Mask-PINNs: Mitigating Internal Covariate Shift in Physics-Informed Neural Networks

Feilong Jiang[a] Xiaonan Hou[a] Jianqiao Ye[a] Min Xia[b]*
[a] Department of Engineering, Lancaster University, LA1 4YW Lancaster, U.K.
[b] Department of Mechanical and Materials Engineering, University of Western Ontario, London, Ontario, Canada
min.xia@uwo.ca

Abstract

Physics-Informed Neural Networks (PINNs) have emerged as a powerful framework for solving partial differential equations (PDEs) by embedding physical laws directly into the loss function. However, as a fundamental optimization issue, internal covariate shift (ICS) hinders the stable and effective training of PINNs by disrupting feature distributions and limiting model expressiveness. Conventional remedies for ICS—such as Batch Normalization and Layer Normalization—aim to stabilize feature distributions through statistical regularization. However, PINNs require deterministic coordinate-to-solution mappings for enforcing physical constraints, making such strategies fundamentally misaligned with their formulation. To address this issue, we propose Mask-PINNs, which introduce a smooth, learnable mask to adaptively regulate internal features without altering the pointwise physics-based formulation. We provide a theoretical analysis showing that the mask suppresses the expansion of feature representations through a carefully designed modulation mechanism. Empirically, we validate the method on multiple PDE benchmarks across diverse activation functions. Our results show consistent improvements in prediction accuracy, convergence stability, and robustness. Furthermore, we demonstrate that Mask-PINNs enable the effective use of wider networks, overcoming a key limitation in existing PINN frameworks. The codes of the experiments can be found on https://github.com/flongjiang/Mask-PINNs.

Key words: Deep learning, Physics-informed neural networks, Partial differential equations, Scientific machine learning

1 Introduction

As a bridging approach between classical scientific computing and machine learning, Physics-Informed Neural Networks (PINNs) have emerged as a powerful framework for solving partial differential equations (PDEs) [1]. In PINNs, neural networks approximate the solution of a PDE, with the governing equations enforced through the loss function. This approach enables data-efficient learning and provides a flexible framework for incorporating prior physical knowledge [2-5]. Due to their strong interpretability and ease of implementation, PINNs have found successful applications in diverse fields such as fluid mechanics [6], heat transfer problems [7], solid mechanics [8], geoengineering [9], chemical engineering [10], finance [11], and climate modelling [12].

Despite their growing popularity, training PINNs effectively remains challenging. Numerous studies have been conducted to improve their performance. Most of these methods focus on training pathologies of PINNs. To mitigate activation sensitivity, several methods have been proposed to learn or adapt activation functions during training thereby improve efficiency and stability of training [13], [14]. To address the performance degradation in deeper PINN architectures, methods like PirateNets, HyResPINNs, and Deeper-PINNs are proposed to

improve the scalability of PINNs [15-17]. To overcome vanishing or exploding gradients caused by poor initial choices, theoretically informed initialization approaches are proposed to ensure stable training and robust convergence in PINNs [18], [19]. Another well-known challenge comes from the composite loss function. Different loss components (e.g., PDE residuals, boundary conditions, data terms) often compete, leading to gradient conflicts, which hinder convergence. Numerous methods have been proposed to alleviate such imbalanced optimization [20-25]. Moreover, as demonstrated in [26-30], adopting more effective network architectures can strengthen the representational capacity of PINNs, offering a promising route to improve the performance of PINNs.

One critical yet underexplored challenge in PINNs is internal covariate shift (ICS)—the instability of feature distributions during training—which undermines both the learning dynamics and the expressiveness of PINNs [31]. While normalization techniques such as Batch Normalization (BN) and Layer Normalization (LN) are widely used in deep learning to mitigate ICS, their normalization mechanisms are not naturally aligned with the functional structure of PINNs. Specifically, BN introduces cross-sample dependence through batch statistics, breaking the pointwise enforcement of PDE constraints. LN preserves pointwise determinism but enforces identical sample-wise mean and variance, restricting coordinate-dependent feature variation and thereby limiting expressiveness.

This raises a fundamental question: how can feature distributions be stabilized in PINNs without disrupting their pointwise, coordinate-dependent functional structure?

In this paper, we introduce Mask-PINNs, a novel PINN architecture designed to address this issue. Our contributions are summarized as follows:

- We propose Mask-PINNs, which introduce a learnable nonlinear mask to regulate feature distributions in a pointwise, input-conditioned manner. This preserves the pointwise mapping while maintaining coordinate-dependent feature variation throughout the network.
- We demonstrate that the proposed mask mechanism effectively suppresses feature drift, resulting in more centered and stable feature distributions throughout training, thereby improving optimization stability and expressiveness of PINNs.
- Extensive experiments across diverse PDE benchmarks and activation functions show that Mask-PINNs achieve consistent and significant improvements in prediction accuracy and training stability.
- We show that Mask-PINNs enable effective training of wider neural networks, overcoming a critical limitation in conventional PINNs where increasing network width typically leads to performance degradation. This makes wider networks viable in practice, expanding the design space for PINNs.

The remainder of the paper is organized as follows: Section 2 reviews relevant prior work and discusses key limitations; Section 3 presents Mask-PINNs and shows theoretically how the mask improves stability and signal propagation; Section 4 presents comprehensive experimental validation; Section 5 provides further theoretical and practical insights, including NTK-based analysis, advanced optimization studies, and sensitivity analysis of the mask parameter; and finally, Section 6 summarizes the findings and discusses potential directions for future research.

2 Problem Statement

This section introduces the working principle of PINNs. We also discuss traditional normalization techniques' limitations in PINNs.

2.1 Physics-Informed Neural Networks

Given the following partial differential equation (PDE):

$$\mathsf{N}\left[\mathbf{u}(\mathbf{x})\right] = f(\mathbf{x}),\ \mathbf{x} \in \Omega, \tag{1}$$

$$\mathsf{B}\left[\mathbf{u}(\mathbf{x})\right] = g(\mathbf{x}),\ \ \mathbf{x} \in \partial\Omega, \tag{2}$$

where $\mathbf{u}(\mathbf{x})$ indicates the solution, $\mathsf{N}[\cdot]$ represents a differential operator, $\mathsf{B}[\cdot]$ indicates boundary operator.

The task of PINNs is approximating the solution of the given PDE by a deep neural network $\mathbf{u}_\theta(\mathbf{x})$, where θ denotes all trainable parameters. For PINNs, $\mathbf{u}_\theta(\mathbf{x})$ is constrained by the following loss function:

$$\mathsf{L}(\theta) = \lambda_b \mathsf{L}_b(\theta) + \lambda_r \mathsf{L}_r(\theta), \tag{3}$$

where

$$\mathsf{L}_b(\theta) = \frac{1}{N_b}\sum_{i=1}^{N_c}\left|\mathsf{B}\left[\mathbf{u}_\theta\left(\mathbf{x}_b^i\right)\right] - g\left(\mathbf{x}_b^i\right)\right|^2, \tag{4}$$

$$\mathsf{L}_r(\theta) = \frac{1}{N_r}\sum_{i=1}^{N_r}\left|\mathsf{N}\left[\mathbf{u}_\theta\left(\mathbf{x}_r^i\right)\right] - f\left(\mathbf{x}_r^i\right)\right|^2, \tag{5}$$

λ is responsible for balancing the weight of different loss terms. $\mathbf{x}_r$ are collocation points in the domain. $\mathbf{x}_b$ are boundary points.

2.2 Why Standard Normalization Fails in PINNs

During training, the feature distributions at each layer can drift in mean, variance and overall shape, a phenomenon referred to as internal covariate shift. For bounded activation functions such as Tanh, such drift can drive outputs toward their saturation limits, leading to vanishing gradients and reduced expressiveness [32]. Although unbounded activations such as GELU, SiLU, or SoftPlus can alleviate this effect, they remain susceptible to ICS and often behave quasi-linearly when inputs are far from zero, limiting the network's ability to capture complex nonlinear patterns [33], [34].

In conventional deep learning, ICS is typically addressed by normalization techniques such as BN or LN [31], [35], [36]. However, in the context of PINNs, these traditional methods can distort the underlying physical relationships between inputs and outputs, due to their reliance on batch-level or feature-level statistics.

To clarify this issue, we briefly revisit the formulations of BN and LN:

The output of a BN layer can be written as:

$$\hat{h}(\mathbf{x}_i; \mathsf{S}) = \gamma\left(\frac{h(\mathbf{x}_i) - \mu_B}{\sigma_B + \varepsilon}\right) + \beta, \tag{6}$$

where $h(\mathbf{x}_i)$ is the pre-activation feature map for input $\mathbf{x}_i$, S is the current batch containing $\mathbf{x}_i$, ε is for numerical stability in case the denominator becomes zero by chance. γ and β are learnable scale and shift.

$$\mu_B = \frac{1}{N}\sum\nolimits_{x_j \in \mathsf{S}} h(\mathbf{x}_j), \tag{7}$$

$$\sigma_B^2 = \frac{1}{N}\sum\nolimits_{\mathbf{x}_j \in \mathsf{S}}\left(h(\mathbf{x}_j) - \mu_B\right)^2, \tag{8}$$

where N denotes the batch size.

As μ_B and σ_B are computed using all samples in the batch, the normalized output $\hat{h}(\mathbf{x}_i;\mathsf{S})$ is no longer solely a function of $\mathbf{x}_i$, but also depends on the batch-level statistics determined by the sample set S. Based on these, the following proposition is presented

Proposition 1: For a fixed parameter set θ, a PINN equipped with BN does not represent a pointwise solution function. Instead, the network output $u_\theta(\mathbf{x}_i)$ depends on the composition of the batch S. This is incompatible with the PINN formulation that requires a deterministic solution at each collocation point.

The proof is detailed in Appendix A

Unlike BN, LN operates independently on each input sample. Given an input $\mathbf{x}_i$, and $h(\mathbf{x}_i) \in \mathbb{R}^d$ be the pre-activation output of an FC layer. LN produces:

$$\hat{h}(\mathbf{x}_i) = \gamma \frac{h(\mathbf{x}_i) - \mu_i}{\sigma_i + \varepsilon} + \beta, \tag{9}$$

where $\mu_i = \frac{1}{d}\sum_{j=1}^{d} h_j(\mathbf{x}_j)$, $\sigma_i^2 = \frac{1}{d}\sum_{j=1}^{d}\left(h_j(\mathbf{x}_i) - \mu_i\right)^2$.

The normalized representation satisfies:

$$\frac{1}{d}\sum_{j=1}^{d}\hat{h}_j = 0, \quad \frac{1}{d}\sum_{j=1}^{d}\hat{h}_j^2 = 1 \tag{10}$$

Based on these, we characterize the effect of LN as follows:

Remark 1: LN enforces identical sample-wise mean and variance at every input. In PINNs, scale and shift variations are directly induced by the input coordinates and reflect how intermediate representations vary with the input. By suppressing these variations, LN impairs the expressive capacity of the network by restricting how such input-dependent signals propagate through the model. This effect is consistently reflected in the following experiments.

In the following, we empirically confirm these issues with an experiment on the heat conduction problem:

$$\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} = 0, \ (x,t) \in [0,1]\times[0,1], \tag{11}$$

$$u(x,0) = \sin(\pi x), \ x \in [0,1], \tag{12}$$

$$u(0,t) = u(1,t) = 0, \ t \in [0,1], \tag{13}$$

the analytical solution for this problem is:

$$u(x,t) = \sin(\pi x)\cdot\exp(-t\pi^2). \tag{14}$$

Experiments are conducted on vanilla PINN, BN-based PINN, and LN-based PINN. For BN and LN, both pre-activation and post-activation placements are evaluated, as the normalization position can significantly influence training behavior and gradient propagation. All the outcomes are averaged from 5 independent trials.

As shown in Fig. 1, BN-based PINN and LN-based PINN show worse results than vanilla PINN. For BN-PINN, even though the loss curve decreases during the training, the relative L^2 error curve stays almost flat. This indicates that for PINN with BN, the model is not trained with the given physical constraint. Although LN-PINN shows a consistent decrease in both loss and relative L^2 error, the accuracy is lower than that of vanilla method. This indicates that LN impairs the model's ability to capture physical details.

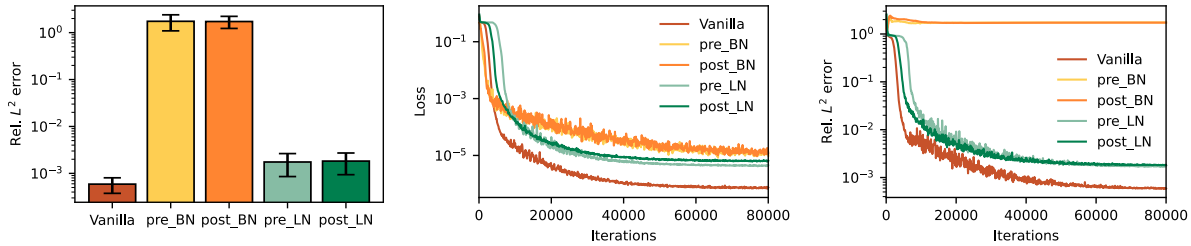


Figure 1. Performance comparison of vanilla PINN, BN-PINN, and LN-PINN. (a) Mean relative L^2 error. (b) Averaged loss curve. (c) Averaged relative L^2 error curve.

In summary, BN breaks the pointwise formulation by making outputs batch-conditioned, resulting in an invalid mapping. LN suppresses input-dependent variations, thereby restricting the expressive capability of PINNs.

3 Methodology

3.1 Motivation and Design Principles

Instead of forcing the feature representations to conform to fixed statistical distributions, as BN and LN do, we propose an input-conditioned modulation mechanism. Fig. 2 shows the schematic illustration of the proposed method.

The input coordinates are first mapped into a high-dimensional feature space through an embedding layer (e.g., a fully connected layer or random Fourier features), followed by a stack of N residual blocks. Within each block, a tailored mask function is applied to modulate intermediate representations, stabilizing feature distributions across hidden layers.

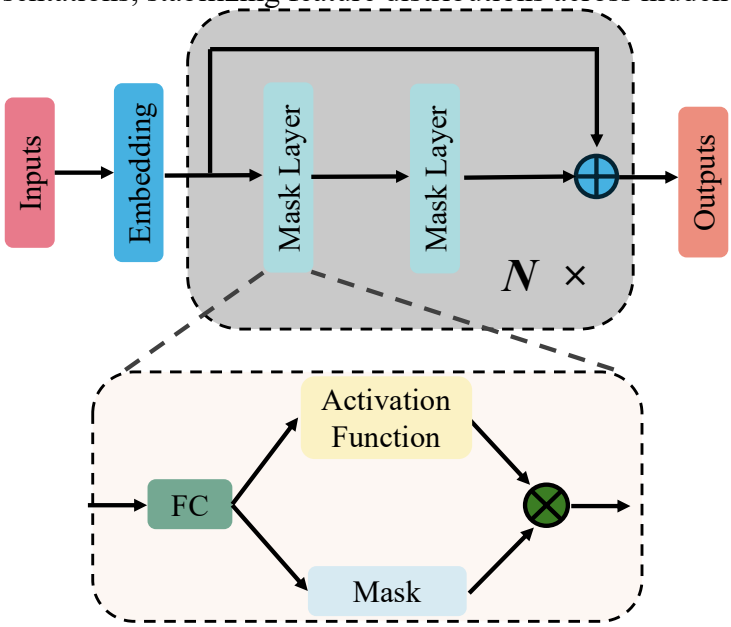


Figure 2. Illustration of the architecture of the Mask-PINNs.

Given an input vector $\mathbf{x}$, a mask function can be expressed as:

$$F(\mathbf{x}) = 1 - \exp\left[-(\boldsymbol{\alpha} \odot \mathbf{x})^2\right] \tag{15}$$

where $\boldsymbol{\alpha}$ is a learnable vector that controls the sharpness of the mask function. $\boldsymbol{\alpha}$ is initialized as a positive vector. Fig. 3 shows the visualization of the proposed mask function. The proposed mask function, resembling an inverted Gaussian centered at zero, is smooth and differentiable, ensuring the network output remains differentiable as required by PINNs. Because the mask is applied pointwise as a deterministic function of the pre-activation, the

transformation remains coordinate-conditioned, and variations induced by distinct coordinates continue to be encoded in the propagated representations.

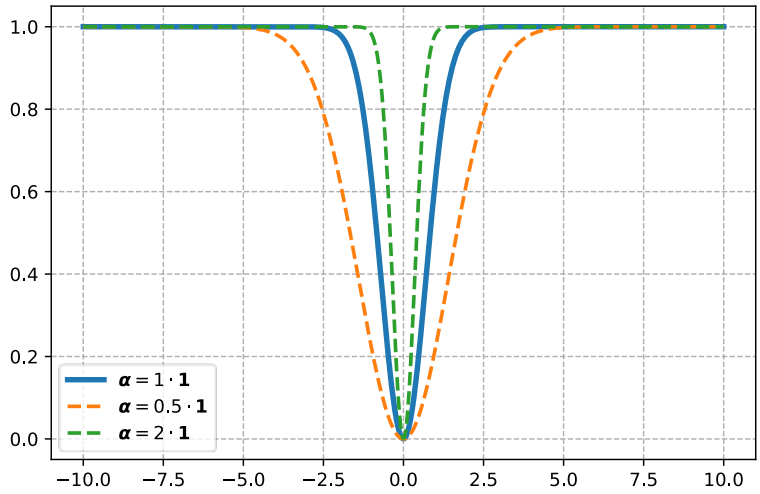


Figure 3. Mask function with different $\boldsymbol{\alpha}$.

With the mask inserted, the feed-forward process of layer l can be expressed as:

$$z^{(l)} = w^{(l)}H + b^{(l)}, \tag{16}$$

$$H^{(l)} = F^{(l)}(z^{(l)}) \odot \sigma(z^{(l)}), \tag{17}$$

where z represents the pre-activation, l is the number of the layer, H denotes the output of a layer, w represents weights and b represents biases. Since the mask satisfies $F(z) \leq 1$, the multiplicative form in Eq. (17) introduces a suppressive effect on the features.

3.2 Theoretical Analysis

Remark 2. For any scalar input $u \in \mathbb{R}$, since $F(u) \leq 1$, we have:

$$|H(u)| \leq |\sigma(u)|. \tag{18}$$

Thus, the mask contracts activation magnitudes globally, suppressing extreme values and yielding a more compact feature distribution.

Proposition 2. For any $r > 0$, quantify local sensitivity by the minimal Lipschitz constant on $[-r, r]$:

$$L_H(r) := \sup_{x \neq y,\ |x|,|y| \leq r} \frac{|H(x) - H(y)|}{|x - y|},\ L_\sigma(r) := \sup_{x \neq y,\ |x|,|y| \leq r} \frac{|\sigma(x) - \sigma(y)|}{|x - y|}. \tag{19}$$

For differentiable scalar functions on an interval, the mean value theorem implies

$$L_H(r) := \sup_{|u| \leq r} |H'(u)|,\ L_\sigma(r) := \sup_{|u| \leq r} |\sigma'(u)|. \tag{20}$$

As $r \to 0$

$$\frac{L_H(r)}{L_\sigma(r)} = \begin{cases} O(r) \text{ if } \sigma(0) \neq 0, \\ O(r^2) \text{ if } \sigma(0) = 0. \end{cases} \tag{21}$$

Proof. The proof is detailed in Appendix B

Remark 3. Under Xavier initialization, pre-activations typically concentrate near zero. In this regime, Proposition 2 shows that the proposed mask contracts the local Lipschitz constant by at least one order compared to the vanilla method. This contraction dampens the effect of perturbations, thereby keeping the output zero centered and stable around initialization.

In summary, the proposed mask mechanism globally contracts activation magnitudes, suppressing extreme values and keeping feature distributions compact, while it locally

stabilizes behavior near initialization by reducing sensitivity to perturbations. Collectively, these properties suppress the distributional drift thereby mitigating internal covariate shift in PINNs.

3.3 Mask Placement

Notice that unlike normalization methods, which are typically applied before the activation function, our mask function is applied alongside it. Such design helps the mask function robustly scale down the feature values across different activation functions.

Suppose placing the mask function before the activation function, we have the output as:

$$H = \sigma(z \odot F(z)). \tag{22}$$

Although this may seem to be a more straightforward approach, the distribution of H depends heavily on the properties of the activation function, which can therefore reintroduce the ICS. For activations with $\sigma(0) \neq 0$ (e.g., SoftPlus), such design leads to biased outputs even for zero-centered inputs, causing a systematic shift in feature distributions across layers. In contrast, applying the mask alongside the activation function can effectively suppress the activation magnitude while alleviating the mean shift caused by activation functions.

To verify this claim, we conduct an experiment on the heat conduction problem introduced in Section 2.2 under the same configuration, with SoftPlus adopted as the activation function. The mask is applied either before the activation function or alongside it for comparison.

As shown in Fig. 4, applying the mask before the activation function fails to stabilize the feature distributions. In particular, the pre-activation statistics exhibit progressive dispersion as the network depth increases, indicating that internal covariate shift is not effectively mitigated. Consequently, the corresponding loss decreases slowly and quickly plateaus at a relatively high level. In contrast, when the mask is applied alongside the activation function, feature distributions remain well-controlled across layers, demonstrating effective suppression of activation-induced mean shift.

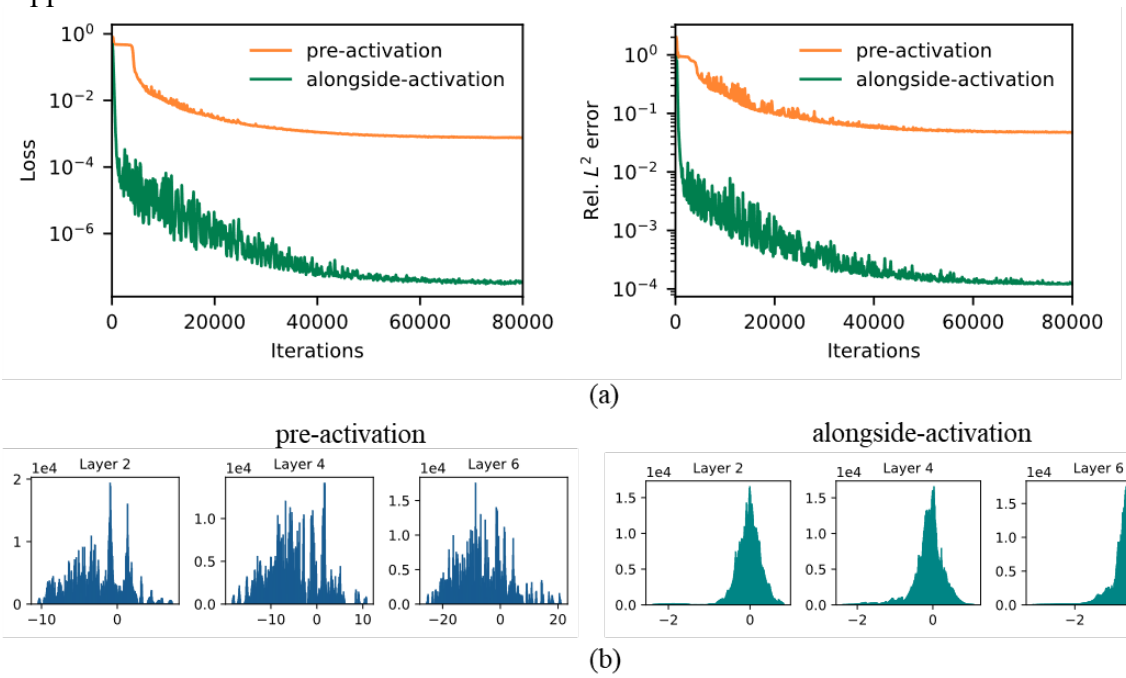


Figure 4. (a) Loss and relative L^2 error curves of different mask placement. (b) Pre-activation feature distributions of different mask placement.

4 Experiments

To demonstrate the effectiveness of the proposed method, we conducted experiments across several benchmarks. The methods compared against the proposed methods include vanilla PINNs [1], ResNet-based PINNs (ResNet-PINN), Weight Normalization-based PINNs (WN-PINNs), locally adaptive activation functions (LAAF) based PINNs [37], PirateNets [15], and ABU-PINN[13]. All the methods are trained with Tanh, GELU, SiLU, and SoftPlus as activation functions. Note that ABU-PINN is specifically designed to automatically select activation functions during training, and therefore produces only one result, while the other methods are evaluated separately with each of the four activation functions. Due to the difference in the structure, the depth of each model may vary, but the total trainable parameters are kept about the same. Adam is utilized as the optimizer. All experiments are conducted on a single NVIDIA GeForce RTX 4090 GPU. All results are averaged over 5 independent trials.

4.1 Convection Equation

We consider the convection equation expressed as below:

$$\frac{\partial u}{\partial t} + \beta \frac{\partial u}{\partial x} = 0,\ x \in [0,\ 2\pi],\ t \in [0,\ 1], \tag{23}$$

$$u(0,\ x) = \sin(x), \tag{24}$$

$$u(t,\ 0) = u(t,\ 2\pi), \tag{25}$$

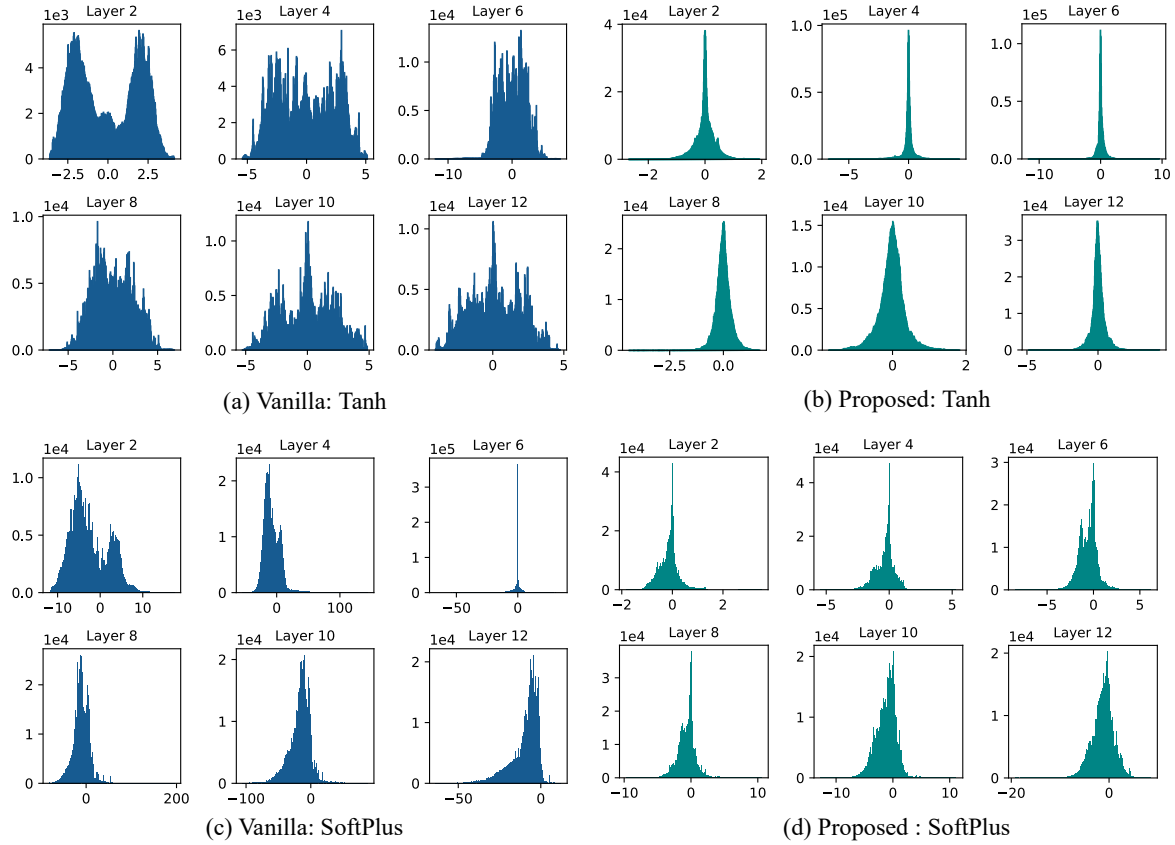
where we set β as 30.

Each model is composed of 13 hidden layers, each containing 256 neurons, and is trained for 50,000 iterations. For the proposed method, $\boldsymbol{\alpha}$ is initialized with all entries set to 1.0.

Table 1 shows the outcomes of different models. The best-performing method under each activation function is underlined, while the overall best result across all methods is highlighted in bold. As seen, the proposed method achieves consistently the best results across different activation functions. Specifically, for Tanh, the proposed method achieves a relative L^2 error one order of magnitude lower than that of the vanilla method. For SoftPlus, the result of the proposed method is 2 orders of magnitude lower than the vanilla method.

Table 1: Convection equation: relative L^2 error of different models.

Method	Tanh	GELU	SiLU	SoftPlus
Vanilla	7.29e-3	9.91e-4	1.37e-3	3.55e-1
ResNet-PINN	1.75e-2	1.74e-3	1.50e-3	2.41e-2
LAAF	7.30e-3	9.44e-4	7.23e-4	7.35e-2
PirateNet	1.98e-3	9.73e-3	4.54e-3	1.39e-2
WN-PINNs	3.17e-3	1.02e-3	4.89e-3	5.12e-3
ABU-PINN			2.98e-3	
Mask-PINN	<u>6.93e-4</u>	<u>5.26e-4</u>	**<u>4.79e-4</u>**	<u>1.66e-3</u>



(a) Vanilla: Tanh (b) Proposed: Tanh

(c) Vanilla: SoftPlus (d) Proposed : SoftPlus

Figure 5. Convection Equation: pre-activation distributions of the vanilla and proposed methods.

Fig. 5 gives the pre-activation distributions of the vanilla and proposed methods at the end of training, using SoftPlus and Tanh as activation functions. For Tanh, most of the vanilla pre-activations drift away from the origin, pushing many neurons into the saturated regime where gradients vanish. However, the proposed method maintains a more uniform, Gaussian-like distribution across hidden layers, which prevents saturation and ensures stable gradient flow throughout the network. This distributional stability accounts for the superior performance of the proposed method over the vanilla baseline. For SoftPlus, as the layer goes deeper, the spread of the feature values changes dramatically, and more values shift to the negative side. This arises from the fact that SoftPlus is an asymmetric activation function with $\mathrm{SoftPlus}(0) \neq 0$, which introduces a systematic bias in the feature values, as discussed in Section 3.3. Besides, such phenomenon implies that during training, the vanilla method's pre-activations tend to drift toward the linear regime of the SoftPlus activation function, which can negatively impact both the training dynamics and the model's expressiveness. In contrast, the proposed method shows a much slower change in the scale and keeps the feature distributions in a Gaussian-like form. This distinct difference between the feature distributions explains why our method yields significantly better performance when using SoftPlus.

4.2 1D Wave Propagation

Here we consider the 1D wave propagation equation as:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}, \ (x,t) \in [0,1] \times [0,5], \tag{26}$$

$$u(0,t) = u(1,t) = 0, \ t \in [0,5], \tag{27}$$

$$u(x,0)=\sin(\pi x),\ x\in[0,1], \tag{28}$$

$$\frac{\partial u}{\partial t}(x,0)=0, \tag{29}$$

the analytical solution for this problem is:

$$u(x,t)=\sin(\pi x)\cos(c\pi t). \tag{30}$$

Here we set $c=1$. All the models have 7 hidden layers, each containing 256 neurons, and are trained for 20,000 iterations. For the proposed method, $\boldsymbol{\alpha}$ is initialized with all entries set to 1.0.

Table 2: 1D Wave propagation: relative L^2 errors of different models.

Method	Tanh	GELU	SiLU	SoftPlus
Vanilla	7.33e-2	5.09e-3	8.52e-3	3.61e-2
ResNet-PINN	4.85e-2	7.52e-3	5.22e-3	7.67e-1
LAAF	1.04e-2	4.37e-3	5.73e-3	1.31e-2
PirateNet	1.59e-2	8.83e-3	9.95e-3	5.04e-2
WN-PINNs	6.14e-2	5.60e-3	7.30e-3	1.46e-1
ABU-PINN		4.82e-2		
Mask-PINN	**2.72e-3**	2.90e-3	4.82e-3	6.38e-3

Table 2 shows the outcomes of different models. The proposed method consistently achieves the best result across different activation functions. Interestingly, while the vanilla method yielded the poorest performance with Tanh, the proposed method achieved its best result using the same activation function. In Fig. 6, we show the variance evolution of the proposed method and vanilla method with Tanh as activation function. The variance for the vanilla method increases notably with training, suggesting severe distribution drift. The variance curves of the proposed method show relatively smaller values and stable trends after initial iterations, indicating controlled variance growth. This reflects effective regulation of pre-activations as depth and training progress.

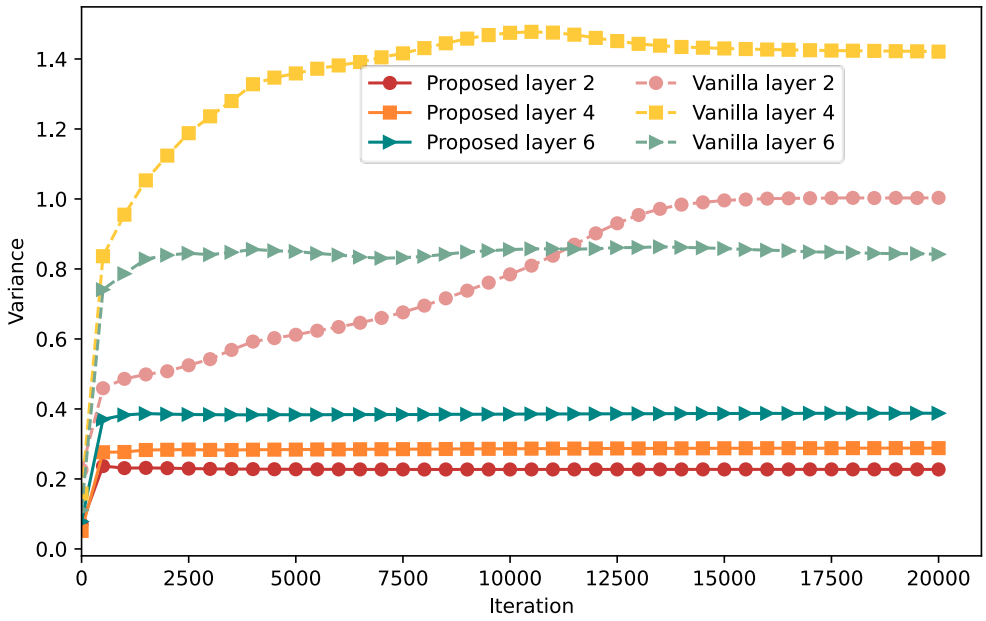


Figure 6. 1D wave propagation equation: variances of pre-activations of vanilla and proposed methods with Tanh activation.

4.3 Helmholtz Equation

We consider the Helmholtz equation expressed as below:

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+k^2 u-q(x,y)=0,\ x\in[-1,\ 1],\ y\in[-1,\ 1], \tag{31}$$

$$u(-1,y)=u(1,y)=u(x,-1)=u(x,1)=0, \tag{32}$$

where the source term $q(x,y)$ is in the form of:

$$\begin{aligned} q(x,y)=&-(a_1\pi)\sin(a_1\pi x)\sin(a_2\pi y)\\ &-(a_2\pi)\sin(a_1\pi x)\sin(a_2\pi y)\\ &+k^2\sin(a_1\pi x)\sin(a_2\pi y), \end{aligned} \tag{33}$$

here we set $a_1=6$, $a_2=6$ and $k=1$.

All models are composed of 11 hidden layers with 128 neurons each, except PirateNet, which uses 10 hidden layers to keep about the same number of parameters with other methods. Each model is trained for 50,000 iterations. For the proposed method, $\boldsymbol{\alpha}$ is initialized with all entries set to 10.0 for GELU and SiLU, 5.0 for Tanh, and 2.0 for SoftPlus.

Table 3: Helmholtz equation: relative L^2 errors of different models.

Method	Tanh	GELU	SiLU	SoftPlus
Vanilla	9.37e-1	7.20e-1	1.00	1.00
ResNet-PINN	3.48e-1	8.02e-1	7.10e-1	1.00
LAAF	3.26e-2	2.26e-2	1.93e-1	5.89e-1
PirateNet	1.17	3.56e-2	2.68e-1	7.15e-1
WN-PINNs	7.70e-1	7.12e-1	7.12e-1	1.00
ABU-PINN		2.83e-2		
Mask-PINN	**9.94e-3**	1.52e-2	1.99e-2	1.75e-2

Table 3 shows the outcomes of different models. As seen, the proposed method achieves the best performance across different activation functions. Notably, for activation functions SiLU and SoftPlus, other methods fail to get reliable results. In contrast, our model maintains comparable accuracy across all activation functions. This highlights its robustness across different activation functions.

Since the vanilla method performs best with GELU, we report the averaged loss curves of all models under this activation in Fig. 7 (ABU-PINN is excluded as it dynamically selects activation functions during training.) As shown, our method converges faster and reaches a lower final loss, indicating improved training dynamics.

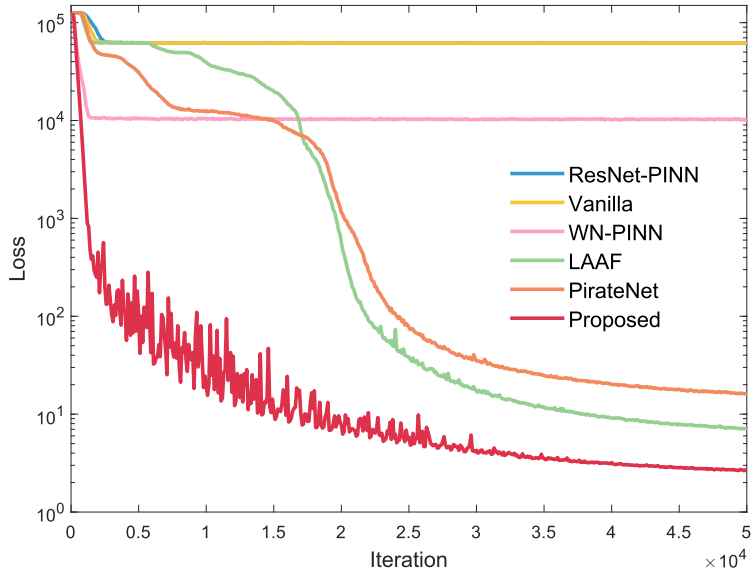
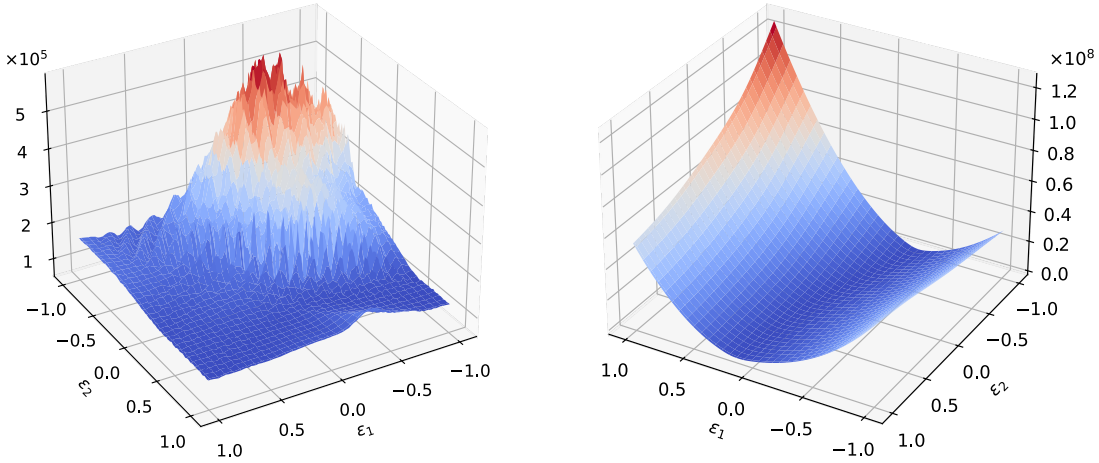


Figure 7. Helmholtz equation: averaged loss curves of different models with GELU as activation function.



(a) Vanilla (b) Proposed

Figure 8. Helmholtz equation: loss landscapes of vanilla method and the proposed method with GELU as activation function.

To further explore the reason behind such difference, Fig. 8 presents the loss landscapes of vanilla method and the proposed method with GELU as activation function. We visualize the loss landscape by perturbing the trained model along the first two leading eigenvectors of the Hessian and evaluating the loss at each perturbed point. It's evident that the proposed method yields a smoother loss landscape. For vanilla method, the surface is highly non-smooth, characterized by sharp spikes and irregular fluctuations. The irregular and rugged nature of the loss landscape indicates a tendency for the optimizer to become trapped in local minima with high loss value, which is consistent with the loss curve shown in Fig. 7.

4.4 Wider Structure

Although it is well-known that deeper and wider neural networks can improve approximation capability, for PINNs the situation is quite different. Theoretical analysis and empirical experiments have proven that PINNs degrade when the neural networks become wider or deeper [15]. While some solutions have been proposed to address the degradation problem in deep PINN architectures [15], [17], [38], the challenge of effectively utilizing wider

architectures remains largely unexplored. Here we show that with the proposed method, PINNs can effectively utilize wider structures.

With the Convection equation mentioned above, we conduct experiments on neural networks with different widths across different activation functions. For all the neural networks, we fix the depth to 3 hidden layers. Other settings are aligned with section 4.1. All the results are averaged from 5 independent trials. As shown in Fig. 9, vanilla PINNs suffer from a degradation issue as the width of the neural network increases. For the proposed method, a consistent improvement is witnessed as the neural network becomes wider. This indicates that the proposed method can mitigate the degradation problem observed in vanilla PINNs. Consequently, it allows PINNs to utilize wider structures, enhancing their robustness and accuracy.

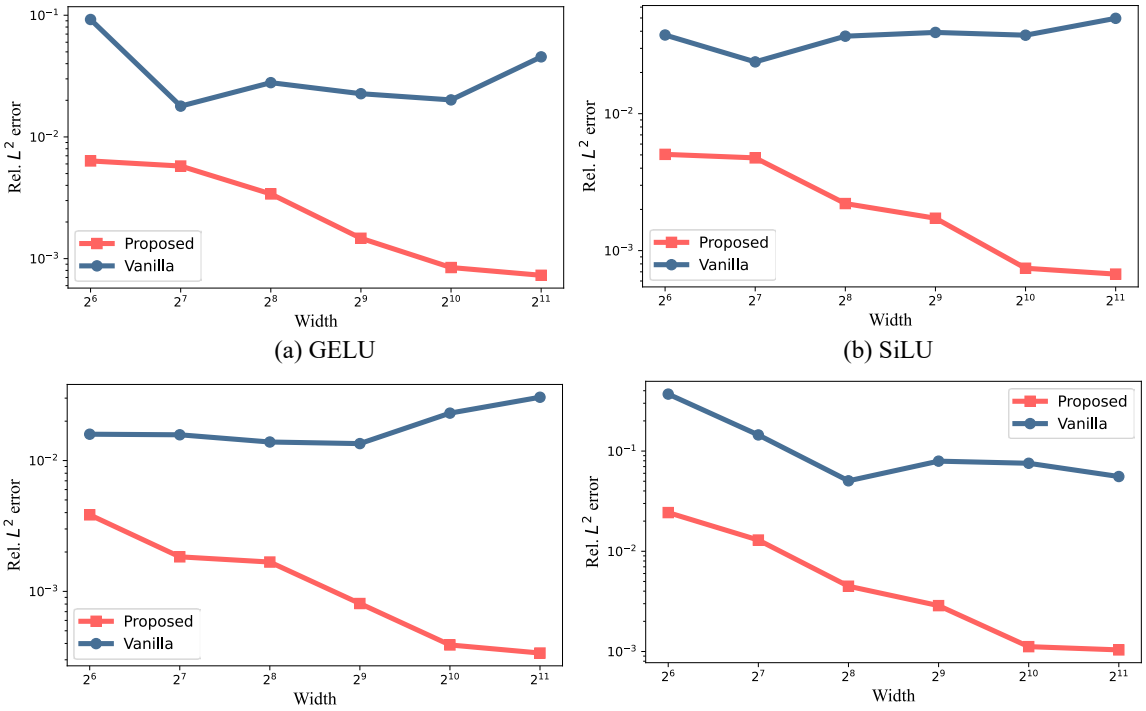
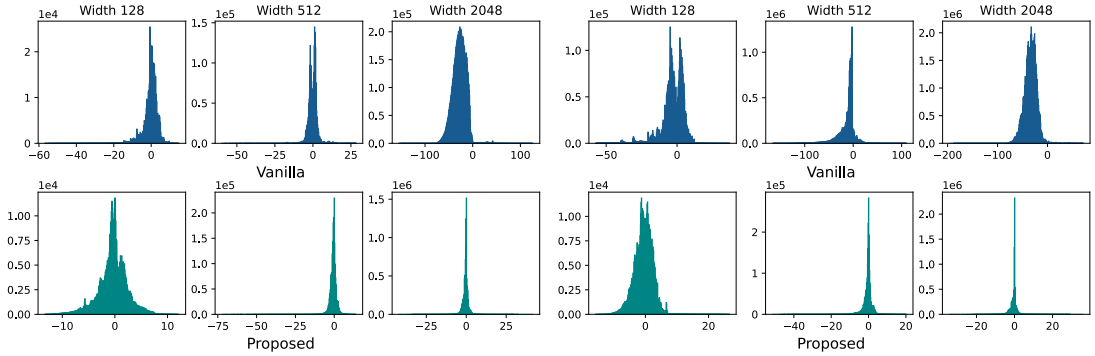


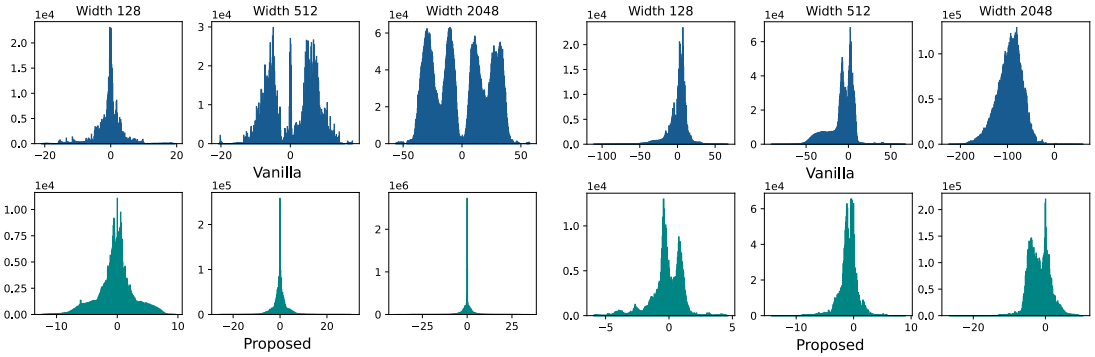
Figure 9. Relative L^2 errors of vanilla PINNs and the proposed method across various network widths.

To further explore the underlying reasons behind this improvement, we present the pre-activation distributions of the vanilla and proposed methods in Fig. 10. It clearly shows that with the width increasing, the pre-activation distribution of vanilla PINNs tends to drift away from the origin. Also note that for the GELU, SiLU, and SoftPlus activation functions, the pre-activation distributions in the vanilla method exhibit a clear tendency to drift toward the negative range. This suggests that neurons may become inactive or saturated due to these activation functions' activation properties, which severely limit the network's expressive power. The proposed method can effectively retain the balanced distribution shape with a narrower variance.

The results presented above highlight that the feature distribution instability and neuron saturation issues of wider neural networks lead to the degradation of PINNs. Our approach directly mitigates these issues by maintaining stable pre-activation distributions, allowing wider architectures to maintain their representational capability and achieve better accuracy.



(a) GELU (b) SiLU



(c) Tanh (d) SoftPlus

Figure 10. The pre-activation distributions of the 3rd hidden layer at the end of the training.

5 Discussion

5.1 Neural Tangent Kernel Analysis

In this section, we employ the neural tangent kernel (NTK) framework as a diagnostic tool to probe the training dynamics of Mask-PINN. While classical NTK theory is rigorously established in the infinite-width, small-learning-rate regime where the kernel remains nearly constant [39], recent studies have extended its use to finite-width PINNs by analyzing the instantaneous NTK during training [40-42]. From this perspective, the instantaneous NTK at each training stage allows us to examine convergence behavior through its spectral properties, even as the kernel evolves over time.

The neural tangent kernel operator $\mathbf{K}$ can be defined as:

$$\mathbf{K}(\tau)=\begin{bmatrix}\mathbf{K}_{uu}(\tau) & \mathbf{K}_{ur}(\tau)\\ \mathbf{K}_{ru}(\tau) & \mathbf{K}_{rr}(\tau)\end{bmatrix}, \tag{34}$$

where $\mathbf{K}_{ru}(\tau)=\mathbf{K}_{ur}^{T}(\tau)$. Here τ represents iteration. The specific entries are given by:

$$\left(\mathbf{K}_{uu}\right)_{ij}(\tau)=\left\langle \frac{d\mathsf{B}\left(x_b^i;\theta\right)}{d\theta}, \frac{d\mathsf{B}\left(x_b^j;\theta\right)}{d\theta}\right\rangle, \tag{35}$$

$$\left(\mathbf{K}_{ur}\right)_{ij}(\tau)=\left\langle \frac{d\mathsf{B}\left(x_b^i;\theta\right)}{d\theta}, \frac{d\mathsf{N}\left[u\left(x_r^j;\theta\right)\right]}{d\theta}\right\rangle, \tag{36}$$

$$\left(\mathbf{K}_{rr}\right)_{ij}(\tau)=\left\langle \frac{d\mathsf{N}\left[u\left(x_r^i;\theta\right)\right]}{d\theta}, \frac{d\mathsf{N}\left[u\left(x_r^j;\theta\right)\right]}{d\theta}\right\rangle. \tag{37}$$

NTK theory demonstrates that, with an infinitesimally small learning rate the gradient descent process can be represented as a continuous-time gradient flow:

$$\begin{bmatrix} \dfrac{d\mathrm{B}\left(x_b;\theta(\tau)\right)}{dt} \\ \dfrac{d\mathrm{N}\left[u\left(x_r;\theta(\tau)\right)\right]}{dt} \end{bmatrix} \approx -\mathbf{K}\cdot\begin{bmatrix} \mathrm{B}\left(x_b;\theta(\tau)\right)-g\left(x_b\right) \\ \mathrm{N}\left[u\left(x_r;\theta(\tau)\right)\right]-f\left(x_r\right) \end{bmatrix}, \tag{38}$$

$\theta(\tau)$ denotes the parameters of the network at τ. The solution of the ODE can be expressed as:

$$\begin{bmatrix} \mathrm{B}\left(x_b;\theta(\tau)\right) \\ \mathrm{N}\left[u\left(x_r;\theta(\tau)\right)\right] \end{bmatrix} \approx \left(I-e^{-\mathbf{K}\tau}\right)\cdot\begin{bmatrix} g\left(x_b\right) \\ f\left(x_r\right) \end{bmatrix}, \tag{39}$$

Since $\mathbf{K}$ is positive semi-definite, we can apply its spectral decomposition as $\mathbf{K}=\mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^T$, where $\boldsymbol{Q}$ is an orthogonal matrix whose i-th column is the eigenvector q_i of $\mathbf{K}$. $\boldsymbol{\Lambda}$ is a diagonal matrix containing the corresponding eigenvalues. Using the identity $e^{-K\tau}=Q^Te^{-\Lambda\tau}Q$, we obtain:

$$\mathbf{Q}^T\begin{bmatrix} \mathrm{B}\left(x_b;\theta(\tau)\right)-g\left(x_b\right) \\ \mathrm{N}\left[u\left(x_r;\theta(\tau)\right)\right]-f\left(x_r\right) \end{bmatrix} \approx -e^{-\boldsymbol{\Lambda}\tau}\mathbf{Q}^T\begin{bmatrix} g\left(x_b\right) \\ f\left(x_r\right) \end{bmatrix}, \tag{40}$$

The above equation indicates that the NTK's eigenvalues directly govern the convergence behaviour. Here we analyse the spectral properties of NTK under Burger's equation which expressed as follows:

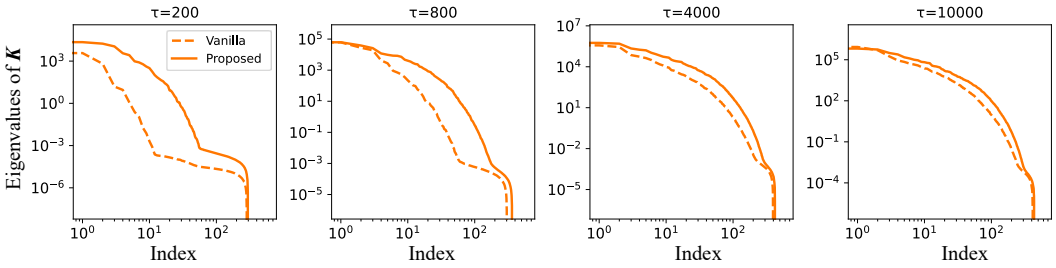
$$u_t+uu_x-\left(0.01/\pi\right)u_{xx}=0,\ x\in\left[-1,1\right],\ t\in\left[0,1\right], \tag{41}$$

$$u\left(0,x\right)=-\sin\left(\pi x\right), \tag{42}$$

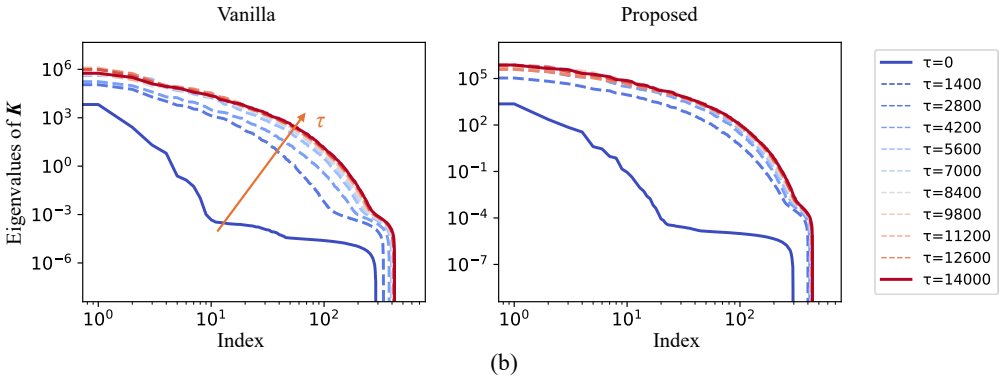
$$u\left(t,-1\right)=u\left(t,1\right)=0. \tag{43}$$

The comparison is conducted between the proposed method and vanilla PINN. Both have 5 hidden layers with 512 neurons each, and Tanh is adopted as activation function. The evolution of the NTK eigenvalue spectra is given in Fig. 11. Fig. 11 (a) reveals that the NTK spectra of the proposed method consistently outperforms vanilla across training. At small τ, it exhibits noticeably larger leading eigenvalues and a slower decay, which indicates richer feature representation and faster convergence. As τ increases, the gap narrows, but the proposed spectra still stay above vanilla across most of the range, showing it maintains a persistent advantage. In Fig. 11 (b), the proposed method's NTK spectrum converges to its steady shape significantly earlier than vanilla method, indicating stable optimization and efficient learning. These are confirmed by the loss and relative L^2 error curves observed during training. As shown in Fig. 12, both the curves of the proposed method decrease faster and to lower final values.

The NTK spectra reveal that Mask-PINNs maintain higher eigenvalues and converge to a steady shape earlier than the vanilla PINN, indicating more stable and efficient training dynamics.



(a)



(b)

Figure 11. Evolution of the NTK eigenvalue spectra during training.

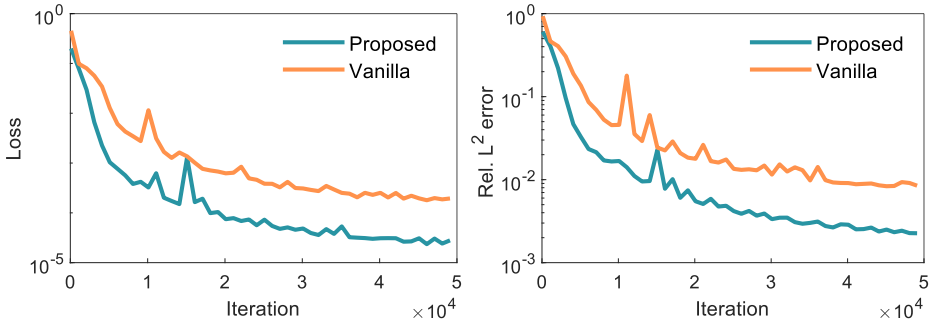


Figure 12. Loss and relative L^2 error curves during training.

5.2 Mask-PINN with Strong Optimization Protocols

While Mask-PINNs have shown consistent advantages across different benchmarks, recent studies suggest that optimization strategies can substantially affect the attainable accuracy of PINNs [43]. We therefore employ advanced optimization protocols to further unlock the performance potential of Mask-PINNs and compare our results with current state-of-the-art benchmarks.

5.2.1 Korteweg–De Vries Equation

The Korteweg–De Vries (KdV) equation is a nonlinear partial differential equation that describes the propagation of dispersive solitary waves. In this work, the KdV equation is considered in the following form.

$$u_t + \eta u u_x + \mu^2 u_{xxx} = 0,\ t \in [0,1],\ x \in [-1,1], \tag{44}$$

$$u(x,0) = \cos(\pi x), \tag{45}$$

$$u(t,-1) = u(t,1), \tag{46}$$

where $\eta = 1$ and $\mu = 0.022$.

The model has 4 hidden layers, each containing 64 neurons. Tanh is employed as the activation function. $\boldsymbol{\alpha}$ is initialized with all entries set to 1.0. We adopt a two-stage

optimization strategy: an initial 100,000-iteration Adam phase for coarse training, followed by 5,000 iterations of the SSBroyden method for fine refinement. The comparison of different models is presented in Table 4. The proposed method outperforms state-of-the-art model while utilizing an order of magnitude fewer parameters.

Table 4. KDV equation: relative L^2 errors of different models.

Method	Parameters	Rel. L^2 errors
Dirac delta function causal training [44]	3.35e4	2.45e-3
PirateNets [45]	5.27e5	4.27e-4
Deeper-PINNs [46]	1.52e5	2.95e-4
PINN with Optimized Optimizer [43]	1.68e4	4.15e-4
Mask-PINN	1.72e4	2.03e-4

The loss and relative L^2 error curves of the proposed method and vanilla PINN are shown in Fig. 13. As can be observed, the proposed method achieves faster convergence and higher final accuracy compared with the vanilla methods.

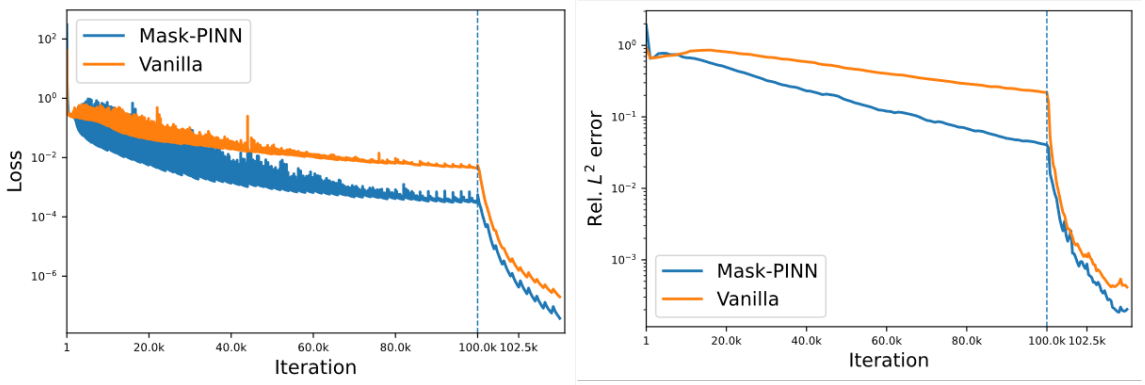


Figure 13. KDV equation: Loss and relative L^2 error curves of the proposed method and vanilla PINN under the Adam–SSBroyden hybrid optimization scheme.

5.2.2 Navier–Stokes Flow around A Cylinder

We consider steady incompressible flow with density $\rho = 1$ past a cylinder, described by the Navier–Stokes equations.

$$\mathbf{u}\nabla\mathbf{u} - \nu\Delta\mathbf{u} + \nabla p = 0, \tag{47}$$

$$\nabla\mathbf{u} = 0. \tag{48}$$

$\mathbf{u} = (u, v)$ denotes the velocity field and p the pressure. The kinematic viscosity is set to $\nu = 0.001$. The computational domain is a pipe geometry defined as $\Omega = [0, 2.2] \times [0, 0.41]$, containing a circular cylinder of radius 0.05 centered at (0.2, 0.2). No-slip boundary conditions are imposed on the wall and cylinder boundaries, i.e., $u(x, y) = v(x, y) = 0$. A parabolic velocity profile is prescribed at the inlet:

$$\mathbf{u}(0, y) = \left(\frac{4Uy(0.41 - y)}{0.41^2}, 0 \right), \tag{49}$$

with $U = 0.3$. For the outlet, the following outflow condition is adopted:

$$\nu\partial_{\mathbf{n}}\mathbf{u} - p\mathbf{n} = 0, \tag{50}$$

$\mathbf{n}$ represents the outer normal vector.

The model has 6 hidden layers, each containing 64 neurons. Tanh is utilized as activation function. $\boldsymbol{\alpha}$ is initialized with all entries set to 1.0. A two-stage training strategy is employed, consisting of 100,000 iterations of Adam optimization followed by 1500 iterations of the SSBroyden method. Table 5 shows that Mask-PINN outperforms existing PINN variants in

terms of accuracy while maintaining fewer or comparable parameters. Compared to the vanilla model, relative L^2 error is reduced by nearly 50%.

Table 5. Navier–Stokes flow around a cylinder: relative L^2 errors of different models.

Method	Parameters	Rel. L^2 errors
Expert PINNs [47]	5.03e4	5.41e-4
Deeper-PINNs [46]	3.96e4	2.90e-4
PINN with Optimized Optimizer [43]	2.53e4	1.36e-4
Mask-PINN	2.58e4	7.39e-5

As illustrated in the convergence plots in Fig.14. Mask-PINN consistently achieves lower training loss and relative L^2 error across iterations, indicating improved optimization efficiency and better solution quality.

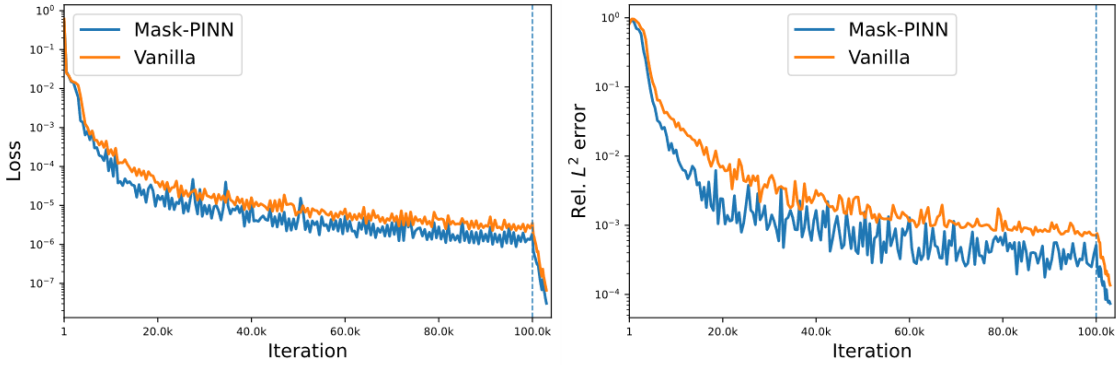


Figure 14. Navier–Stokes flow around a cylinder: Loss and relative L^2 error curves of the proposed method and vanilla PINN under the Adam–SSBroyden hybrid optimization scheme.

5.3 Sensitivity Study of α

In our experiments, the performance is observed to vary with the initialization strategy of the scaling parameter $\boldsymbol{\alpha}$. We therefore provide a sensitivity study to characterize this dependence and offer practical initialization guidelines. The sensitivity study of $\boldsymbol{\alpha}$ is conducted on Helmholtz equation mentioned above. Table 6 reports the relative L^2errors for different initial values of $\boldsymbol{\alpha}$. The results consistently show that neither very small nor very large α values yield optimal performance. Instead, satisfactory accuracy is obtained within a moderate range. To understand this behaviour, we revisit the masked activation defined in Section 3. For a hidden layer, the pre-activation and masked activation are defined as:

$$z = wH + b, \tag{51}$$

$$H = F(z;\boldsymbol{\alpha}) \odot \sigma(z). \tag{52}$$

During backpropagation, the gradient with respect to the pre-activation $z^{(l)}$is given by:

$$\frac{\partial \mathsf{L}}{\partial z} = \frac{\partial \mathsf{L}}{\partial H} \odot \frac{\partial H}{z}, \tag{53}$$

where

$$\frac{\partial H}{z} = \sigma'(z) \odot F(z;\boldsymbol{\alpha}) + \sigma(z) \odot F'(z;\boldsymbol{\alpha}), \tag{54}$$

and

$$F'(z;\boldsymbol{\alpha}) = 2\boldsymbol{\alpha}^2 \odot z \odot \exp\left[-(\boldsymbol{\alpha} \odot z)^2\right]. \tag{55}$$

For sufficiently small $\boldsymbol{\alpha}$, both $F(z;\boldsymbol{\alpha})$ and $F'(z;\boldsymbol{\alpha})$ are uniformly small over the typical initialization range. This leads to suppressed gradient propagation across layers. When α is large, the contraction region of $F(z;\boldsymbol{\alpha})$shrinks to a narrow neighborhood around the origin, so the mask behaves nearly as an identity mapping and provides limited global regularization. Meanwhile, the sharp transition induces strong localized gradient amplification near the

origin. As a result, features in this region may undergo rapid updates, weakening the intended stabilization effect of the mask.

Empirically, we recommend the range [1, 10] for activations satisfying $\sigma(0)=0$ and [1, 5] for activations with $\sigma(0)\neq 0$.

However, since the optimal α depends not only on the activation function but also on the underlying PDE and its induced feature statistics, developing a problem-adaptive strategy for selecting α remains an open research direction.

Table 6. Helmholtz equation: relative L^2 errors of different configurations

Activation function	$\boldsymbol{\alpha}$ = 0.01	$\boldsymbol{\alpha}$ = 1.0	$\boldsymbol{\alpha}$ = 2.0	$\boldsymbol{\alpha}$ = 5.0	$\boldsymbol{\alpha}$ = 10.0	$\boldsymbol{\alpha}$ = 20.0
Tanh	1.0	2.77e-2	1.11e-2	9.94e-3	1.44e-2	3.26e-2
GELU	1.0	4.07e-2	1.54e-2	1.57e-2	1.52e-2	1.64e-2
SiLU	1.0	4.75e-2	1.77e-1	8.42e-2	1.99e-2	2.01e-2
SoftPlus	1.0	3.62e-2	1.75e-2	4.53e-2	1.02e1	2.74e1

6 Conclusion

In this work, we focus on internal covariate shift, a well-known problem in deep learning that remains unresolved in PINNs and significantly undermines training stability and solution accuracy. We proposed Mask-PINNs, a novel architecture designed to alleviate this issue of PINNs without compromising the physical constraints encoded in the model. Specifically, a smooth, learnable mask function is applied pointwise across hidden layers to regulate feature distributions thereby mitigating activation drift. This leads to significantly improved training stability, predictive accuracy, and robustness across a wide range of PDE benchmarks and activation functions. Our experiments demonstrate that Mask-PINNs not only outperform existing methods across different settings but also enable the effective use of wider network architectures. By stabilizing feature distributions, Mask-PINNs provide a simple yet powerful tool for enhancing the representational capability of PINNs. The introduction of Mask-PINNs represents a significant advancement towards stable and scalable neural network architectures for solving PDEs, opening several promising avenues for future exploration in PINNs.

Since the proposed mask function mitigates internal covariate shift while preserving the coordinate-based nature of PINNs, it is worth exploring its applicability to other coordinate-based tasks, such as implicit neural representations [48]. While our results are promising, further investigation is needed. Notably, we observed that the initial value of the mask scaling parameter α has a considerable impact on training dynamics. Future research should aim to develop principled initialization strategies for α, tailored to specific tasks and network configurations, to further enhance the reliability and performance of Mask-PINNs.

Appendix A. Proof of Proposition 1

We analyze the network at a fixed training state, with parameters θ held constant. Let X denote the set of all collocation points, and let S be a mini-batch sampled from X. Consider a fixed input coordinate $x_i \in \mathsf{X}$.

By the definition of BN, the normalized feature $\hat{h}$ corresponding to $\mathbf{x}_i$ is given by:

$$\hat{h}(\mathbf{x}_i;\mathsf{S})=\gamma\left(\frac{h(\mathbf{x}_i)-\mu_{\mathsf{B}}(\mathsf{S})}{\sigma_{\mathsf{B}}(\mathsf{S})+\varepsilon}\right)+\beta. \tag{56}$$

where the μ_B and σ_B are computed over all samples in the current mini-batch.

Since these statistics are determined by the sample composition of S, there exist at least two mini-batches S and S' such that $x_i \in S \cap S'$ and:

$$(\mu_{\mathsf{B}}(\mathsf{S}), \sigma_{\mathsf{B}}^2(\mathsf{S})) \neq (\mu_{\mathsf{B}}(\mathsf{S}'), \sigma_{\mathsf{B}}^2(\mathsf{S}')). \tag{57}$$

Consequently, the normalized features at the identical coordinate $\mathbf{x}_i$ differ across batches:

$$\hat{h}(\mathbf{x}_i; \mathsf{S}) \neq \hat{h}(\mathbf{x}_i; \mathsf{S}'). \tag{58}$$

Consequently, the final output:

$$u_\theta(\mathbf{x}_i; \mathsf{S}) \neq u_\theta(\mathbf{x}_i; \mathsf{S}'). \tag{59}$$

Therefore, the network output at $\mathbf{x}_i$ is not uniquely determined by the input coordinate alone but depends on the arbitrary grouping of other samples in the batch. This results in an ambiguous representation that is incompatible with the fundamental requirement of PINNs to represent a single-valued, deterministic solution field over the domain X.

Appendix B. Proof of Proposition 2

Proof. Using the local expansions near 0:

$$F(u) = (\alpha u)^2 + O(u^4), \tag{60}$$

$$F'(u) = 2\alpha^2 u + O(u^3). \tag{61}$$

$$\sigma(u) = \sigma(0) + \sigma'(0)u + O(u^2), \tag{62}$$

$$\sigma'(u) = \sigma'(0) + O(u). \tag{63}$$

We have:

$$H'(u) = F'(u)\sigma(u) + F(u)\sigma'(u). \tag{64}$$

If $\sigma(0) = 0$:

$$H'(u) = 3\alpha^2\sigma'(0)u^2 + O(u^3). \tag{65}$$

Hence:

$$L_H(r) = 3\alpha^2 |\sigma'(0)| r^2 + O(r^3). \tag{66}$$

If $\sigma(0) \neq 0$:

$$H'(u) = 2\alpha^2\sigma(0)u + O(u^2). \tag{67}$$

Hence:

$$L_H(r) = 2\alpha^2 |\sigma(0)| r + O(r^2). \tag{68}$$

For vanilla method:

$$L_\sigma(r) = |\sigma'(0)| + O(r). \tag{69}$$

Dividing the leading asymptotic gives the stated result.

References

[1] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational physics,* vol. 378, pp. 686-707, 2019.

[2] X. Jia *et al.*, "Physics guided RNNs for modeling dynamical systems: A case study in simulating lake temperature profiles," in *Proceedings of the 2019 SIAM international conference on data mining*, 2019: SIAM, pp. 558-566.
[3] M. Raissi, A. Yazdani, and G. E. Karniadakis, "Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations," *Science,* vol. 367, no. 6481, pp. 1026-1030, 2020.
[4] S. Cai *et al.*, "Artificial intelligence velocimetry and microaneurysm-on-a-chip for three-dimensional analysis of blood flow in physiology and disease," *Proceedings of the National Academy of Sciences,* vol. 118, no. 13, p. e2100697118, 2021.
[5] J. Zhang and X. Zhao, "Three-dimensional spatiotemporal wind field reconstruction based on physics-informed deep learning," *Applied Energy,* vol. 300, p. 117390, 2021.
[6] S. Cai, Z. Mao, Z. Wang, M. Yin, and G. E. Karniadakis, "Physics-informed neural networks (PINNs) for fluid mechanics: A review," *Acta Mechanica Sinica,* vol. 37, no. 12, pp. 1727-1738, 2021.
[7] S. Cai, Z. Wang, S. Wang, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks for heat transfer problems," *Journal of Heat Transfer,* vol. 143, no. 6, p. 060801, 2021.
[8] H. Hu, L. Qi, and X. Chao, "Physics-informed Neural Networks (PINN) for computational solid mechanics: Numerical frameworks and applications," *Thin-Walled Structures,* p. 112495, 2024.
[9] X.-X. Chen, P. Zhang, and Z.-Y. Yin, "Physics-Informed neural network solver for numerical analysis in geoengineering," *Georisk: Assessment and Management of Risk for Engineered Systems and Geohazards,* vol. 18, no. 1, pp. 33-51, 2024.
[10] Z. Wu, H. Wang, C. He, B. Zhang, T. Xu, and Q. Chen, "The application of physics-informed machine learning in multiphysics modeling in chemical engineering," *Industrial & Engineering Chemistry Research,* vol. 62, no. 44, pp. 18178-18204, 2023.
[11] Y. Bai, T. Chaolu, and S. Bilige, "The application of improved physics-informed neural network (IPINN) method in finance," *Nonlinear Dynamics,* vol. 107, no. 4, pp. 3655-3667, 2022.
[12] K. Kashinath *et al.*, "Physics-informed machine learning: case studies for weather and climate modelling," *Philosophical Transactions of the Royal Society A,* vol. 379, no. 2194, p. 20200093, 2021.
[13] H. Wang, L. Lu, and G. Huang, "Learning Specialized Activation Functions for Physics-Informed Neural Networks," *Communications in Computational Physics,* vol. 34, no. 4, pp. 869-906, 2023.
[14] J. Zhang and C. Ding, "Simple yet effective adaptive activation functions for physics-informed neural networks," *Computer Physics Communications,* vol. 307, p. 109428, 2025.
[15] S. Wang, B. Li, Y. Chen, and P. Perdikaris, "PirateNets: Physics-informed Deep Learning with Residual Adaptive Networks," *arXiv preprint arXiv:2402.00326,* 2024.
[16] M. Cooley, R. M. Kirby, S. Zhe, and V. Shankar, "HyResPINNs: Adaptive Hybrid Residual Networks for Learning Optimal Combinations of Neural and RBF Components for Physics-Informed Modeling," *arXiv preprint arXiv:2410.03573,* 2024.
[17] F. Jiang, X. Hou, and M. Xia, "Element-wise multiplication based deeper physics-informed neural networks," *arXiv preprint arXiv:2406.04170,* 2024.
[18] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," *Advances in neural information processing systems,* vol. 33, pp. 7462-7473, 2020.

[19] H. woo Lee, H. Choi, and H. Kim, "Robust Weight Initialization for Tanh Neural Networks with Fixed Point Analysis," in *The Thirteenth International Conference on Learning Representations*.

[20] R. Bischof and M. A. Kraus, "Multi-objective loss balancing for physics-informed deep learning," *Computer Methods in Applied Mechanics and Engineering,* vol. 439, p. 117914, 2025.

[21] Q. Liu, M. Chu, and N. Thuerey, "Config: Towards conflict-free training of physics informed neural networks," *arXiv preprint arXiv:2408.11104,* 2024.

[22] S. Wang, A. K. Bhartari, B. Li, and P. Perdikaris, "Gradient Alignment in Physics-informed Neural Networks: A Second-Order Optimization Perspective," *arXiv preprint arXiv:2502.00604,* 2025.

[23] S. Wang, Y. Teng, and P. Perdikaris, "Understanding and mitigating gradient flow pathologies in physics-informed neural networks," *SIAM Journal on Scientific Computing,* vol. 43, no. 5, pp. A3055-A3081, 2021.

[24] Y. Zhang and L.-J. Deng, "Loss-driven dynamic weight and residual transformation in physics-informed neural network," *Neural Networks,* p. 107841, 2025.

[25] G. Farhani, N. H. Dashtbayaz, A. Kazachek, and B. Wang, "A simple remedy for failure modes in physics informed neural networks," *Neural Networks,* vol. 183, p. 106963, 2025.

[26] H. Gao, L. Sun, and J.-X. Wang, "PhyGeoNet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain," *Journal of Computational Physics,* vol. 428, p. 110079, 2021.

[27] K. Shukla, J. D. Toscano, Z. Wang, Z. Zou, and G. E. Karniadakis, "A comprehensive and FAIR comparison between MLP and KAN representations for differential equations and operator networks," *Computer Methods in Applied Mechanics and Engineering,* vol. 431, p. 117290, 2024.

[28] B. Yuan, H. Wang, A. Heitor, and X. Chen, "f-PICNN: a physics-informed convolutional neural network for partial differential equations with space-time domain," *Journal of Computational Physics,* vol. 515, p. 113284, 2024.

[29] Y. Wang, Y. Yao, and Z. Gao, "An extrapolation-driven network architecture for physics-informed deep learning," *Neural Networks,* vol. 183, p. 106998, 2025.

[30] W. Wu, S. Duan, Y. Sun, Y. Yu, D. Liu, and D. Peng, "Deep fuzzy physics-informed neural networks for forward and inverse PDE problems," *Neural Networks,* vol. 181, p. 106750, 2025.

[31] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," presented at the Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37, Lille, France, 2015.

[32] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," *arXiv preprint arXiv:1811.03378,* 2018.

[33] M. Lee, "Mathematical analysis and performance evaluation of the gelu activation function in deep learning," *Journal of Mathematics,* vol. 2023, no. 1, p. 4229924, 2023.

[34] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, "Activation functions in deep learning: A comprehensive survey and benchmark," *Neurocomputing,* vol. 503, pp. 92-108, 2022.

[35] J. L. Ba, "Layer normalization," *arXiv preprint arXiv:1607.06450,* 2016.

[36] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," *Advances in neural information processing systems,* vol. 29, 2016.

[37] A. D. Jagtap, K. Kawaguchi, and G. Em Karniadakis, "Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks," *Proceedings of the Royal Society A,* vol. 476, no. 2239, p. 20200334, 2020.

[38] A. Aygun, R. Maulik, and A. Karakus, "Physics-informed neural networks for mesh deformation with exact boundary enforcement," *Engineering Applications of Artificial Intelligence,* vol. 125, p. 106660, 2023.

[39] A. Jacot, F. Gabriel, and C. Hongler, "Neural tangent kernel: Convergence and generalization in neural networks," *Advances in neural information processing systems,* vol. 31, 2018.

[40] S. Wang, X. Yu, and P. Perdikaris, "When and why PINNs fail to train: A neural tangent kernel perspective," *Journal of Computational Physics,* vol. 449, p. 110768, 2022.

[41] S. A. Faroughi and F. Mostajeran, "Neural tangent kernel analysis to probe convergence in physics-informed neural solvers: PIKANs vs. PINNs," *arXiv preprint arXiv:2506.07958,* 2025.

[42] S. Wang, H. Wang, and P. Perdikaris, "Improved architectures and training algorithms for deep operator networks," *Journal of Scientific Computing,* vol. 92, no. 2, p. 35, 2022.

[43] E. Kiyani, K. Shukla, J. F. Urbán, J. Darbon, and G. E. Karniadakis, "Optimizing the optimizer for physics-informed neural networks and Kolmogorov-Arnold networks," *Computer Methods in Applied Mechanics and Engineering,* vol. 446, p. 118308, 2025.

[44] V. A. Es' kin, D. V. Davydov, E. D. Egorova, A. O. Malkhanov, M. A. Akhukov, and M. E. Smorkalov, "About optimal loss function for training physics-informed neural networks under respecting causality," *arXiv preprint arXiv:2304.02282,* 2023.

[45] S. Wang, B. Li, Y. Chen, and P. Perdikaris, "Piratenets: Physics-informed deep learning with residual adaptive networks," *Journal of Machine Learning Research,* vol. 25, no. 402, pp. 1-51, 2024.

[46] F. Jiang, M. Xia, X. Hou, and J. Ye, "Deeper-Pinns: Unlocking the Power of Deep Physics-Informed Neural Networks," *Available at SSRN 5266541*.

[47] S. Wang, S. Sankaran, H. Wang, and P. Perdikaris, "An expert's guide to training physics-informed neural networks," *arXiv preprint arXiv:2308.08468,* 2023.

[48] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 165-174.