. and Bettess, P. Adaptive finite element methods: a review. 1997.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

Liu, X., Zhang, X., Peng, W., Zhou, W., and Yao, W. A novel meta-learning initialization method for physics-informed neural networks. *Neural Computing and Applications*, pp. 1–24, 2022.

Lutter, M., Ritter, C., and Peters, J. Deep lagrangian networks: Using physics as model prior for deep learning. In *International Conference on Learning Representations*, 2018.

McClenny, L. and Braga-Neto, U. Self-adaptive physics-informed neural networks using a soft attention mechanism. *arXiv preprint arXiv:2009.04544*, 2020.

Müller, J. and Zeinhofer, M. Achieving high accuracy with pinns via energy natural gradient descent. In *International Conference on Machine Learning*, pp. 25471–25485. PMLR, 2023.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. PyTorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Patidar, K. C. Nonstandard finite difference methods: recent trends and further developments. *Journal of Difference Equations and Applications*, 22(6):817–849, 2016.

Podina, L., Eastman, B., and Kohandel, M. Universal physics-informed neural networks: symbolic differential operator discovery with sparse data. In *International Conference on Machine Learning*, pp. 27948–27956. PMLR, 2023.

Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F., Bengio, Y., and Courville, A. On the spectral bias of neural networks. In *International Conference on Machine Learning*, pp. 5301–5310. PMLR, 2019.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. 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.

Rudd, K. and Ferrari, S. A constrained integration (cint) approach to solving partial differential equations using artificial neural networks. *Neurocomputing*, 155:277–285, 2015.

Ruder, S. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

Satorras, V. G., Hoogeboom, E., and Welling, M. E (n) equivariant graph neural networks. In *International Conference on Machine Learning*, pp. 9323–9332. PMLR, 2021.

Shukla, K., Jagtap, A. D., and Karniadakis, G. E. Parallel physics-informed neural networks via domain decomposition. *Journal of Computational Physics*, 447:110683, 2021.

Srirekha, A., Bashetty, K., et al. Infinite to finite: an overview of finite element analysis. *Indian Journal of Dental Research*, 21(3):425, 2010.

Strang, G. On the construction and comparison of difference schemes. *SIAM journal on numerical analysis*, 5(3):506–517, 1968.

Toth, P., Rezende, D. J., Jaegle, A., Racanière, S., Botev, A., and Higgins, I. Hamiltonian generative networks. In *International Conference on Learning Representations*, 2019.

Yao, J., Su, C., Hao, Z., Liu, S., Su, H., and Zhu, J. Multi-adam: Parameter-wise scale-invariant optimizer for multi-scale training of physics-informed neural networks. In *International Conference on Machine Learning*, pp. 39702–39721. PMLR, 2023.

Yu, B. et al. The deep ritz method: a deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1):1–12, 2018.

A. Datasets

A.1. 1D Convection-Diffusion-Reaction Equations

Each of these individual PDEs has their own importance and has been studied extensively:

1. Convection-diffusion equations are used in fluid dynamics, particle chemistry, computational finance, and so on,
2. Reaction-diffusion equations are popular in the domain of biophysics and mathematical biology,
3. Convection equations, diffusion equations, and reaction equations are for describing transport, diffusive, and reactive phenomena, respectively in simplified settings.

In total, there are six classes of Convection-Diffusion-Reaction equations, each of which has its own importance in science. For each dataset, we select 1,000 collocation points, 256 initial points, 100 boundary points, and 1,000 test points.

A.2. 2D Helmholtz Equations

$$\begin{gathered}\frac{\partial^2 u(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2} + k^2 u(x,y) - q(x,y) = 0 \\ q(x,y) = (-(a_1\pi)^2 - (a_2\pi)^2 + k^2)\sin(a_1\pi x)\sin(a_2\pi y)\end{gathered} \tag{8}$$

$$u(x,y) = k^2 \sin(a_1\pi x)\sin(a_2\pi y). \tag{9}$$

We employ the specific Helmholtz equations which were used in (McClenny & Braga-Neto, 2020) as benchmark PDEs (cf. Eq. (8)), and it can be directly solved as Eq. (9). The Helmholtz equations describe the behavior of state variable $u(x, y)$ in a 2D space, accounting for the effects of wave propagation, and a source term represented by $q(x, y)$. Here k is a parameter related to wave frequency, while a_1 and a_2 control the spatial variations of the source term. In our experiments, we set k to 1 and the parameters a_1 and a_2 to a common value a. For each dataset, we select 1,000 collocation points, 400 boundary points, and 100 test points.

B. More Details on Experimental Setup

B.1. Loss

With the prediction $\hat{u}$ produced by P^{2}INNs, our basic loss function consists of three terms as follows:

$$L(\Theta) = w_1 L_u + w_2 L_f + w_3 L_b, \tag{10}$$

and L_u, L_f and L_b are defined as follows:

$$L_u = \frac{1}{N_u}\sum_{N_u}\Big(\hat{u}(x,0;\boldsymbol{\mu}) - u(x,0;\boldsymbol{\mu})\Big)^2, \tag{11}$$

$$L_f = \frac{1}{N_f}\sum_{N_f}\Big(\mathcal{F}(x,t,\hat{u};\boldsymbol{\mu})\Big)^2, \tag{12}$$

$$L_b = \frac{1}{N_b}\sum_{N_b}\Big(\hat{u}(0,t;\boldsymbol{\mu}) - \hat{u}(2\pi,t;\boldsymbol{\mu})\Big)^2, \tag{13}$$

where N_u, N_f, and N_b are the cardinalities of the sets of initial conditions, collocation points, and boundary conditions; $w_1, w_2, w_3 \in \mathbb{R}$ are hyperparameters. The first and the second terms denote the data matching loss L_u and the PDE residual loss L_f, respectively. In addition, we separately add the boundary condition term L_b, forcing their values equal at both top and bottom parts (see Figures 2 and 3).

B.2. Baseline

Each baseline and ablation model is trained in the following way:

1. PINN, PINN-R, and PINN-seq2seq do not read PDE parameters, such as β, ν, ρ and a, but are trained separately for each of the coefficient settings.
2. PINN-P, an ablation model of P^{2}INNs, is able to process PDE parameters and is trained for all coefficient settings in each equation type.
3. Therefore, PINN, PINN-R, and PINN-seq2seq require many trained models for solving parameterized PDEs whereas PINN-P and our method require a single trained model to solve them.

B.3. Implementation

We implement P^{2}INNs with PYTHON 3.7.11 and PYTORCH 1.10.2 that supports CUDA 11.4. We run our evaluation on a machine equipped with Intel Core-i9 CPUs and NVIDIA RTX A6000 and RTX 2080 TI GPUs.

C. Model Configuration and Efficiency

C.1. Dataset Statistics

Table 5. Dataset statistics. For each equation type, we test three different coefficient ranges. In Conv.-Diff.-Reac., β, ν, ρ are non-zeros.

Coefficient range	Convection	Diffusion	Reaction	Conv.-Diff.	Reac.-Diff.	Conv.-Diff.-Reac.
1$\sim$ 5	5	5	5	25	25	125
1$\sim$10	10	10	10	100	100	1,000
1$\sim$20	20	20	20	400	400	8,000

Table 5 represents dataset statistics, and our dataset generation source code is mainly based on (Krishnapriyan et al., 2021).

C.2. Model Efficiency and Hyperparameters

Our baselines, PINN, PINN-R, and PINN-seq2seq, are designed with 6 layers, and the dimension of hidden vector is 50. For training, we employ Adam optimizers with learning rate of $1e-3$. For our method, we set D_p, D_c, and D_g to 4, 3, and 5 respectively. In the loss function in Eq. (6), we set w_1, w_2, and w_3 to 1. We use a hidden vector dimension of 50 for g_{θ_c} and g_{θ_g}, and 150 for g_{θ_p}. For g_{θ_g}. Considering that our method is able to solve multiple equations with one model, the total model size for our method is much smaller than other baselines (see Appendix M).

D. Sensitivity Analyses

Table 6. Experimental results of P^{2}INNs by varying the dimension of g_{θ_p}

Dim.	Convection		Reaction	
	Abs. err.	Rel. err.	Abs. err.	Rel. err.
80	0.0030	0.0060	0.0036	0.0054
160	0.0029	0.0054	0.0016	0.0029
320	0.0023	0.0045	0.0061	0.0082

We test P^{2}INNs by varying the hidden vector dimension of g_{θ_p} in $\{80, 160, 320\}$. Convection and reaction equations with the coefficient range of 1 to 5 are used for testing and we summarize the result in Table 6. As shown in Table 6, P^{2}INNs attain small errors in every hyperparameter setting compared to other baselines, which proves the robustness of our model.

E. Additional Experiments

E.1. Large Range

Table 7. Experimental results on reaction equation with $\rho \in [1, 50]$

Metric	PINN	PINN-R	PINN-seq2eq	P^{2}INN
Abs. err.	0.9053	0.4732	0.9190	<u>0.0486</u>
Rel. err.	0.9383	0.5211	0.9582	<u>0.1322</u>

We conduct additional experiments on the reaction equation with an initial condition of Gaussian distribution($N(\pi, (\pi/2)^2)$). In these experiments, we test on equations with $\rho \in [1, 50]$, which is an extremely wide range, to compare how models work in highly challenging scenarios in terms of range. As summarized in Table 7, P^{2}INNs surpass others, showing that P^{2}INNs even works properly in the extremely large coefficient ranges.

E.2. Comparison with Meta-learning Algorithms

Table 8. Comparison of our model with meta-learning based PINNs

PDE type	Metric	MAML	Reptile	P^{2}INN
Convection	**Abs. err.**	0.0579	0.0173	<u>0.0039</u>
	Rel. err.	0.1036	0.0347	<u>0.0079</u>
Reaction	**Abs. err.**	0.0029	0.0033	<u>0.0015</u>
	Rel. err.	0.0057	0.0064	<u>0.0027</u>

We compare P^{2}INNs with two other meta-learning methods (i.e., MAML, Reptile). We first train three models using convection and reaction equations with coefficient range of 1$\sim$5 and then fine-tune MAML and Reptile. As shown in Table 8, our model shows best performance among three models in every case, even without fine-tuning steps.

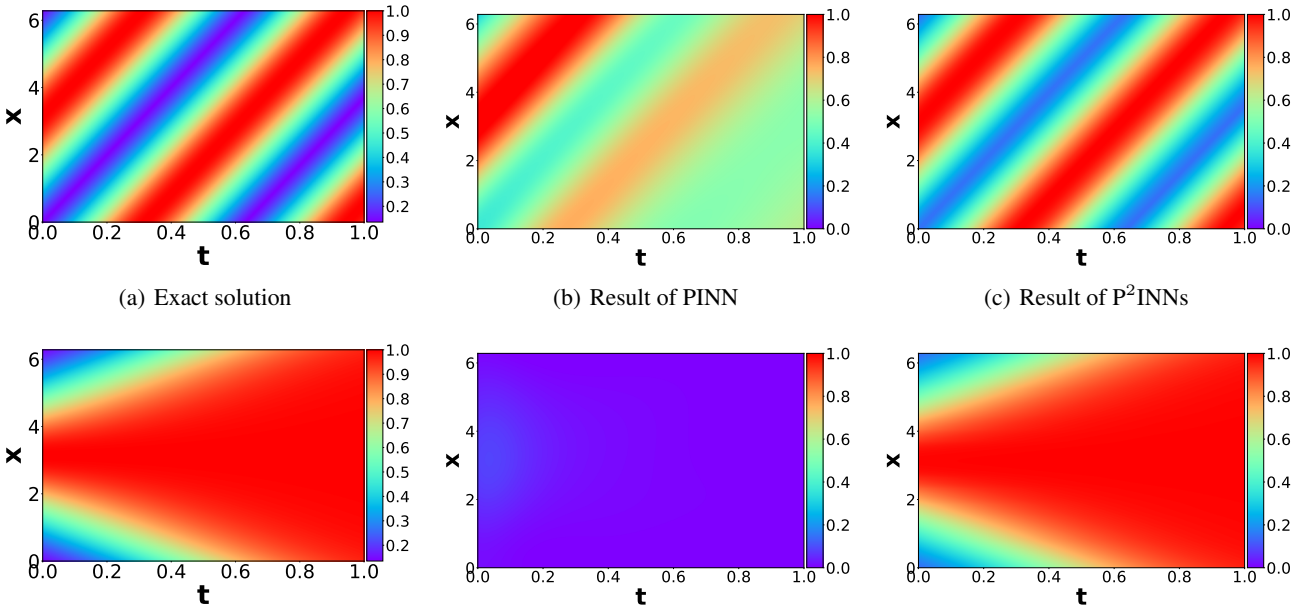
F. Fine-tuning P^{2}INNs

In general, our P^{2}INNs outperform other baselines in most of the tested equations. We can fine-tune the pre-trained model to further increase the accuracy and in this section, we show the efficacy of the fine-tuning step with intuitive visualizations.

F.1. Experiments with Gaussian Distribution as an Initial Condition

Experiments summarized in Table 2 use the initial condition of the Gaussian distribution $N(\pi, (\pi/2)^2)$. We fine-tune P^{2}INN from Table 2 on two PDEs: a convection equation with $\beta = 10$, and a reaction equation with $\rho = 5$. For the coefficient range used in pre-training, we select $\beta \in [1, 20]$ and $\rho \in [1, 10]$, respectively. We compare our fine-tuned model with vanilla PINN and results are summarized in Figure 10.

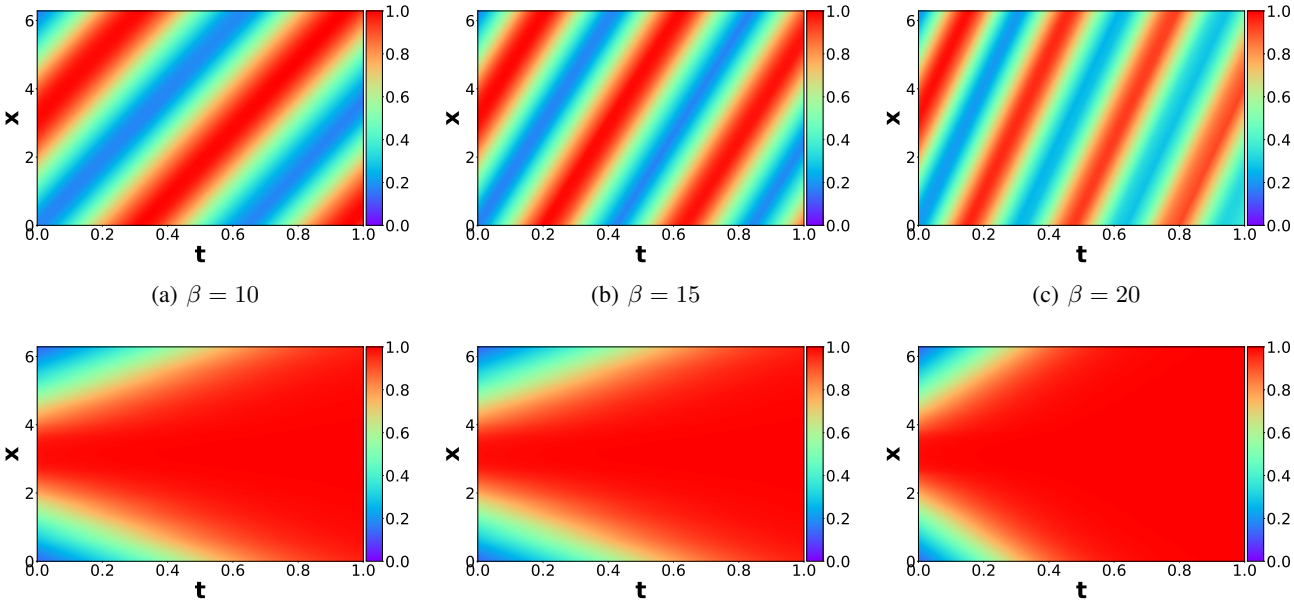
For the additional study, we show how the results of pre-trained P^{2}INNs are affected by varying the PDE parameters. Figures 11(a-c)/(d-f) are the results of convection/reaction equations. As shown in Figure 11, our P^{2}INNs effectively learn the differences among the various coefficient settings.



(a) Exact solution (b) Result of PINN (c) Result of P^{2}INNs

(d) Exact solution (e) Result of PINN (f) Result of P^{2}INNs

Figure 10. Experimental results of fine-tuning P^{2}INN. Convection equation of $\beta = 30$ (Figure 10. (a)-(c)). Reaction equation of $\rho = 5$ (Figure 10. (d)-(f)). Figures 10 (c) and (f) are the results after fine-tuning, and the results before fine-tuning can be checked in Figure 11.

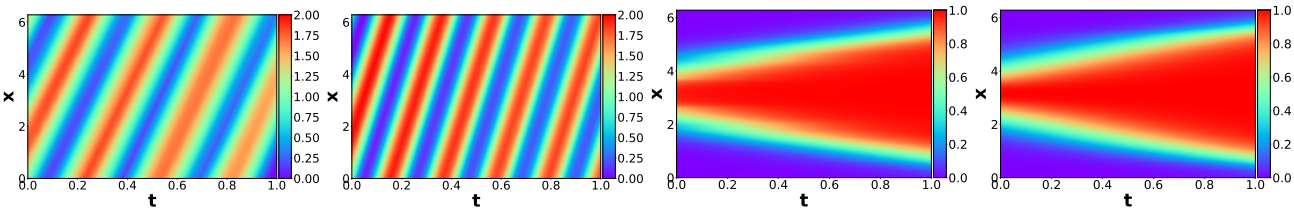


(a) $\beta = 10$ (b) $\beta = 15$ (c) $\beta = 20$

(d) $\rho = 5$ (e) $\rho = 7$ (f) $\rho = 9$

Figure 11. Results of P^{2}INN on convection equation and reaction equation without fine-tuning.

F.2. Failure Mode



(a) Before fine-tuning (b) After fine-tuning (c) Before fine-tuning (d) After fine-tuning

Figure 12. Experimental results of P^{2}INN in Section 4.2.2. Figures (a) and (b) are the results of convection equation $\beta = 30$, and Figures (c) and (d) are reaction equation $\rho = 5$.

Table 9. Results of P^{2}INNs for the failure mode. We use a convection equation with $1 + \sin(x)$ as an initial condition and a reaction equation with the Gaussian distribution $N(\pi, (\pi/4)^2)$.

Failure mode	**PINN**		**P^{2}INN**	
	Abs. err.	Rel. err.	Abs. err.	Rel. err.
$\beta = 30$	0.6132	0.5734	0.0910	0.0916
$\rho = 5$	0.5490	0.9844	0.0058	0.0173

Figure 7 is the result of P^{2}INN for the failure mode, and Figure 12 is a comparison between before and after fine-tuning on the results of P^{2}INN. Figures 12 (a) and (b) are the results on convection equation of $\beta = 30$, and Figures 12 (c) and (d) are the results on reaction equation of $\rho = 5$. As shown in Table 9, P^{2}INN significantly improves the performance compared to PINN.

F.3. Modulation for P^{2}INNs

Here, we provide the full results from Table 4 for six-types of CDR equations employing our SVD modulation.

Table 10. Experimental results of modulations with an initial condition of a Gaussian distribution $N(\pi, (\pi/2)^2)$

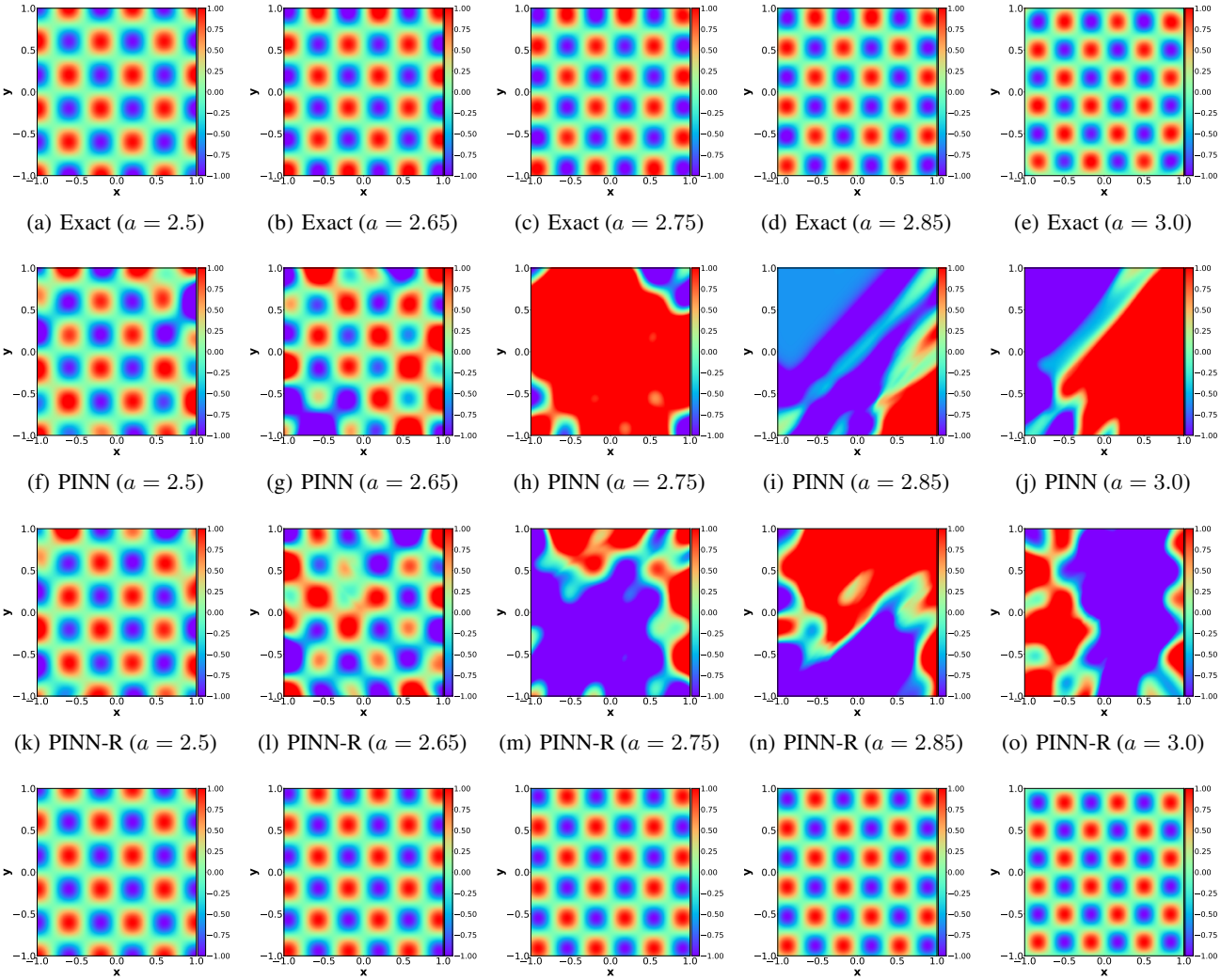
PDE type	**Metric**	**PINN**	**PINN-R**	**PINN-seq2seq**	**P^{2}INN**	**Modulation**		
						All	**Shift**	**SVD**
Convection	Abs. err.	0.0183	0.0222	0.1281	0.0174	0.1959	0.0139	**0.0138**
	Rel. err.	0.0327	0.1665	0.1987	0.0316	0.3319	0.0248	**0.0246**
Diffusion	Abs. err.	0.1335	0.1665	0.1987	0.0764	0.0726	0.0801	**0.0689**
	Rel. err.	0.2733	0.3462	0.4050	0.1694	**0.1392**	0.1785	0.1518
Reaction	Abs. err.	0.3341	0.3336	0.4714	0.0126	0.0713	0.0095	**0.0089**
	Rel. err.	0.3907	0.3907	0.5907	0.0229	0.1211	0.0198	**0.0184**
Conv.-Diff.	Abs. err.	0.0610	0.0654	0.0979	0.0443	0.0555	0.0452	**0.0422**
	Rel. err.	0.1175	0.1289	0.1950	0.0897	0.1074	0.0931	**0.0832**
Reac.-Diff.	Abs. err.	0.1900	0.1876	0.4201	0.0586	0.0581	0.0603	**0.0548**
	Rel. err.	0.2702	0.2777	0.5346	0.1015	**0.0886**	0.1043	0.0947
Conv.-Diff.-Reac.	Abs. err.	0.1663	0.0865	0.4943	0.0315	0.0463	0.0321	**0.0303**
	Rel. err.	0.2057	0.1415	0.6104	0.0508	0.0690	0.0521	**0.0486**

G. Experimental Results on 2D Helmholtz Equation

We undertake an evaluation by training our P^{2}INN model on a 2D Helmholtz equation and subsequently comparing its performance with that of PINNs. In the case of $a = \{2.50, 2.70, 2.80, 3.00\}$, performance is evaluated on the seen PDEs utilized for training, while for $a = \{2.65, 2.75, 2.85\}$, performance is assessed on the unseen PDEs not used during training phase. All test datasets consist of data that is not employed in the training, and the experimental results are reported in Table 11 and Figure 13.

Table 11. Comparision with PINN, PINN-R and P^{2}INN on 2D Helmholtz equations

Model	Metrics	$a = 2.50$	$a = 2.65$	$a = 2.70$	$a = 2.75$	$a = 2.80$	$a = 2.85$	$a = 3.00$
PINN	Abs. err.	0.1484	0.9077	1.9105	1.8942	1.5689	0.9077	2.4981
	Rel. err.	0.4817	2.0937	4.9264	4.7584	3.3739	2.0937	6.1532
PINN-R	Abs. err.	0.1107	0.2916	1.1590	1.4000	1.1095	1.5789	1.8800
	Rel. err.	0.3830	0.7239	2.8633	3.6641	2.6792	3.8059	4.7755
P^{2}INN	Abs. err.	**0.0240**	**0.0259**	**0.0257**	**0.0263**	**0.0321**	**0.0232**	**0.0315**
	Rel. err.	**0.0718**	**0.0767**	**0.0788**	**0.0840**	**0.0975**	**0.0642**	**0.0973**



(a) Exact ($a = 2.5$) (b) Exact ($a = 2.65$) (c) Exact ($a = 2.75$) (d) Exact ($a = 2.85$) (e) Exact ($a = 3.0$)

(f) PINN ($a = 2.5$) (g) PINN ($a = 2.65$) (h) PINN ($a = 2.75$) (i) PINN ($a = 2.85$) (j) PINN ($a = 3.0$)

(k) PINN-R ($a = 2.5$) (l) PINN-R ($a = 2.65$) (m) PINN-R ($a = 2.75$) (n) PINN-R ($a = 2.85$) (o) PINN-R ($a = 3.0$)

(p) P^{2}INN ($a = 2.5$) (q) P^{2}INN ($a = 2.65$) (r) P^{2}INN ($a = 2.75$) (s) P^{2}INN ($a = 2.85$) (t) P^{2}INN ($a = 3.0$)

Figure 13. [2D-Helmholtz equation] Exact solutions and results of PINN, PINN-R and P^{2}INN for various a

H. Architectural Details of PINN-P

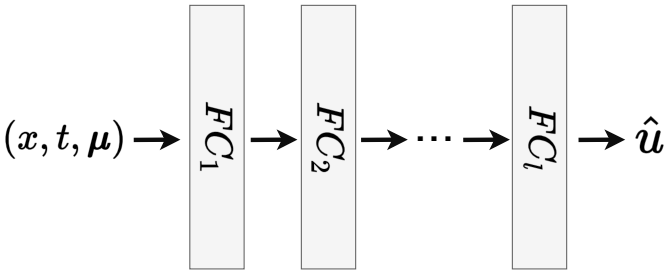


Figure 14. PINN-P architecture.

We propose PINN-P as an ablation model of our P^{2}INN. Unlike P^{2}INN, PINN-P does not have a separate encoder for PDE parameters, so that PDE parameters enter the model with coordinates. As shown in Figure 14, PINN-P consists of l-stacked fully-connected layers. For a fair comparison with P^{2}INNs, we set size of hidden vector to 150 and l to 6, making the model size similar to P^{2}INNs.

I. Reproducibility Statement

To benefit the community, the code will be posted online. The source code for our proposed method and the dataset used in this paper are attached.

J. Additional Statistical Analysis on CDR Equations

In this section, we provide additional statistical analysis on experiments of Table 2 with two other metrics, explained variance score (Exp. var.) and max error (Max. err.), and summarize the results in Table 12.

Table 12. The max error and explained variance score over all the equations with the initial condition of Gaussian distribution $N(\pi, (\pi/2)^2)$. We do not fine-tune P^{2}INN for each coefficient setting. However, our pre-trained models already outperform others in many cases.

	PDE type	Coefficient range	Metric	PINN	PINN-R	PINN-seq2seq	P^{2}INN
Class 1	Convection	1∼5	Max. err.	0.0513	0.0601	0.2751	0.0293
			Exp. var.	0.9948	0.9919	0.6070	0.9997
		1∼10	Max. err.	0.0593	0.1681	0.3744	0.0496
			Exp. var.	0.9952	0.8227	0.3009	0.9985
		1∼20	Max. err.	0.2431	0.3173	0.4190	0.1430
			Exp. var.	0.6076	0.4107	0.1500	0.9887
	Diffusion	1∼5	Max. err.	0.3989	0.5190	0.5784	0.4382
			Exp. var.	-0.1383	-1.0922	-1.5233	-0.4871
		1∼10	Max. err.	0.6194	0.7576	0.7638	-0.4871
			Exp. var.	-5.4421	-8.4565	-7.6938	-2.0589
		1∼20	Max. err.	1.3538	1.4417	0.7564	0.4516
			Exp. var.	-79.3751	-85.9107	-13.9281	-6.3671
	Reaction	1∼5	Max. err.	0.4053	0.4093	0.7653	0.0142
			Exp. var.	0.6232	0.6123	-0.1824	0.9999
		1∼10	Max. err.	0.7040	0.5116	0.8847	0.0376
			Exp. var.	0.3008	0.5036	-0.0936	0.9995
		1∼20	Max. err.	0.8529	0.7022	0.9429	0.0846
			Exp. var.	0.1492	0.2561	-0.0478	0.9977
Class 2	Conv.-Diff.	1∼5	Max. err.	0.1412	0.1601	0.3062	0.0399
			Exp. var.	0.7469	0.6008	0.2710	0.0892
		1∼10	Max. err.	0.2171	0.3023	0.3634	0.3169
			Exp. var.	-0.4752	-1.2314	-0.3168	0.6039
		1∼20	Max. err.	0.5396	0.5468	0.3606	0.4549
			Exp. var.	-19.0867	-17.8158	-0.4771	0.0253
	Reac.-Diff.	1∼5	Max. err.	0.4901	0.5272	0.7646	0.4335
			Exp. var.	0.1600	-0.2223	-0.3109	0.5052
		1∼10	Max. err.	0.8332	0.7536	0.8982	0.7501
			Exp. var.	-1.0339	-1.0530	-0.6734	-0.4028
		1∼20	Max. err.	0.9637	0.8345	0.9445	1.0133
			Exp. var.	-1.8660	-1.4076	-0.5779	-3.4065
Class 3	Conv.-Diff.-Reac.	1∼5	Max. err.	0.2694	0.3267	0.6831	0.2307
			Exp. var.	0.4267	0.5533	-0.2518	0.9333
		1∼10	Max. err.	0.6367	0.5262	0.8418	0.5249
			Exp. var.	0.1239	0.3869	-0.1386	0.7111
		1∼20	Max. err.	0.8192	0.6859	0.9213	0.7230
			Exp. var.	0.0587	0.2529	-0.0720	0.6803

K. Experiments on Another Initial Condition

In this section, we present the experimental results with the initial condition of $1+\sin(x)$, with other conditions remain unchanged. In Table 13, we summarize the results in terms of L_2 absolute and relative errors, and in Table 14, we use max error and explained variance score for evaluation. Even with the initial condition of $1+\sin(x)$, our P^{2}INNs show adequate performance, especially in a wide coefficient range, i.e., 1$\sim$20. On top of that, particularly in the case of reaction and conv.-diff.-reac. equations, P^{2}INNs outperform the baselines by a huge gap.

Table 13. The relative and absolute L_2 errors over all the equations with the initial condition of $1+\sin(x)$. We do not fine-tune P^{2}INN for each coefficient setting. However, our pre-trained models already outperform others in many cases.

	PDE type	Coefficient range	Metric	PINN	PINN-R	PINN-seq2seq	P^{2}INN
Class 1	Convection	1$\sim$5	Abs. err.	0.0135	0.0073	0.2213	0.0045
			Rel. err.	0.0147	0.0076	0.2159	0.0044
		1$\sim$10	Abs. err.	0.0117	0.0233	0.4101	0.0095
			Rel. err.	0.0127	0.0244	0.3821	0.0092
		1$\sim$20	Abs. err.	0.1283	0.2558	0.5355	0.0826
			Rel. err.	0.1295	0.2503	0.5092	0.0827
	Diffusion	1$\sim$5	Abs. err.	0.0496	0.0688	0.2682	0.4027
			Rel. err.	0.0714	0.0956	0.3048	0.4608
		1$\sim$10	Abs. err.	0.0835	0.1106	0.3047	0.4863
			Rel. err.	0.1078	0.1416	0.3356	0.5487
		1$\sim$20	Abs. err.	0.1206	0.2045	0.3015	0.5254
			Rel. err.	0.1500	0.2333	0.3308	0.5963
	Reaction	1$\sim$5	Abs. err.	0.2349	0.2396	0.5100	0.0254
			Rel. err.	0.3241	0.3316	0.6117	0.0736
		1$\sim$10	Abs. err.	0.5688	0.3418	0.7054	0.0617
			Rel. err.	0.6561	0.4494	0.7994	0.1199
		1$\sim$20	Abs. err.	0.7615	0.4290	0.8294	0.1487
			Rel. err.	0.8274	0.5268	0.8987	0.2901
Class 2	Conv.-Diff.	1$\sim$5	Abs. err.	0.0213	0.0182	0.2382	0.1997
			Rel. err.	0.0261	0.0233	0.2605	0.2296
		1$\sim$10	Abs. err.	0.0294	0.0345	0.2316	0.1297
			Rel. err.	0.0350	0.0410	0.2574	0.1514
		1$\sim$20	Abs. err.	0.0824	0.0960	0.2504	0.1173
			Rel. err.	0.0928	0.1094	0.2741	0.1339
	Reac.-Diff.	1$\sim$5	Abs. err.	0.0386	0.0368	0.3038	0.1150
			Rel. err.	0.0632	0.0581	0.3562	0.1566
		1$\sim$10	Abs. err.	0.2305	0.1632	0.5427	0.0816
			Rel. err.	0.2681	0.2054	0.5807	0.1190
		1$\sim$20	Abs. err.	0.5454	0.3107	0.6892	0.0413
			Rel. err.	0.5710	0.3497	0.7131	0.0727
Class 3	Conv.-Diff.-Reac.	1$\sim$5	Abs. err.	0.0593	0.0932	0.3626	0.0551
			Rel. err.	0.0899	0.1465	0.4554	0.0735
		1$\sim$10	Abs. err.	0.4953	0.3407	0.6500	0.0337
			Rel. err.	0.5341	0.4006	0.7196	0.0538
		1$\sim$20	Abs. err.	0.7329	0.4283	0.8101	0.0614
			Rel. err.	0.7663	0.4911	0.8588	0.1240

Table 14. The max error and explained variance score over all the equations with the initial condition of $1+\sin(x)$. We do not fine-tune P^2INN for each coefficient setting. However, our pre-trained models already outperform others in many cases.

	PDE type	Coefficient range	Metric	PINN	PINN-R	PINN-seq2seq	P^2INN
Class 1	Convection	1∼5	Max. err.	0.0531	0.0316	0.6235	0.0141
			Exp. var.	0.9993	0.9998	0.8369	1.0000
		1∼10	Max. err.	0.0479	0.0919	0.8839	0.0358
			Exp. var.	0.9994	0.9947	0.4700	0.9998
		1∼20	Max. err.	0.3495	0.5931	1.1652	0.2295
			Exp. var.	0.8662	0.6638	0.2362	0.9681
	Diffusion	1∼5	Max. err.	0.2706	0.3166	0.6683	0.9959
			Exp. var.	0.9322	0.8433	0.3848	-1.3721
		1∼10	Max. err.	0.3190	0.3693	0.6645	1.0063
			Exp. var.	0.7238	0.5667	0.0207	-4.0842
		1∼20	Max. err.	0.4056	0.4829	0.6612	0.9708
			Exp. var.	0.0209	-0.4390	-0.5409	-21.5206
	Reaction	1∼5	Max. err.	0.8754	0.8815	1.1852	0.8770
			Exp. var.	0.6996	0.6760	0.1010	0.9692
		1∼10	Max. err.	1.2884	1.0865	1.4784	1.0001
			Exp. var.	0.3725	0.3932	0.0618	0.8961
		1∼20	Max. err.	1.4931	1.2158	1.5880	1.0000
			Exp. var.	0.1881	0.2003	0.0331	0.0000
Class 2	Conv.-Diff.	1∼5	Max. err.	0.0811	0.0702	0.6079	0.6515
			Exp. var.	0.9914	0.9888	0.5136	0.4635
		1∼10	Max. err.	0.1002	0.1154	0.6613	0.5590
			Exp. var.	0.9641	0.9532	0.2876	0.6620
		1∼20	Max. err.	0.2583	0.3162	0.7221	0.3527
			Exp. var.	0.7337	0.6718	-0.1857	0.1614
	Reac.-Diff.	1∼5	Max. err.	0.2870	0.2659	0.9199	0.5404
			Exp. var.	0.8536	0.8962	0.2670	0.5430
		1∼10	Max. err.	0.7232	0.6471	1.2166	0.5853
			Exp. var.	0.2866	0.1213	0.1293	0.4415
		1∼20	Max. err.	1.1176	0.9268	1.3338	0.3204
			Exp. var.	0.1430	-0.0407	-0.0576	0.3938
Class 3	Conv.-Diff.-Reac.	1∼5	Max. err.	0.3264	0.5064	1.1676	0.3694
			Exp. var.	0.9039	0.6877	-0.0153	0.8884
		1∼10	Max. err.	1.0318	0.9348	1.4829	0.4511
			Exp. var.	0.4165	0.3708	-0.0199	0.8822
		1∼20	Max. err.	1.3732	1.1398	1.5986	0.9006
			Exp. var.	0.2066	0.1825	-0.0091	0.0134

L. Standard Deviation of the Evaluation Metrics from Table 2

In Table 15, we report the standard deviation of evaluation metrics, Abs. err. and Rel. err., from Table 2. We conduct the experiments in Table 2 with three different random seeds.

Table 15. Standard deviation of evaluation metrics from Table 2.

	PDE type	Coefficient range	Metric	PINN	PINN-R	PINN-seq2seq	P^2INN
Class 1	Convection	1∼5	Abs. err.	0.0012	0.0058	0.0130	0.0005
			Rel. err.	0.0022	0.0095	0.0221	0.0008
		1∼10	Abs. err.	0.0022	0.0177	0.0065	0.0039
			Rel. err.	0.0051	0.0305	0.0109	0.0072
		1∼20	Abs. err.	0.0021	0.0066	0.0032	0.0064
			Rel. err.	0.0012	0.0125	0.0055	0.0100
	Diffusion	1∼5	Abs. err.	0.0023	0.0156	0.0066	0.0010
			Rel. err.	0.0068	0.0321	0.0142	0.0013
		1∼10	Abs. err.	0.0178	0.0282	0.0116	0.0001
			Rel. err.	0.0326	0.0583	0.0225	0.0001
		1∼20	Abs. err.	0.0093	0.0115	0.0183	0.0001
			Rel. err.	0.0205	0.0244	0.0358	0.0002
	Reaction	1∼5	Abs. err.	0.0018	0.1605	0.0159	0.0025
			Rel. err.	0.0023	0.1853	0.0087	0.0030
		1∼10	Abs. err.	0.0011	0.0915	0.0080	0.0017
			Rel. err.	0.0014	0.1009	0.0044	0.0019
		1∼20	Abs. err.	0.0005	0.0199	0.0040	0.0649
			Rel. err.	0.0007	0.0258	0.0022	0.1445
Class 2	Conv.-Diff.	1∼5	Abs. err.	0.0018	0.0021	0.0023	0.0097
			Rel. err.	0.0026	0.0045	0.0034	0.0176
		1∼10	Abs. err.	0.0033	0.0105	0.0023	0.0035
			Rel. err.	0.0064	0.0194	0.0043	0.0037
		1∼20	Abs. err.	0.0012	0.0066	0.0007	0.0055
			Rel. err.	0.0018	0.0117	0.0013	0.0146
	Reac.-Diff.	1∼5	Abs. err.	0.0100	0.0322	0.0126	0.0245
			Rel. err.	0.0135	0.0352	0.0105	0.0420
		1∼10	Abs. err.	0.0045	0.0174	0.0043	0.0507
			Rel. err.	0.0025	0.0182	0.0046	0.0745
		1∼20	Abs. err.	0.0007	0.0070	0.0018	0.2010
			Rel. err.	0.0011	0.0070	0.0030	0.2114
Class 3	Conv.-Diff.-Reac.	1∼5	Abs. err.	0.0072	0.0015	0.0042	0.0054
			Rel. err.	0.0071	0.0024	0.0021	0.0062
		1∼10	Abs. err.	0.0102	0.0022	0.0055	0.0021
			Rel. err.	0.0118	0.0047	0.0062	0.0037
		1∼20	Abs. err.	0.0034	0.0082	0.0008	0.0024
			Rel. err.	0.0047	0.0077	0.0011	0.0041

M. Ablation Studies on PINN-P and LargePINN

Table 16. Number of model parameters.

	PINN	PINN-R	PINN-seq2seq	LargePINN	PINN-P	P^2INN
#params	10,401	10,401	10,401	82,941	91,651	76,851

Table 17. The relative and absolute L_2 errors over all the equations. Our P^2INNs surpass LargePINN and PINN-P in all but one cases, even without fine-tuning.

	PDE type	Metric	PINN	LargePINN	PINN-P	P2INN
Class 1	Convection	Abs. err.	0.1140	0.1191	0.0209	**0.0198**
		Rel. err.	0.1978	0.2084	**0.0410**	0.0464
	Diffusion	Abs. err.	0.6782	0.5868	0.3800	**0.1916**
		Rel. err.	1.2825	1.0994	0.7912	**0.3745**
	Reaction	Abs. err.	0.7902	0.7910	0.8975	**0.0042**
		Rel. err.	0.8460	0.8469	0.9908	**0.0092**
Class 2	Conv.-Diff.	Abs. err.	0.2735	0.1626	0.1253	**0.0622**
		Rel. err.	0.5106	0.3189	0.3009	**0.1495**
	Reac.-Diff.	Abs. err.	0.7167	0.7399	0.1756	**0.0898**
		Rel. err.	0.7998	0.8186	0.2632	**0.1411**
Class 3	Conv.-Diff.-Reac.	Abs. err.	0.7450	0.7415	0.8590	**0.0353**
		Rel. err.	0.7960	0.7915	0.9532	**0.0812**

For more comprehensive evaluation, we conduct additional ablation studies following the experimental settings of Table 2 with the coefficient range of $1 \sim 20$ using PINN-P (cf. Appendix B.2) and LargePINN, which is PINN with bigger network size. As shown in Table 16, since the model size of our proposed P^2INN is larger than original PINN, we conduct experiments using a LargePINN model. The LargePINN has the same MLP architecture as the original PINN but with increased hidden dimensions, resulting in a model size of 82,941.

The experimental results of LargePINN, PINN-P, and P^2INN are summarized in Table 17. In all scenarios, as indicated by Table 17, the LargePINN model consistently performs inferiorly compared to P^2INNs, and P^2INNs outperforms PINN-P in all cases except one. That is, while the baselines struggles when learning the equations encompassing wide coefficient ranges, i.e., $1 \sim 20$. For instance, considering Conv.-Reac.-Diff. equation, the L_2 absolute error exhibited by P^2INN is 0.0353 whereas LargePINN and PINN-P have errors of 0.7415 and 0.8590, respectively. Note that this collective outcome underscores that P^2INN's separation of PDE parameters and spatial/temporal coordinates during the learning process significantly enhances both generalization capabilities and scalability.

N. Ablation Studies on Varying Data Points

We conduct a more challenging experiment by extremely reducing the training data points. Assuming the training of reaction equations for $\rho \in [1, 10]$, a total of 10,000 collocation points are utilized during the PINN's training process - with 10 separate models processing 1,000 collocation points each. On the other hand, for P^2INNs, 10 equations are jointly learned by a single model, also with a total of 10,000 collocation points. We name those two models PINN(10,000) and P^2INN(10,000) respectively, and compare these two models with new ablation model, P^2INN(1,000). To be specific, for P^2INN(1,000), a single model learned from a subset of 100 data points taken from each of the 10 equations, resulting in a total of 1,000 collocation points. The outcome of this experiment is summarized in the table presented below.

Table 18. Experimental results on Reaction equation with varying ρ

	$\rho = 1$	$\rho = 2$	$\rho = 3$	$\rho = 4$	$\rho = 5$	$\rho = 6$	$\rho = 7$	$\rho = 8$	$\rho = 9$	$\rho = 10$
PINN(10,000)	0.0045	0.0041	0.0047	0.7974	0.8598	0.8820	0.9021	0.9159	0.9263	0.9350
P^2INN(1,000)	0.0078	0.0049	0.0036	0.0036	0.0029	0.0020	0.0020	0.0014	0.0014	0.0016
P^2INN(10,000)	0.0034	0.0029	0.0017	0.0022	0.0014	0.0012	0.0008	0.0008	0.0008	0.0009

According to the Table 18, P^2INNs with only 1,000 data points, i.e., P^2INN(1,000), achieves successful outcomes for the failure mode of PINN involving reaction equations with values over 4, outperforming PINN by a big margin. Notably, its results are comparable to P^2INNs with the original setting, i.e., P^2INN(10,000).