# ROBUST WEIGHT INITIALIZATION FOR TANH NEURAL NETWORKS WITH FIXED POINT ANALYSIS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

As a neural network's depth increases, it can achieve high generalization performance. However, training deep networks is challenging due to gradient and signal propagation issues. To address these challenges, extensive theoretical research and various methods have been introduced. Despite these advances, effective weight initialization methods for tanh neural networks remain underexplored. This paper presents a novel weight initialization method for Neural Networks with tanh activation function. Based on an analysis of the fixed points of the function $\tanh(ax)$, our proposed method aims to determine values of $a$ that mitigate activation saturations. A series of experiments on various classification datasets and Physics-Informed Neural Networks demonstrate that the proposed method outperforms Xavier initialization methods (with or without normalization) in terms of robustness to network size variations, data efficiency, and convergence speed.

## 1 INTRODUCTION

Deep learning has enabled substantial advancements in state-of-the-art performance across various domains (LeCun et al., 2015; He et al., 2016). In general, the expressivity of neural networks increases exponentially with depth (Poole et al., 2016; Raghu et al., 2017), enabling high generalization performance. However, deeper networks often face challenges such as vanishing or exploding gradients and poor signal propagation (Bengio et al., 1993). These challenges have driven the development of effective weight initialization methods tailored to various activation functions. Xavier initialization (Glorot & Bengio, 2010) ensures signals stay in the non-saturated region for sigmoid and hyperbolic tangent activations, while He initialization (He et al., 2015) maintains stable variance for ReLU networks. Especially in ReLU neural networks, several weight initialization methods have been proposed to mitigate the dying ReLU problem, which hinders signal propagation in deep networks (Lu et al., 2019; Lee et al., 2024). However, to the best of our knowledge, research on initialization methods that are robust to the size of tanh networks remains underexplored. Tanh networks commonly use Xavier initialization (Raissi et al., 2019; Jagtap et al., 2022; Rathore et al., 2024) and are widely applied in various domains, such as Physics-Informed Neural Networks (PINNs) (Raissi et al., 2019) and Recurrent Neural Networks (RNNs) (Rumelhart et al., 1986), with performance often dependent on model size and initialization randomness (Liu et al., 2022).

The main contribution of this paper is the proposal of a simple weight initialization method for Feed-Forward Neural Networks (FFNNs) with tanh activation function. The proposed method is data-efficient and demonstrates robustness to variations in network size. It reduces the dependency on normalization techniques such as Batch Normalization (Ioffe, 2015) and Layer Normalization (Ba, 2016). As a result, it reduces the requirement for extensive hyperparameter tuning, such as the number of hidden layers and units, and avoids normalization computational overhead. The theoretical foundation for this approach is based on the fixed point of the function $\tanh(ax)$. We investigate the performance of the proposed method on two tasks: classification and Physics-Informed Neural Networks (PINNs). For classification tasks, we assess the method's performance across various FFNN sizes using datasets including MNIST, Fashion MNIST, CIFAR-10, and CIFAR-100. The results demonstrated improved validation accuracy and loss compared to Xavier initialization with Batch Normalization (BN) or Layer Normalization (LN). For PINNs, the method exhibits robustness across diverse network sizes and demonstrates its effectiveness in solving a wide range of PDE problems. Notably, for both tasks, the proposed method outperforms Xavier initialization in terms of data efficiency, achieving improved performance even with limited data.

**Contributions.** Our contributions can be summarised as follows:

- We identify the conditions under which activation values remain non-vanishing as the depth of the neural network increases, using a fixed-point analysis (Section 3.1 and 3.2).
- We propose a novel weight initialization method for tanh neural networks that is robust to variations in network size and demonstrates high data efficiency (Section 3.2 and 3.3).
- We experimentally show that the proposed method is more robust to network size variations on image benchmarks and PINNs (Section 4).
- We experimentally show that the proposed method is more data-efficient than Xavier initialization, with or without normalization, on image benchmarks and PINNs (Section 4).

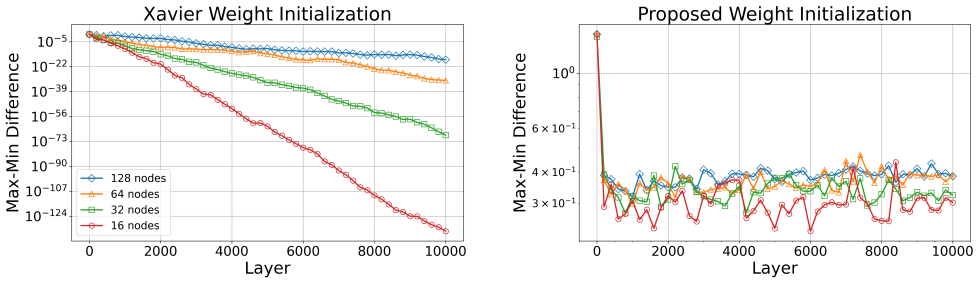
## 2 Related works

The expressivity of neural networks grows exponentially with depth, resulting in improved generalization performance (Poole et al., 2016; Raghu et al., 2017). Weight initialization is crucial for training deep networks effectively (Saxe et al., 2014; Mishkin & Matas, 2016). Xavier (Glorot & Bengio, 2010) and He He et al. (2015) initialization are common initialization methods typically used with tanh and ReLU activation functions, respectively. Various initialization methods have been proposed to facilitate the training of deeper ReLU neural networks (Lu et al., 2019; Bachlechner et al., 2021; Zhao et al., 2022; Lee et al., 2024). In contrast, research on weight initialization for neural networks using tanh activation remains limited. Despite this, tanh neural networks have gained popularity in recent years, particularly in applications such as physics-informed neural networks (PINNs), where their performance can be sensitive to the randomness of initialization methods.

PINNs have shown promising results in solving forward, inverse, and multiphysics problems arising in science and engineering. (Lu et al., 2021; Karniadakis et al., 2021; Cuomo et al., 2022b;a; Yin et al., 2021; Wu et al., 2023; Hanna et al., 2022; Bararnia & Esmaeilpour, 2022; Shukla et al., 2020; Zhu et al., 2024; Hosseini et al., 2023; Mao et al., 2020). PINNs approximate solutions to partial differential equations (PDEs) using neural networks. They are trained by minimizing a loss function, typically the sum of least-squares, which incorporates the residual of PDEs, boundary conditions, and initial conditions. This loss is usually minimized using gradient-based optimizers such as Adam (Kingma, 2014), L-BFGS (Liu & Nocedal, 1989), or a combination of both. Universal approximation theories (Cybenko, 1989; Hornik et al., 1989; Hornik, 1991; Park et al., 2020; Guliyev & Ismailov, 2018b; Shen et al., 2022; Guliyev & Ismailov, 2018a; Maiorov & Pinkus, 1999; Yarotsky, 2017; Gripenberg, 2003) guarantee the capability and performance of neural networks as an approximation of the analytic solution to PDE. However, PINNs still face challenges in accuracy, stability, computational complexity, and tuning optimal hyperparameters of loss terms.

To alleviate these issues, many authors have introduced enhanced versions of PINNs: (1) the self-adaptive loss balanced PINNs (lbPINNs) that automatically adjust the hyperparameters of loss terms during the training process (Xiang et al., 2022), (2) the Bayesian PINNs (B-PINNs) that are specialized to deal with forward and inverse nonlinear problems with noisy data (Yang et al., 2021), (3) Rectified PINNs (RPINNs) that are trained with the gradient information from the numerical solution by the multigrid method and designed for solving stationary PDEs (Peng et al., 2022), (4) Auxiliary Pinns (A-PINNs) that effectively handle integro-differential equations (Yuan et al., 2022), (5) conservative PINNs (cPINNs) and exetended PINNs (XPINNs) that adopt the domain decomposition technique (Jagtap et al., 2020; Jagtap & Karniadakis, 2020), (6) parrel PINNs that reduces the computational cost of cPINNs and XPINNs (Shukla et al., 2021), (7) gradient-enhanced PINNs (gPINNs) that use the gradient of the PDE loss term with respect to the network inputs (Yu et al., 2022).

While these advancements address various challenges in PINNs, activation functions, and their initialization strategies remain crucial for achieving optimal performance. The tanh activation function is known to perform well in PINNs (Raissi et al., 2019), as detailed experimental results provided in Appendix C.2. Xavier initialization is commonly used as the standard choice for tanh networks in existing studies (Jin et al., 2021; Son et al., 2023; Yao et al., 2023; Gnanasambandam et al., 2023; Song et al., 2024). However, our experimental results indicate that the effectiveness of Xavier initialization decreases as network size increases. Moreover, performance improvements achieved




(a) Xavier Initialization

(b) Proposed Initialization

Figure 1: Difference between maximum and minimum activation values at each layer when propagating $3,000$ input data through a $10,000$-layer tanh FFNN, using Xavier initialization **(left)** and the proposed initialization **(right)**. Experiments were conducted on networks with $10,000$ hidden layers, each having the same number of nodes: 16, 32, 64, or 128.

through Batch Normalization or Layer Normalization are limited, and the method demonstrates sensitivity to the amount of training data, particularly with smaller datasets. Although there has been a recent result on an initialization method for PINNs, the method relies on transfer learning (Tarbiyati & Nemati Saray, 2023). Thus, we propose a weight initialization method that is robust across varying network sizes, achieves high data efficiency, and reduces reliance on both transfer learning and normalization techniques.

## 3 Proposed Weight Initialization Method

In this section, we discuss the proposed weight initialization method. Section 3.1 introduces the theoretical motivation behind the method. Section 3.2 presents how to derive the initial weight matrix that satisfies the conditions outlined in Section 3.1. Finally, in Section 3.3, we suggest the optimal hyperparameter $\sigma_z$ in the proposed method.

### 3.1 Theoretical Motivation

Experimental results in Figure 1 reveal that when Xavier initialization is employed in FFNNs with tanh activation, the distribution of activation values tends to cluster around zero in deeper layers. This vanishing of activation values can hinder the training process due to a discrepancy between the activation values and the desired output. However, theoretically preventing this phenomenon is not straightforward. In this section, we provide a theoretical analysis based on a fixed point of $\tanh(ax)$ to bypass the phenomenon. Before giving the theoretical foundations, consider the basic results for a tanh activation function. Recall that $x^*$ is a fixed point of a function $f$ if $x^*$ belongs to both the domain and the codomain of $f$, and $f(x^*) = x^*$.

**Lemma 1.** *For a fixed $a > 0$ define the function $\phi_a : \mathbb{R} \to \mathbb{R}$ given as*

$$\phi_a(x) := \tanh(ax).$$

*Then, there exists a fixed point $x^*$. Furthermore,*

(1) *if $0 < a \leq 1$, then $\phi$ has a unique fixed point $x^* = 0$.*

(2) *if $a > 1$, then $\phi$ has three distinct fixed points: $x^* = -\xi_a$, $0$, $\xi_a$ such that $\xi_a > 0$.*

*Proof.* The proof is detailed in in Appendix A.1. □

The function $\tanh(x)$ satisfies $\tanh(x) < x$ for all $x > 0$. However, according to Lemma 1, the behavior of $\tanh(ax)$ changes when $a > 1$. When $x > \xi_a$, the inequality $\tanh(ax) < x$ holds. When $x < \xi_a$, the inequality $\tanh(ax) > x$ is satisfied. At $x = \xi_a$, the equality $\tanh(ax) = x$ holds. In Lemma 2, we address the convergence properties of iteratively applying $\tanh(ax)$ for any $x > 0$.

**Lemma 2.** *For a given initial value $x_0 > 0$ define*

$$x_{n+1} = \phi_a(x_n), \quad n = 0, 1, 2, \ldots.$$

*Then $\{x_n\}_{n=1}^{\infty}$ converges regardless of the positive initial value $x_0 > 0$. Moreover,*

(1) *if $0 < a \leq 1$, then $x_n \to 0$ as $n \to \infty$.*

(2) *if $a > 1$, then $x_n \to \xi_a$ as $n \to \infty$.*

*Proof.* The proof is detailed in Appendix A.2. □

According to Lemma 2, for $a > 1$ and $x_0 > \xi_a > 0$, the sequence $\{x_n\}$ satisfies $\tanh(ax_n) > \tanh(ax_{n+1}) > \xi_a$ for all $n \in \mathbb{N}$. Similarly, when $0 < x_0 < \xi_a$, the sequence satisfies $\tanh(ax_n) < \tanh(ax_{n+1}) < \xi_a$ for all $n \in \mathbb{N}$. Given $a > 1$ and $x_0 < 0$, the sequence converges to $-\xi_a$ as $n \to \infty$ due to the odd symmetry of $\tanh(ax)$. Therefore, when $a > 1$, for an arbitrary initial value $x_0 > 0$ or $x_0 < 0$, the sequence $\{x_n\}$ converges to $\xi_a$ or $-\xi_a$, respectively, as $n \to \infty$.

Note that the parameter $a$ in Lemma 2 does not change across all iterations. In Propositions 3 and Corollary 4, we address cases where the value of $a$ varies with each iteration.

**Proposition 3.** *Let $\{a_n\}_{n=1}^{\infty}$ be a positive real sequence, i.e., $a_n > 0$ for all $n \in \mathbb{N}$, such that only finitely many elements are greater than 1. Suppose that $\{\Phi_m\}_{m=1}^{\infty}$ is a sequence of functions defined as for each $m \in \mathbb{N}$*

$$\Phi_m = \phi_{a_m} \circ \phi_{a_{m-1}} \circ \cdots \circ \phi_{a_1}.$$

*Then for any $x \in \mathbb{R}$*

$$\lim_{m \to \infty} \Phi_m(x) = 0.$$

*Proof.* The proof is detailed in Appendix A.3. □

**Corollary 4.** *Let $\epsilon > 0$ be given. Suppose that $\{a_n\}_{n=1}^{\infty}$ be a positive real sequence such that only finitely many elements are lower than $1 + \epsilon$. Then for any $x \in \mathbb{R} \setminus \{0\}$*

$$\lim_{m \to \infty} |\Phi_m(x)| \geq \xi_{1+\epsilon}$$

*Proof.* The proof is detailed in Appendix A.4. □

Based on Proposition 3 and Corollary 4, if there exists a sufficiently large $N$ such that all elements $a_n$ for $n \geq N$ are either less than 1 or greater than $1 + \epsilon$, then for any $x_0 \in \mathbb{R} \setminus \{0\}$, the sequence either converges to 0 or satisfies $|\Phi_m(x_0)| \geq \xi_{1+\epsilon}$ as $m \to \infty$, respectively. This result implies that if the sequence $\{\Phi_m\}_{m=1}^{M}$ is finite, $a_n$ for $N \leq n \leq M$, where $N$ is an arbitrarily chosen index close to $M$, significantly influence the values of $\Phi_M(x_0)$.

### 3.2 THE DERIVATION OF THE PROPOSED WEIGHT INITIALIZATION METHOD

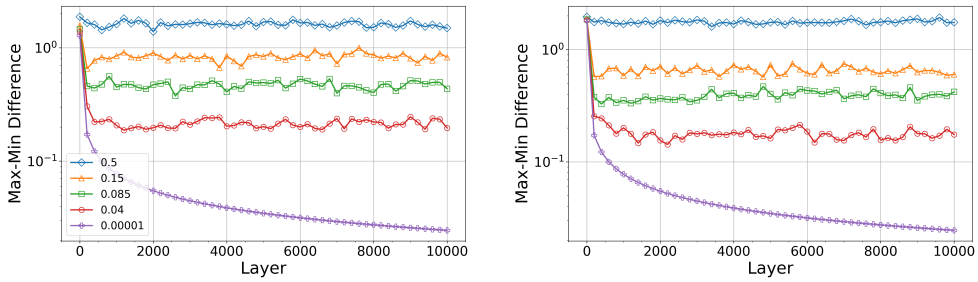
**Remark** Based on the theoretical motivations discussed in the previous section, we propose a weight initialization method that satisfies the following conditions during the initial forward pass:
(i) It avoids the phenomenon where activation values cluster around zero in deeper layers.
(ii) It ensures that the distribution of activation values in deeper layers is approximately normal.

**Notation** Consider a feedforward neural network with $L$ layers. The network processes $K$ training samples, denoted as pairs $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{K}$, where $\boldsymbol{x}_i \in \mathbb{R}^{N_x}$ is training input and $\boldsymbol{y}_i \in \mathbb{R}^{N_y}$ is its corresponding output. The iterative computation at each layer $\ell$ is defined as follows:

$$\boldsymbol{x}^{\ell} = \tanh(\mathbf{W}^{\ell}\boldsymbol{x}^{\ell-1} + \mathbf{b}^{\ell}) \in \mathbb{R}^{N_\ell} \quad \text{for all } \ell = 1, \ldots, L,$$

where $\mathbf{W}^{\ell} \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$ is the weight matrix, $\mathbf{b}^{\ell} \in \mathbb{R}^{N_\ell}$ is the bias, and $\tanh(\cdot)$ is an element-wise activation hyperbolic tangent function.

**Signal Propagation Analysis** We present a simplified analysis of signal propagation in FFNNs with the tanh activation function. For notational convenience, it is assumed that all hidden layers, as well as the input and output layers, have a dimension of $n$, i.e., $N_\ell = n$ for all $\ell$. Given an arbitrary input




(a) Same dimension

(b) Varying dimensions

Figure 2: Difference between maximum and minimum activation values at each layer when propagating $3,000$ input data through a $10,000$-layer tanh FFNN, using the proposed initialization with $\alpha$ set to $0.04, 0.085, 0.15$, and $0.5$. Network with $10,000$ hidden layers, each with 32 nodes **(left)**, and a network with alternating hidden layers of 64 and 32 nodes **(right)**.

vector $\boldsymbol{x} = (x_1, \ldots, x_n)$, the first layer activation $\boldsymbol{x}^1 = \tanh(\mathbf{W}^1\boldsymbol{x})$ can be expressed componentwise as:

$$x_i^1 = \tanh\left(w_{i1}^1 x_1 + \cdots + w_{in}^1 x_n\right) = \tanh\left(\left(w_{ii}^1 + \sum_{\substack{j=1\\ j\neq i}}^{n} \frac{w_{ij}^1 x_j}{x_i}\right)x_i\right), \text{ for } i = 1, \ldots, n.$$

For the $k+1$-th layer, $i = 1, \ldots, n$, this expression can be generalized as:

$$x_i^{k+1} = \tanh\left(a_i^{k+1} x_i^k\right), \text{ where } \quad a_i^{k+1} = w_{ii}^{k+1} + \sum_{\substack{j=1\\ j\neq i}}^{n} \frac{w_{ij}^{k+1} x_j^k}{x_i^k}. \tag{1}$$

The equation 1 follows the form of $\tanh(ax)$, as discussed in Section 3.2. According to Lemma 2, when $a > 1$, for an arbitrary initial value $x_0 > 0$ or $x_0 < 0$, the sequence $\{x_k\}$ defined by $x_{k+1} = \tanh(ax_k)$ converges to $\xi_a$ or $-\xi_a$, respectively, as $k \to \infty$. This result indicates that the sequence converges to the fixed point $\xi_a$ regardless of the initial value $x_0$. From the perspective of signal propagation in tanh-based FFNNs, this ensures that the activation values do not vanish as the network depth increases. Furthermore, by Proposition 3, if $a_i^k \leq 1$ for all $N \leq k \leq L$, where $N$ is an arbitrarily chosen index close to $L$, the value of $x_i^L$ approaches zero. Therefore, to satisfy condition (i), $a_i^k$ remains close to 1, and the inequality $a_i^k \leq 1$ does not hold for all $N \leq k \leq L$.

**Proposed Weight Initialization** The proposed initial weight matrix is defined as $\mathbf{W}^\ell = \mathbf{D}^\ell + \mathbf{Z}^\ell \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$, where $\mathbf{D}_{i,j}^\ell = 1$ if $i \equiv j \pmod{N_{\ell-1}}$, 0 otherwise (Examples of $\mathbf{D}^\ell$ are provided in Appendix D). The noise matrix $\mathbf{Z}^\ell$ is drawn from $\mathcal{N}(0, \sigma_z^2)$, where $\sigma_z$ is set to $\alpha/\sqrt{N^{\ell-1}}$ with $\alpha = 0.085$. Then $a_i^{k+1}$ follows the distribution:

$$a_i^{k+1} \sim \mathcal{N}\left(1,\, \sigma_z^2 + \sigma_z^2 \sum_{\substack{j=1\\ j\neq i}}^{n} \left(\frac{x_j^k}{x_i^k}\right)^2\right). \tag{2}$$

According to Equation 2, $a_i^{k+1}$ follows a Gaussian distribution with a mean of 1. Additionally, if $x_i^k$ becomes small relative to other elements in $\boldsymbol{x}^k$, the variance of the distribution 2 increases. Consequently, the probability that the absolute value of $x_i^{k+1}$ exceeds that of $x_i^k$ becomes higher. Figure 1 (b) shows that activation values maintain consistent scales in deeper layers.

### 3.3 Preventing activation saturation via appropriate $\sigma_z$ tuning

In this section, we determine the appropriate value of $\alpha$ in $\sigma_z = \alpha/\sqrt{N_{\ell-1}}$ that satisfies condition (ii). Condition (ii) is motivated by normalization methods (Ioffe, 2015; Ba, 2016). Firstly, we experimentally investigated the impact of $\sigma_z$ on the scale of the activation values. As shown in

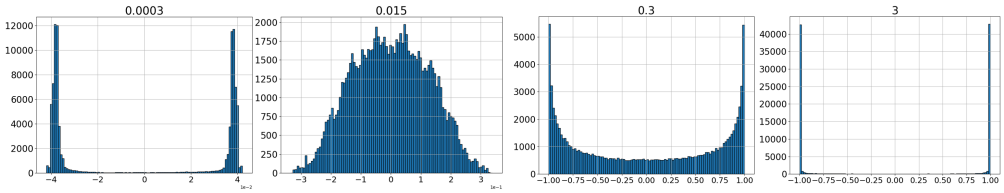



Figure 3: The activation values in the 1000th layer, with 32 nodes per hidden layer, were analyzed using the proposed weight initialization method with $\sigma_z$ values of 0.0003, 0.015, 0.3, and 3. The analysis was conducted on 3000 input samples uniformly distributed within the range $[-1, 1]$.

Figure 2, increasing $\sigma_z = \alpha/\sqrt{N_{\ell-1}}$ broadens the activation range in each layer, while decreasing $\sigma_z$ narrows it.

**When $\sigma_z$ is Large** Setting $\sigma_z$ to a large value can lead to saturation. If $\sigma_z$ is too large, Equation (2) implies that the likelihood of $a_i^k$ deviating significantly from 1 increases. This increases the likelihood of activation values being bounded by $\xi_{1+\epsilon}$ in sufficiently deep layers, as stated in Corollary 4. Consequently, in deeper layers, activation values are less likely to approach zero and tend to saturate toward specific values. Please refer to the Figure 3 for the cases where $\sigma_z = 0.3$ and 3.

**When $\sigma_z$ is Small** If $\sigma_z$ is too small, Equation 2 implies that the distribution of $a_i^k$ has a standard deviation close to zero. Consequently, $x_i^{k+1}$ can be approximated as the result of applying $\tanh(x)$ to $x_i > 0$ repeatedly for a finite number of iterations, $k$. Since $\tanh'(x)$ decreases for $x \geq 0$, the values resulting from finite iterations eventually saturate. Plese refer to the Figure 3 when $\sigma_z = 0.0003$.

For these reasons, we experimentally determined an optimal $\sigma_z$ that balances between being too large or too small. As shown in Figure 3, $\sigma_z = 0.015$ maintains an approximately normal activation distribution without collapse. Additional experimental results are provided in Appendix B.1. Considering the number of hidden layer nodes, we set $\sigma_z = \alpha/\sqrt{N^{\ell-1}}$ with $\alpha = 0.085$. Experimental results for solving the Burgers' equation using PINNs with varying $\sigma_z$ are provided in Appendix C.3.

## 4 Experiments

In this section, we conduct a series of experiments to validate the proposed weight initialization method. In Section 4.1, we evaluate the performance of FFNNs with tanh activation function on benchmark datasets. In Section 4.2, we solve PDEs using Physics-Informed Neural Networks (PINNs). Both experiments evaluated the proposed method's robustness to network size and dataset efficiency.

Table 1: Validation accuracy and loss are presented for FFNNs with varying numbers of nodes $(2, 8, 32, 128, 512)$, each with 20 hidden layers using tanh activation function. All models were trained for 20 epochs, and the highest average accuracy and lowest average loss, computed from 10 runs, are presented. The better-performing method is highlighted in bold when comparing different initialization methods under the same experimental settings.

| Dataset | Method | 2 Nodes | | 8 Nodes | | 32 Nodes | | 128 Nodes | | 512 Nodes | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Accuracy | Loss | Accuracy | Loss | Accuracy | Loss | Accuracy | Loss | Accuracy | Loss |
| MNIST | Xavier | 49.78 | 1.632 | 68 | 0.958 | 91.67 | 0.277 | 95.45 | 0.154 | 97.35 | 0.087 |
| | Proposed | **62.82** | **1.185** | **77.95** | **0.706** | **92.51** | **0.255** | **96.12** | **0.134** | **97.96** | **0.067** |
| FMNIST | Xavier | 42.89 | 1.559 | 68.55 | 0.890 | 81.03 | 0.533 | 86.20 | 0.389 | 88.28 | 0.331 |
| | Proposed | **51.65** | **1.324** | **71.31** | **0.777** | **83.06** | **0.475** | **87.12** | **0.359** | **88.59** | **0.323** |
| CIFAR-10 | Xavier | 32.82 | 1.921 | 43.51 | 1.608 | 48.62 | 1.473 | 47.58 | 1.510 | 51.71 | 1.369 |
| | Proposed | **38.16** | **1.780** | **47.04** | **1.505** | **48.80** | **1.463** | **48.51** | **1.471** | **52.21** | **1.359** |
| CIFAR-100 | Xavier | 10.87 | 4.065 | 18.53 | 3.619 | 23.71 | 3.301 | 23.83 | 3.324 | 17.72 | 3.672 |
| | Proposed | **15.22** | **3.818** | **23.07** | **3.350** | **24.93** | **3.237** | **24.91** | **3.240** | **22.80** | **3.435** |

Table 2: Validation accuracy and loss are presented for FFNNs with varying numbers of layers $(3, 10, 50, 100)$, each with $64$ number of nodes using the tanh activation function. All models were trained for $40$ epochs, and the highest average accuracy and lowest average loss, computed from $10$ runs, are presented.

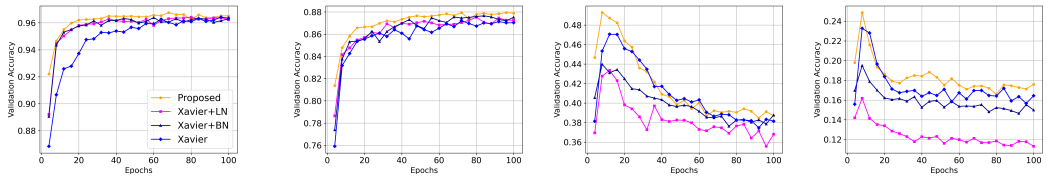
| Dataset | Method | 3 Layers | | 10 Layers | | 50 Layers | | 100 Layers | |
|---|---|---|---|---|---|---|---|---|---|
| | | Accuracy | Loss | Accuracy | Loss | Accuracy | Loss | Accuracy | Loss |
| MNIST | Xavier | 95.98 | 0.130 | 96.55 | 0.112 | 96.57 | 0.123 | 94.08 | 0.194 |
| | Proposed | **96.32** | **0.123** | **97.04** | **0.102** | **96.72** | **0.109** | **96.06** | **0.132** |
| FMNIST | Xavier | 85.91 | 0.401 | 88.73 | 0.319 | 87.72 | 0.344 | 83.41 | 0.463 |
| | Proposed | **86.51** | **0.379** | **89.42** | **0.305** | **88.51** | **0.324** | **86.01** | **0.382** |
| CIFAR-10 | Xavier | 42.91 | 1.643 | 48.39 | 1.468 | 47.87 | 1.474 | 46.71 | 1.503 |
| | Proposed | **45.05** | **1.588** | **48.41** | **1.458** | **48.71** | **1.461** | **48.96** | **1.437** |
| CIFAR-100 | Xavier | 19.10 | 3.628 | 22.73 | 3.400 | 24.27 | 3.283 | 20.32 | 3.515 |
| | Proposed | **19.30** | **3.609** | **23.83** | **3.309** | **25.07** | **3.190** | **24.41** | **3.234** |

### 4.1 Classification Task

**Experimental Setting** To evaluate the effectiveness of the proposed weight initialization method, we conduct experiments on the MNIST, Fashion MNIST (FMNIST), CIFAR-10, and CIFAR-100 (Krizhevsky & Hinton, 2009) datasets with the Adam optimizer. All experiments are conducted with a batch size of $64$ and a learning rate of $0.0001$. Fifteen percent of the total dataset is allocated for validation. The experiments were conducted in TensorFlow without skip connections and learning rate decay in any of the experiments.

**Width Independence in Classification Task** We evaluate the proposed weight initialization method in training tanh FFNNs, focusing on its robustness to variations in network width. Five tanh FFNNs are designed, each with 20 hidden layers, and with $2, 8, 32$, $128$, and $512$ nodes per hidden layer, respectively. In Table 1, for both the MNIST, Fashion MNIST and CIFAR-10 datasets, the network with $512$ nodes achieves the highest accuracy and lowest loss when the proposed method is employed. However, for the CIFAR-100 dataset, the network with 32 nodes yields the highest accuracy and lowest loss when employing the proposed method. In summary, the Proposed method demonstrates robustness to variations in the number of nodes in tanh FFNNs. Detailed experimental results are provided in Appendix B.2.

**Depth Independence in Classification Task** The expressivity of neural networks is known to increase exponentially with depth, enabling strong generalization performance (Poole et al., 2016; Raghu et al., 2017). To evaluate the robustness of the proposed weight initialization method to variations in network depth, we conduct experiments on deep FFNNs with tanh activation functions. Specifically, we construct four tanh FFNNs, each with $64$ nodes per hidden layer and $3$, $10$, $50$, or $100$ hidden layers. respectively. In Table 2, for both the MNIST and Fashion MNIST datasets, the network with 10 hidden layers achieves the highest accuracy and lowest loss when our proposed method is employed. Both initialization methods showed lower performance in networks with 3 layers compared to those with more layers. Moreover, for complex datasets like CIFAR-10 and CIFAR-100, the proposed method demonstrated improved performance when training deeper networks.




(a) MNIST (b) Fashion MNIST (c) CIFAR10 (d) CIFAR100

Figure 4: Validation accuracy for a tanh FFNN with 50 hidden layers (32 nodes each). Xavier + BN and Xavier + LN represent Xavier initialization with Batch Normalization or Layer Normalization applied every 5 layers, respectively.

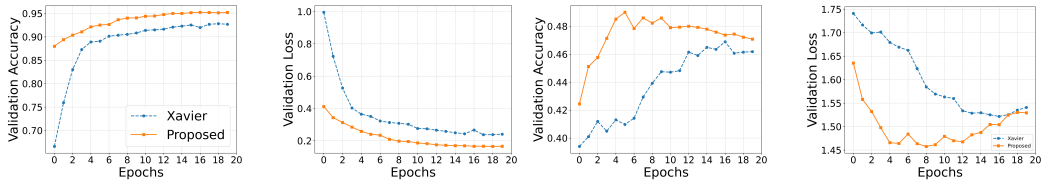
Table 3: Validation accuracy and loss for a 10-layer FFNN (64 nodes per layer) trained on datasets of sizes 10, 20, 30, 50, and 100. Results show the highest average accuracy and lowest average loss over 5 runs after 100 epochs.

| Dataset | Method | 10 | | 20 | | 30 | | 50 | | 100 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Accuracy | Loss | Accuracy | Loss | Accuracy | Loss | Accuracy | Loss | Accuracy | Loss |
| MNIST | Xavier | 31.13 | 2.281 | 35.03 | 2.078 | 45.05 | 1.771 | 58.45 | 1.227 | 64.02 | 1.139 |
| | Xavier + BN | 22.46 | 2.267 | 33.73 | 2.053 | 37.13 | 2.042 | 39.78 | 1.944 | 57.51 | 1.464 |
| | Xavier + LN | 28.52 | 2.411 | 41.54 | 1.796 | 41.94 | 1.886 | 54.97 | 1.362 | 65.11 | 1.093 |
| | Proposed | **37.32** | **2.204** | **46.79** | **1.656** | **48.60** | **1.645** | **61.54** | **1.131** | **68.44** | **1.043** |
| FMNIST | Xavier | 36.16 | 2.320 | 41.69 | 1.814 | 53.86 | 1.459 | 64.53 | 1.140 | 63.58 | 1.048 |
| | Xavier + BN | 35.44 | **2.136** | 38.58 | 1.925 | 40.16 | 1.819 | 53.93 | 1.728 | 59.78 | 1.237 |
| | Xavier + LN | 34.94 | 2.362 | 37.90 | 1.793 | 53.27 | 1.470 | 59.50 | 1.198 | 62.01 | 1.073 |
| | Proposed | **37.31** | 2.217 | **49.25** | **1.651** | **55.19** | **1.372** | **66.14** | **1.057** | **67.58** | **0.914** |

**Normalization Methods** Xavier initialization is known to cause vanishing gradients and activation problems in deeper networks. These issues are known to be mitigated by employing Batch Normalization (BN) or Layer Normalization (LN) in the network. Therefore, we compared the proposed method with Xavier, Xavier with BN, and Xavier with LN. To validate the effectiveness of normalization, we conducted experiments using a sufficiently deep neural network with 50 hidden layers. As shown in Figure 4, for datasets with relatively fewer features, such as MNIST and FMNIST, Xavier with normalization converges faster than Xavier. However, for feature-rich datasets like CIFAR-10 and CIFAR-100, the accuracy of Xavier with normalization is lower than that of Xavier. Normalization typically incurs a 30% computational overhead, and additional hyperparameter tuning is required to determine which layers should apply normalization. In contrast, the proposed method achieves the best performance among all four approaches across all datasets, without the need for normalization.

**Dataset Efficiency in Classification Task** Based on the results in Table 2, we evaluated data efficiency on a network with 50 hidden layers, each containing 64 nodes, where Xavier showed strong performance. As shown in Table 3, the highest average accuracy and lowest average loss over 5 runs after 100 epochs are presented for datasets containing 10, 20, 30, 50, and 100 samples. The proposed method achieved the best performance across all sample sizes.

**Non-uniform Hidden Layer Dimensions** We validate the performance of the proposed initialization in networks where hidden layer dimensions are not uniform. As shown in Figure 5, the network consists of 60 hidden layers, where the number of nodes alternates between 32 and 16 in each layer. We demonstrate improved performance in terms of both loss and accuracy across all epochs on the MNIST and CIFAR-10 datasets. Additionally, Appendix B.3 presents experiments on networks with larger differences in the number of nodes. Motivated by these results, Appendix B.3 explores autoencoders with significant variations in hidden layer dimensions.




(a) MNIST Accuracy (b) MNIST Loss (c) CIFAR10 Accuracy (d) CIFAR10 Loss

Figure 5: Validation accuracy and loss for a tanh FFNN with 60 hidden layers, where the number of nodes alternates between 32 and 16 across layers, repeated 30 times. The model was trained for 20 epochs on the MNIST and CIFAR-10 datasets.

## 4.2 Physics-Informed Neural Networks

Xavier initialization is the most commonly employed method for training PINNs (Jin et al., 2021; Son et al., 2023; Yao et al., 2023; Gnanasambandam et al., 2023). In this section, we experimentally

show that the proposed method is more robust to variations in network size and achieves better data efficiency compared to Xavier initialization with or without normalization methods.

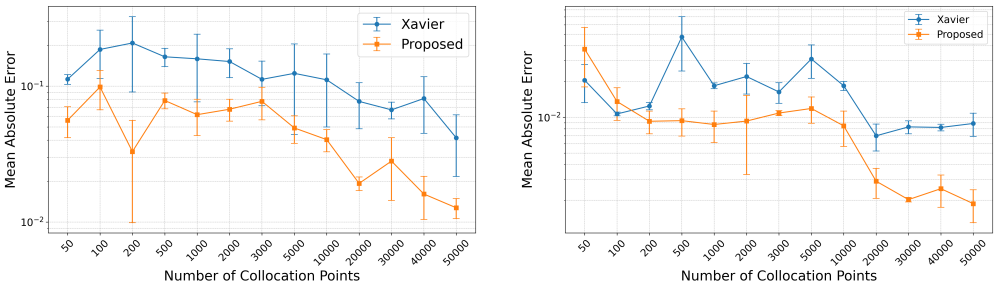
**Experimental Setting** All experiments on Physics-Informed Neural Networks (PINNs) use full-batch training with a learning rate of 0.001. In this section, we solve the Allen-Cahn, Burgers, Diffusion, and Poisson equations using a tanh FFNN-based PINN with 20,000 collocation points. Details on the PDEs are provided in Appendix C.1.

**Network Size Independence in PINNs** We construct eight tanh FFNNs, each with 16 nodes per hidden layer and $5, 10, 20, 30, 40, 50, 60,$ or $80$ hidden layers. As shown in Table 4, for the Allen-Cahn and Burgers' equations, Xavier+BN and Xavier+LN achieve the lowest loss at a network depth of 30. However, their loss gradually increases as the depth grows. In contrast, the proposed method achieves the lowest loss at depths of 50 and 60, respectively, maintaining strong learning performance even in deeper networks. For the Diffusion and Poisson equations, Xavier+LN achieves the lowest loss at depths of 5 and 10, respectively. While all methods show increasing loss as network depth increases, the proposed method consistently maintains lower loss in deeper networks. Similar results are observed with 32 nodes, double the previous size. Across all tested network sizes and PDEs, the proposed method consistently achieves the lowest loss.

Table 4: A PINN loss is presented for FFNNs with varying numbers of layers $(5, 10, 20, 30, 40, 50, 60, 80)$ using the tanh activation function. The top table shows results with 16 nodes per layer, and the bottom table shows results with 32 nodes per layer. All models were trained for 300 iterations using Adam and 300 iterations using L-BFGS. The median PINN loss from the final iteration for the Burgers, Allen–Cahn, Diffusion, and Poisson equations, computed over 5 runs, is presented.

| **Allen-Cahn (16 Nodes)** | 5 | 10 | 20 | 30 | 40 | 50 | 60 | 80 |
|---|---|---|---|---|---|---|---|---|
| Xavier | 9.58e-04 | 8.16e-04 | 7.61e-04 | 1.06e-03 | 1.1e-03 | 1.24e-03 | 3.55e-03 | 1.81e-03 |
| Xavier + BN | 1.42e-03 | 8.17e-04 | 8.56e-04 | 7.07e-04 | 7.77e-04 | 8.87e-04 | 9.11e-04 | 2.15e-03 |
| Xavier + LN | 6.29e-01 | 1.77e-03 | 6.98e-04 | 1.27e-03 | 1.82e-03 | 6.65e-01 | 3.29e-01 | 5.86e-01 |
| Proposed | **9.21e-04** | **7.29e-04** | **5.76e-04** | **5.29e-04** | **5.37e-04** | **4.03e-04** | **4.73e-04** | **5.77e-04** |
| **Burgers (16 Nodes)** | 5 | 10 | 20 | 30 | 40 | 50 | 60 | 80 |
| Xavier | 6.97e-03 | 1.11e-02 | 7.9e-03 | 9.71e-03 | 2.45e-02 | 2.65e-02 | 6.5e-02 | 5.71e-02 |
| Xavier + BN | 8.07e-03 | 7.72e-03 | 6.24e-03 | 1.70e-02 | 1.50e-02 | 1.85e-02 | 2.91e-02 | 6.84e-02 |
| Xavier + LN | 3.89e-02 | 1.88e-02 | 9.48e-03 | 9.28e-03 | 2.46e-02 | 3.30e-02 | 6.91e-02 | 4.42e-02 |
| Proposed | **6.19e-03** | **5.08e-03** | **5.28e-03** | **9.31e-04** | **3.56e-03** | **8.27e-04** | **3.43e-04** | **2.05e-03** |
| **Diffusion (16 Nodes)** | 5 | 10 | 20 | 30 | 40 | 50 | 60 | 80 |
| Xavier | 2.52e-03 | 4.82e-03 | 9.69e-03 | 1.33e-02 | 2.08e-02 | 1.50e-02 | 2.92e-02 | 7.24e-02 |
| Xavier + BN | 2.89e-03 | 5.77e-03 | 1.05e-02 | 9.65e-03 | 2.76e-02 | 1.07e-02 | 9.07e-03 | 1.43e-02 |
| Xavier + LN | 1.72e-03 | 6.10e-03 | 8.04e-03 | 9.48e-03 | 2.14e-02 | 7.59e-03 | 2.05e-02 | 2.21e-02 |
| Proposed | **9.14e-04** | **2.59e-03** | **2.40e-03** | **1.01e-03** | **1.97e-03** | **1.21e-03** | **1.12e-03** | **1.91e-03** |
| **Poisson (16 Nodes)** | 5 | 10 | 20 | 30 | 40 | 50 | 60 | 80 |
| Xavier | 1.52e-02 | 2.87e-02 | 1.28e-01 | 9.82e-02 | 1.15e-01 | 1.37e-01 | 1.82e-01 | 2.55e-01 |
| Xavier + BN | 1.62e-02 | 2.02e-02 | 8.72e-02 | 1.12e-01 | 2.45e-01 | 9.85e-02 | 1.00e-01 | 1.34e-01 |
| Xavier + LN | 5.39e-01 | 4.40e-02 | 1.34e-01 | 3.91 | 2.52e+02 | 2.58 | 9.79e+02 | nan |
| Proposed | **1.37e-02** | **1.70e-02** | **4.62e-02** | **2.43e-02** | **3.75e-02** | **4.03e-02** | **6.07e-02** | **6.01e-02** |

| **Allen-Cahn (32 Nodes)** | 5 | 10 | 20 | 30 | 40 | 50 | 60 | 80 |
|---|---|---|---|---|---|---|---|---|
| Xavier | 3.13e-01 | 5.03e-02 | 3.64e-03 | 2.37e-03 | 4.03e-03 | 5.27e-03 | 1.73e-02 | 6.94e-01 |
| Xavier + BN | 4.05e-01 | 8.85e-04 | 8.41e-04 | 7.82e-04 | 9.97e-04 | 6.80e-04 | 9.34e-04 | 6.94e-01 |
| Xavier + LN | 3.31e-01 | 2.10e-03 | 5.99e-04 | 6.71e-04 | 1.49e-03 | 1.29e-03 | 3.31e-02 | 6.93e-01 |
| Proposed | **1.04e-03** | **6.92e-04** | **5.34e-04** | **4.26e-04** | **3.31e-04** | **3.52e-04** | **3.85e-04** | **5.96e-04** |
| **Burgers (32 Nodes)** | 5 | 10 | 20 | 30 | 40 | 50 | 60 | 80 |
| Xavier | 1.12e-02 | 3.53e-03 | 2.72e-03 | 1.81e-03 | 7.60e-03 | 8.56e-03 | 9.86e-03 | 1.66e-01 |
| Xavier + BN | 5.88e-03 | **1.04e-03** | 1.79e-03 | 2.80e-03 | 5.95e-03 | 3.66e-02 | 6.60e-02 | 1.66e-01 |
| Xavier + LN | 4.31e-02 | 1.21e-02 | 1.88e-03 | 7.22e-03 | 5.54e-03 | 8.46e-03 | 9.04e-03 | 4.86e-02 |
| Proposed | **4.14e-03** | 4.11e-03 | **1.58e-03** | **1.29e-03** | **7.96e-04** | **5.85e-04** | **9.80e-04** | **1.47e-03** |
| **Diffusion (32 Nodes)** | 5 | 10 | 20 | 30 | 40 | 50 | 60 | 80 |
| Xavier | 1.69e-03 | 6.85e-03 | 7.63e-03 | 4.50e-03 | 8.98e-03 | 5.67e-03 | 6.33e-01 | 1.59 |
| Xavier + BN | 1.68e-03 | 2.66e-03 | 1.08e-02 | 6.00e-03 | 8.58e-03 | 6.60e-03 | 5.66e-02 | 1.69e+02 |
| Xavier + LN | 8.16e-04 | 2.85e-03 | 8.46e-03 | 4.57e-03 | 9.40e-03 | 1.04e-02 | 2.42e-01 | 1.67e+02 |
| Proposed | **2.89e-04** | **8.03e-04** | **5.25e-04** | **5.07e-04** | **5.33e-04** | **6.17e-04** | **9.80e-04** | **1.53e-03** |
| **Poisson (32 Nodes)** | 5 | 10 | 20 | 30 | 40 | 50 | 60 | 80 |
| Xavier | 1.09e-02 | 1.33e-02 | 3.13e-02 | 7.69e-02 | 6.72e-02 | 8.90e-02 | 9.68e+02 | 1.46e+02 |
| Xavier + BN | 1.14e-02 | 1.47e-02 | 2.68e-02 | 3.55e-02 | 8.25e-02 | 8.97e-02 | 4.50e-02 | 7.75e-01 |
| Xavier + LN | 2.36e-02 | 2.18e-02 | 3.07e-02 | 3.85e-01 | 1.40 | 4.69 | 2.60 | 6.14 |
| Proposed | **9.63e-03** | **8.29e-03** | **1.41e-02** | **1.88e-02** | **1.65e-02** | **1.85e-02** | **1.73e-02** | **3.59e-02** |




(a) Burgers Equation (b) Diffusion Equation

Figure 6: Mean absolute error between the exact solution and PINN-predicted solution with varying numbers of collocation points. The FFNN has 30 hidden layers (32 nodes each) and is trained for 300 iterations using Adam followed by 300 iterations using L-BFGS. The results are averaged over 5 experiments.

**Data Efficiency** Based on the results in Table 4, we evaluated data efficiency on a network with 30 hidden layers, each containing 32 nodes, where Xavier achieved the lowest PINN loss. As shown in Figure 6, for the Burgers equation, the Mean Absolute Error (MAE) of the proposed initialization shows a clear difference compared to Xavier initialization across varying numbers of collocation points. In contrast, for the Diffusion equation, the difference in MAE between the two methods becomes more pronounced when the number of collocation points exceeds $20,000$. Additionally, Figure 7 illustrates that increasing the number of collocation points enables PINNs with the proposed initialization to predict solutions with lower absolute error. For detailed experiments on the Burgers equation, please refer to Appendix C.4.

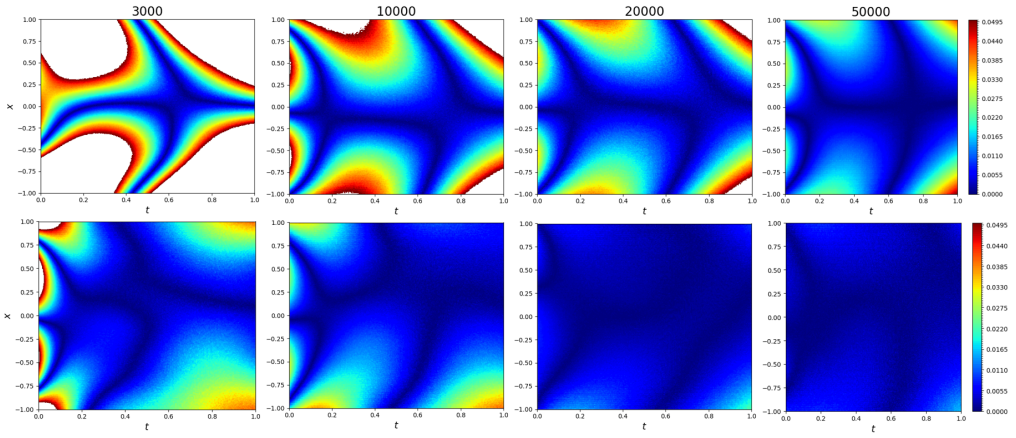



Figure 7: Absolute error between the exact solution and the PINN-predicted solution for the Diffusion equation with varying numbers of collocation points $(3000, 10000, 20000, 50000)$ using **(upper row)** Xavier and **(lower row)** the proposed initialization. The FFNN has 30 hidden layers (32 nodes each) and is trained for 300 iterations using Adam followed by 300 iterations using L-BFGS. The color bar ranges from 0 to 0.05, with values outside this range shown in white.

## 5 Conclusion

In this study, we proposed a novel weight initialization method for tanh neural networks, grounded in the theoretical analysis of fixed points of the $\tanh(ax)$ function. The proposed method is experimentally demonstrated to achieve robustness to variations in network size without normalization methods and to exhibit improved data efficiency. Therefore, the proposed weight initialization method reduces the time and effort required for training on large datasets and optimizing network architectures.

## REFERENCES


Jimmy Lei Ba. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Thomas Bachlechner, Bodhisattwa Prasad Majumder, Henry Mao, Gary Cottrell, and Julian McAuley. Rezero is all you need: fast convergence at large depth. In Cassio de Campos and Marloes H. Maathuis (eds.), *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161 of *Proceedings of Machine Learning Research*, pp. 1352–1361. PMLR, 27–30 Jul 2021.

Hassan Bararnia and Mehdi Esmaeilpour. On the application of physics informed neural networks (pinn) to solve boundary layer thermal-fluid problems. *International Communications in Heat and Mass Transfer*, 132:105890, 2022. ISSN 0735-1933. doi: https://doi.org/10.1016/j.icheatmasstransfer.2022.105890.

Yoshua Bengio, Paolo Frasconi, and Patrice Simard. The problem of learning long-term dependencies in recurrent networks. In *IEEE international conference on neural networks*, pp. 1183–1188. IEEE, 1993.

Salvatore Cuomo, Vincenzo Schiano Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli. Scientific machine learning through physics–informed neural networks: Where we are and what's next. *Journal of Scientific Computing*, 92(3):88, 2022a. doi: https://doi.org/10.1007/s10915-022-01939-z.

Salvatore Cuomo, Vincenzo Schiano Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli. Scientific machine learning through physics–informed neural networks: Where we are and what's next. *Journal of Scientific Computing*, 92(3):88, 2022b.

G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989. URL `https://doi.org/10.1007/BF02551274`.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.

Raghav Gnanasambandam, Bo Shen, Jihoon Chung, Xubo Yue, and Zhenyu Kong. Self-scalable tanh (stan): Multi-scale solutions for physics-informed neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

G. Gripenberg. Approximation by neural networks with a bounded number of nodes at each level. *Journal of Approximation Theory*, 122(2):260–266, 2003. ISSN 0021-9045. doi: https://doi.org/10.1016/S0021-9045(03)00078-9.

Namig J. Guliyev and Vugar E. Ismailov. Approximation capability of two hidden layer feedforward neural networks with fixed weights. *Neurocomputing*, 316:262–269, 2018a. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2018.07.075.

Namig J. Guliyev and Vugar E. Ismailov. On the approximation by single hidden layer feedforward neural networks with fixed weights. *Neural Networks*, 98:296–304, 2018b. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2017.12.007.

John M. Hanna, José V. Aguado, Sebastien Comas-Cardona, Ramzi Askri, and Domenico Borzacchiello. Residual-based adaptivity for two-phase flow simulation in porous media using physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 396:115100, 2022. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2022.115100.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4 (2):251–257, 1991. ISSN 0893-6080. doi: https://doi.org/10.1016/0893-6080(91)90009-T.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. ISSN 0893-6080. doi: https://doi.org/10.1016/0893-6080(89)90020-8.

Vahid Reza Hosseini, Abbasali Abouei Mehrizi, Afsin Gungor, and Hamid Hassanzadeh Afrouzi. Application of a physics-informed neural network to solve the steady-state bratu equation arising from solid biofuel combustion theory. *Fuel*, 332:125908, 2023. ISSN 0016-2361. doi: https://doi.org/10.1016/j.fuel.2022.125908.

Sergey Ioffe. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Ameya D. Jagtap, Ehsan Kharazmi, and George Em Karniadakis. 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. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2020.113028. URL https://www.sciencedirect.com/science/article/pii/S0045782520302127.

Ameya D Jagtap, Zhiping Mao, Nikolaus Adams, and George Em Karniadakis. Physics-informed neural networks for inverse problems in supersonic flows. *Journal of Computational Physics*, 466:111402, 2022.

Ameya Dilip Jagtap and George E. Karniadakis. Extended physics-informed neural networks (xpinns): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. *Communications in Computational Physics*, 2020. URL https://api.semanticscholar.org/CorpusID:229083388.

Xiaowei Jin, Shengze Cai, Hui Li, and George Em Karniadakis. Nsfnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations. *Journal of Computational Physics*, 426:109951, 2021.

George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 2021.

Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

Hyunwoo Lee, Yunho Kim, Seung Yeop Yang, and Hayoung Choi. Improved weight initialization for deep and narrow feedforward neural network. *Neural Networks*, 176:106362, 2024.

Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

Xu Liu, Xiaoya Zhang, Wei Peng, Weien Zhou, and Wen Yao. A novel meta-learning initialization method for physics-informed neural networks. *Neural Computing and Applications*, 34(17): 14511–14534, 2022.

Lu Lu, Yeonjong Shin, Yanhui Su, and George Em Karniadakis. Dying relu and initialization: Theory and numerical examples. *arXiv preprint arXiv:1903.06733*, 2019.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 2021.

Vitaly Maiorov and Allan Pinkus. Lower bounds for approximation by mlp neural networks. *Neurocomputing*, 25(1):81–91, 1999. ISSN 0925-2312. doi: https://doi.org/10.1016/S0925-2312(98)00111-8.

Zhiping Mao, Ameya D. Jagtap, and George Em Karniadakis. Physics-informed neural networks for high-speed flows. *Computer Methods in Applied Mechanics and Engineering*, 360:112789, 2020. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2019.112789.

Dmytro Mishkin and Jiri Matas. All you need is a good init. In *ICLR*, 2016.

Sejun Park, Chulhee Yun, Jaeho Lee, and Jinwoo Shin. Minimum width for universal approximation. *CoRR*, abs/2006.08859, 2020. URL https://arxiv.org/abs/2006.08859.

Pai Peng, Jiangong Pan, Hui Xu, and Xinlong Feng. Rpinns: Rectified-physics informed neural networks for solving stationary partial differential equations. *Computers & Fluids*, 245:105583, 2022. ISSN 0045-7930. doi: https://doi.org/10.1016/j.compfluid.2022.105583.

Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. *Advances in neural information processing systems*, 29, 2016.

Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. In *ICML*, pp. 2847–2854, 2017.

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

Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

Pratik Rathore, Weimu Lei, Zachary Frangella, Lu Lu, and Madeleine Udell. Challenges in training PINNs: A loss landscape perspective. In *ICML*, 2024.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Andrew M Saxe, James McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *ICLR*, 2014.

Zuowei Shen, Haizhao Yang, and Shijun Zhang. Optimal approximation rate of relu networks in terms of width and depth. *Journal de Mathématiques Pures et Appliquées*, 157:101–135, 2022. ISSN 0021-7824. doi: https://doi.org/10.1016/j.matpur.2021.07.009.

Khemraj Shukla, Patricio Clark Di Leoni, James Blackshire, Daniel Sparkman, and George Em Karniadakis. Physics-informed neural network for ultrasound nondestructive quantification of surface breaking cracks. *Journal of Nondestructive Evaluation*, 39(3):61, 2020. doi: 10.1007/s10921-020-00705-1. URL https://doi.org/10.1007/s10921-020-00705-1.

Khemraj Shukla, Ameya D. Jagtap, and George Em Karniadakis. Parallel physics-informed neural networks via domain decomposition. *Journal of Computational Physics*, 447:110683, 2021. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2021.110683. URL https://www.sciencedirect.com/science/article/pii/S0021999121005787.

Hwijae Son, Sung Woong Cho, and Hyung Ju Hwang. Enhanced physics-informed neural networks with augmented lagrangian relaxation method (al-pinns). *Neurocomputing*, 548:126424, 2023.

Yanjie Song, He Wang, He Yang, Maria Luisa Taccari, and Xiaohui Chen. Loss-attentional physics-informed neural networks. *Journal of Computational Physics*, 501:112781, 2024.

Homayoon Tarbiyati and Behzad Nemati Saray. Weight initialization algorithm for physics-informed neural networks using finite differences. *Engineering with Computers*, pp. 1–17, 2023.

Zhiyong Wu, Huan Wang, Chang He, Bing J. Zhang, Tao Xu, and Qinglin Chen. The application of physics-informed machine learning in multiphysics modeling in chemical engineering. *Industrial & Engineering Chemistry Research*, 62, 2023. ISSN 18178-18204.

Zixue Xiang, Wei Peng, Xu Liu, and Wen Yao. Self-adaptive loss balanced physics-informed neural networks. *Neurocomputing*, 496:11–34, 2022. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2022.05.015.

Liu Yang, Xuhui Meng, and George Em Karniadakis. B-pinns: Bayesian physics-informed neural networks for forward and inverse pde problems with noisy data. *Journal of Computational Physics*, 425:109913, 2021. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2020.109913.

Jiachen Yao, Chang Su, Zhongkai Hao, Songming Liu, Hang Su, and Jun Zhu. Multiadam: Parameter-wise scale-invariant optimizer for multiscale training of physics-informed neural networks. In *International Conference on Machine Learning*, pp. 39702–39721. PMLR, 2023.

Dmitry Yarotsky. Error bounds for approximations with deep relu networks. *Neural Networks*, 94: 103–114, 2017. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2017.07.002.

Minglang Yin, Xiaoning Zheng, Jay D. Humphrey, and George Em Karniadakis. Non-invasive inference of thrombus material properties with physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 375:113603, 2021. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2020.113603. URL https://www.sciencedirect.com/science/article/pii/S004578252030788X.

Jeremy Yu, Lu Lu, Xuhui Meng, and George Em Karniadakis. Gradient-enhanced physics-informed neural networks for forward and inverse pde problems. *Computer Methods in Applied Mechanics and Engineering*, 393:114823, 2022. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2022.114823. URL https://www.sciencedirect.com/science/article/pii/S0045782522001438.

Lei Yuan, Yi-Qing Ni, Xiang-Yun Deng, and Shuo Hao. A-pinn: Auxiliary physics informed neural networks for forward and inverse problems of nonlinear integro-differential equations. *Journal of Computational Physics*, 462:111260, 2022. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2022.111260.

Jiawei Zhao, Florian Tobias Schaefer, and Anima Anandkumar. Zero initialization: Initializing neural networks with only zeros and ones. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856.

Jing'ang Zhu, Yiheng Xue, and Zishun Liu. A transfer learning enhanced physics-informed neural network for parameter identification in soft materials. *Applied Mathematics and Mechanics*, 45 (10):1685–1704, 2024. doi: 10.1007/s10483-024-3178-9. URL https://doi.org/10.1007/s10483-024-3178-9.

# A PROOFS OF THE THEORETICAL RESULTS

## A.1 PROOF OF LEMMA 1

*Proof.* We define $g(x) = \tanh(ax) - x$. Since $g(x)$ is continuous, and $g(-M) > 0$, $g(M) < 0$ for a large real number $M \in \mathbb{R}^+$, the Intermediate Value Theorem guarantees the existence of a point $x$ such that $g(x) = 0$.

First, consider the case $0 < a \leq 1$. Since $0 < a \leq 1$, the derivative $g'(x) = a \cdot \text{sech}^2(ax) - 1$ satisfies $-1 \leq g'(x) \leq a - 1 < 0$ for all $x$. Hence, $g(x)$ is strictly decreasing and therefore $g(x)$ has the unique root. At $x = 0$, $\phi(0) = \tanh(a \cdot 0) = 0$. Hence, $x = 0$ is the unique fixed point.

Let us consider the case $a > 1$. For $0 < x \ll 1$, $\tanh(ax) - x \approx (a-1)x$. Since $a > 1$, $\tanh(ax) - x > 0$. On the other hand, since $|\tanh(ax)| < 1$ for all $x$,

$$\lim_{x\to\infty}[-1 - x] \leq \lim_{x\to\infty}[\tanh(ax) - x] \leq \lim_{x\to\infty}[1 - x].$$

By the squeeze theorem, $\lim_{x\to\infty}[\tanh(ax) - x] = -\infty$. By the intermediate value theorem, therefore, there exists at least one $x > 0$ such that $\tanh(ax) = x$. To establish the uniqueness of the positive fixed point, we investigate the derivative $g'(x) = a\,\text{sech}^2(ax) - 1$. We find the critical points to be $x = \pm\frac{1}{a}\sec^{-1}(\frac{1}{\sqrt{a}})$. It is straightforward to see that $g'(x) > 0$ in $\left(-\frac{1}{a}\sec^{-1}(\frac{1}{\sqrt{a}}), \frac{1}{a}\sec^{-1}(\frac{1}{\sqrt{a}})\right)$ and $g'(x) < 0$ in $\mathbb{R}\backslash\left(-\frac{1}{a}\sec^{-1}(\frac{1}{\sqrt{a}}), \frac{1}{a}\sec^{-1}(\frac{1}{\sqrt{a}})\right)$. i.e. $g(x) = 0$ has exactly two fixed points. Because $g(x)$ is an odd function, if $x^*$ is a solution, then $-x^*$ is also a solution. Thus, for $a > 1$, there exists a unique positive fixed point if $x > 0$ and a unique negative fixed point if $x < 0$. □

## A.2 PROOF OF LEMMA 2

*Proof.* (1) Since $(\tanh(ax))' = a\,\text{sech}^2(ax) < 1$ for all $x > 0$, it holds that $x_{n+1} = \phi_a(x_n) < x_n$ for all $n \in \mathbb{N}$. Thus the sequence $\{x_n\}_{n=1}^{\infty}$ is decreasing. Since $x_n > 0$ for all $n \in \mathbb{N}$, by the monotone convergence theorem, it converges to the fixed point $x^* = 0$.
(2) Let $x_0 < \xi_a$. Since $\phi'(x)$ decreasing for $x \geq 0$, with $\phi'(0) > 1$ and $\xi_a$ is the unique fixed point for $x > 0$, it holds that $x_n < x_{n+1} < \xi_a$ for all $n \in \mathbb{N}$. Thus, by the monotone convergence theorem, the sequence converges to the fixed point $\xi_a$. The proof is similar when $x_0 > \xi_a$. □

## A.3 PROOF OF PROPOSITION 3

*Proof.* Set $N = \max\{n | a_n > 1\}$. Define the sequences $\{b_n\}_{n=1}^{\infty}$ and $\{c_n\}_{n=1}^{\infty}$ such that $b_n = c_n = a_n$ for $n \leq N$, with $b_n = 0$ and $c_n = 1$ for $n > N$. Suppose that $\{\hat{\Phi}_m\}_{m=1}^{\infty}$ and $\{\tilde{\Phi}_m\}_{m=1}^{\infty}$ are sequences of functions defined as for each $m \in \mathbb{N}$

$$\hat{\Phi}_m = \phi_{b_m} \circ \phi_{b_{m-1}} \circ \cdots \circ \phi_{b_1}, \quad \tilde{\Phi}_m = \phi_{c_m} \circ \phi_{c_{m-1}} \circ \cdots \circ \phi_{c_1}.$$

Then, the inequality $\hat{\Phi}_m \leq \Phi_m \leq \tilde{\Phi}_m$ holds for all $m$. By Lemma 1, for any $x \geq 0$, we have $\lim_{m\to\infty}\hat{\Phi}_m = 0$ and $\lim_{m\to\infty}\tilde{\Phi}_m = 0$. Therefore, the Squeeze Theorem guarantees that $\lim_{m\to\infty}\Phi_m(x) = 0$. □

## A.4 PROOF OF COROLLARY 4

*Proof.* Set $N = \max\{n \mid a_n < 1 + \epsilon\}$. Define the sequence $\{b_n\}_{n=1}^{\infty}$ such that $b_n = a_n$ for $n \leq N$, and $b_n = 1 + \epsilon$ for $n > N$. The remainder of the proof is analogous to the proof of Proposition 3. □

# B CLASSIFIACATION TASKS

## B.1 ACTIVATION DISTRIBUTION FOR NORMALLY DISTRIBUTED INPUT DATA.

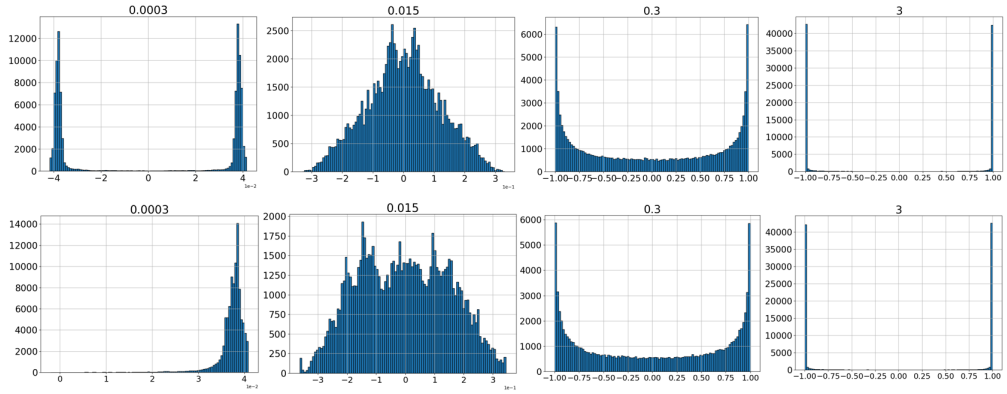



Figure 8: The activation values in the $1000^{\text{th}}$ layer, with 32 nodes per hidden layer, were analyzed using the proposed weight initialization method with $\sigma_z$ values of $0.0003$, $0.015$, $0.3$, and $3$. The **upper row** shows results for $3000$ input samples drawn from a standard normal distribution, while the **lower row** presents results for samples drawn from a Beta distribution with parameters $a = 2.0$ and $b = 5.0$.

## B.2 WIDTH INDEPENDENCE IN CLASSIFICATION TASKS

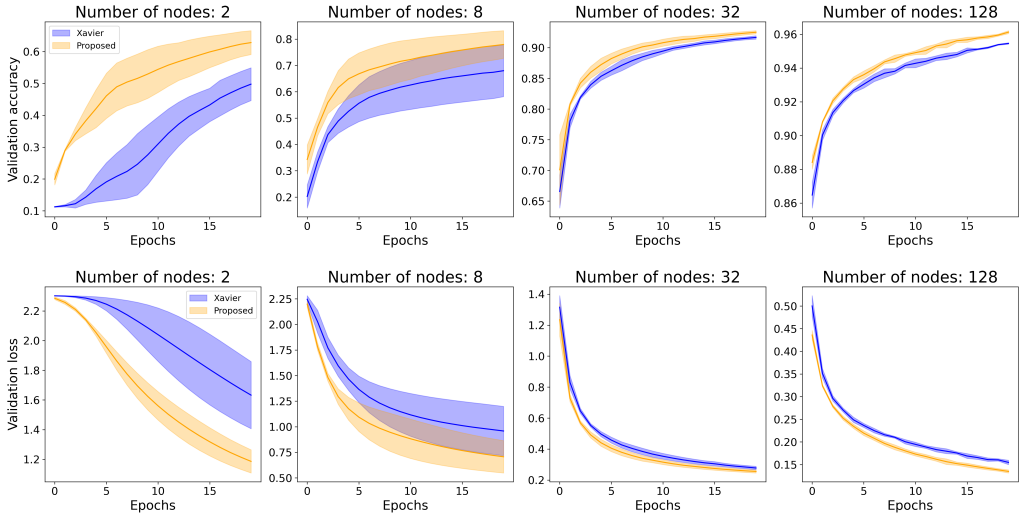



Figure 9: Validation accuracy and loss are presented for tanh FFNNs with varying numbers of nodes $(2, 8, 32, 128)$, each with 20 hidden layers. All models were trained for 20 epochs on the MNIST dataset, with 10 different random seeds.

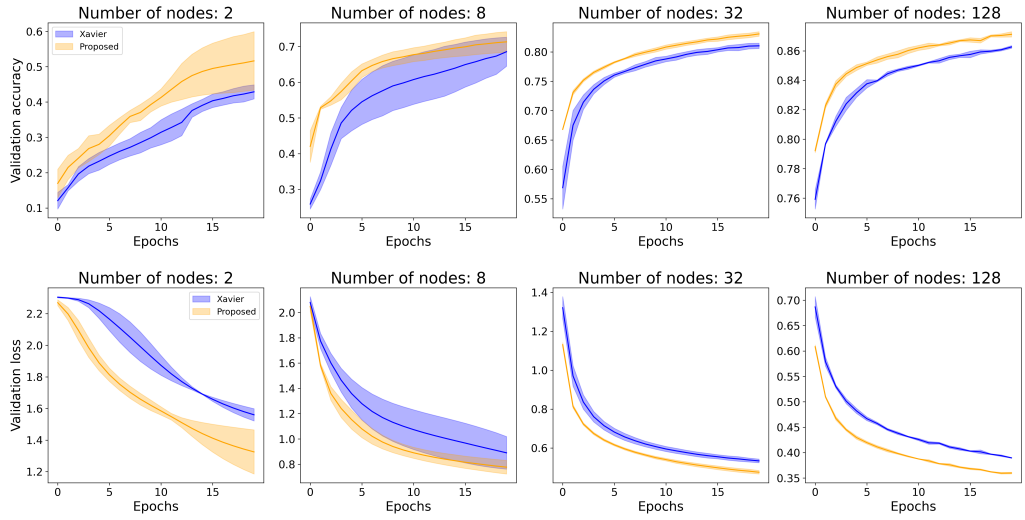



Figure 10: Validation accuracy and loss are presented for tanh FFNNs with varying numbers of nodes $(2, 8, 32, 128)$, each with 20 hidden layers. All models were trained for 20 epochs on the Fashion MNIST dataset, with 10 different random seeds.

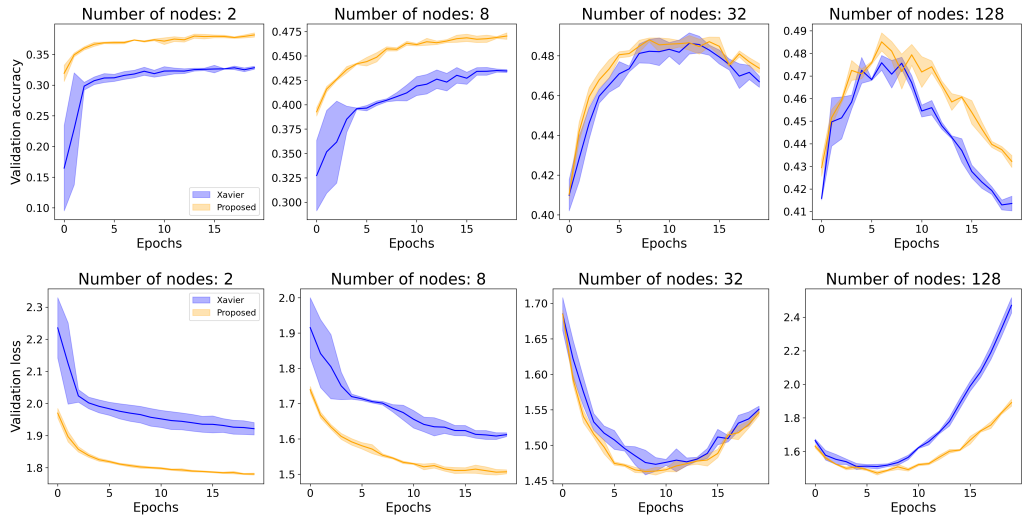



Figure 11: Validation accuracy and loss are presented for tanh FFNNs with varying numbers of nodes $(2, 8, 32, 128)$, each with 20 hidden layers. All models were trained for 20 epochs on the CIFAR-10 dataset, with 10 different random seeds.

### B.3 Non-uniform Hidden Layer Dimensions

Tanh neural networks have been less commonly used compared to ReLU networks due to higher computational complexity, the vanishing gradient problem, and ReLU's superior empirical performance in many deep learning tasks. However, recent success of PINNs with tanh neural networks has led to a resurgence in their usage. In this section, we compare the performance of four methods on architectures that are generally challenging to train: (1) tanh activation with Xavier initialization, (2) tanh activation with the proposed initialization, (3) ReLU activation with He initialization + BN, and (4) ReLU activation with orthogonal initialization.

**FFNN** The experiments were conducted on an FFNN with alternating hidden layers of 16 and 4 nodes, repeated 50 times, over 100 epochs. The results are shown in Figure 13 (a). Both Xavier and

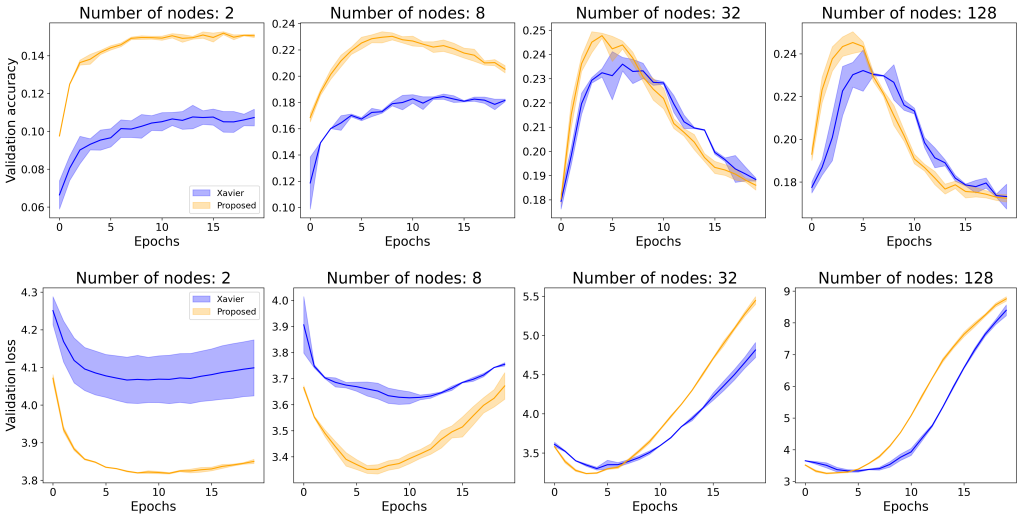
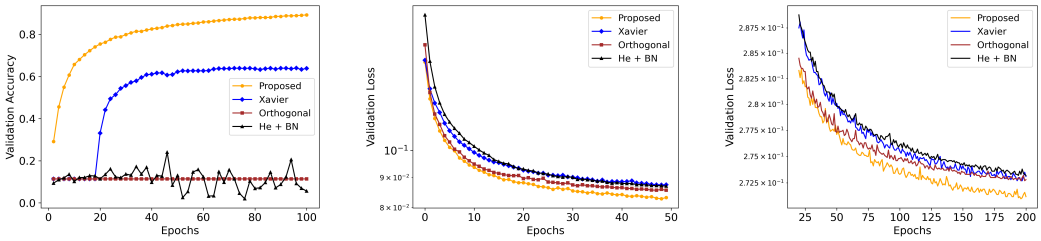



Figure 12: Validation accuracy and loss are presented for tanh FFNNs with varying numbers of nodes $(2, 8, 32, 128)$, each with 20 hidden layers. All models were trained for 20 epochs on the CIFAR-100 dataset, with 10 different random seeds.

the proposed method successfully trained the network, with the proposed method showing overall better performance. Given its strong performance despite significant differences in the number of nodes between hidden layers, we further tested the proposed method on autoencoders with large variations in layer sizes, as shown in Figure 13 (b).

**Autoencoder** The autoencoder architecture consists of an encoder and a decoder, both employing batch normalization and dropout (0.2) for regularization. The encoder compresses the input through layers of sizes 512, 256, 128, and finally maps to a latent space of 64 units. The decoder reconstructs the input by symmetrically expanding the latent space through layers of sizes 128, 256, and 512, followed by a final output layer with sigmoid activation. In Figure 13 (b), the model is trained on the MNIST dataset with a batch size of 256, while in (c), it is trained on the FMNIST dataset with a batch size of 512.




(a) FFNN (MNIST) (b) Autoencoder (MNIST) (c) Autoencoder (FMNIST)

Figure 13: **(a)** Validation loss for an FFNN with alternating hidden layers of 16 and 4 nodes, repeated 50 times, comparing four methods: Tanh with Xavier initialization, Tanh with the proposed initialization, ReLU with He initialization + BN, and ReLU with orthogonal initialization. **(b)** Validation loss for an autoencoder with encoder-decoder layers of 512, 256, 128, and 64 units, comparing the same four methods. **(c)** Same as (b), but on the FMNIST dataset.

# C PHYSICS INFORMED NEURAL NETWORKS

## C.1 PDE DETAILS

**Allen-Cahn Equation** The diffusion coefficient is set to $d = 0.01$. The initial condition is defined as $u(x, 0) = x^2 \cos(\pi x)$ for $x \in [-1, 1]$, with boundary conditions $u(-1, t) = -1$ and $u(1, t) = -1$, applied over the time interval $t \in [0, 1]$. The Allen-Cahn equation is expressed as:

$$\frac{\partial u}{\partial t} - d\frac{\partial^2 u}{\partial x^2} = -\frac{u^3 + u}{d}$$

where $u(x, t)$ represents the solution, $d$ is the diffusion coefficient, and the nonlinear term $u^3 - u$ models the phase separation dynamics.

**Burgers' Equation** The Burgers' equation, a viscosity coefficient of $\nu = 0.01$ is employed. The initial condition is given by $u(x, 0) = -\sin(\pi x)$ for $x \in [-1, 1]$, with boundary conditions $u(-1, t) = 0$ and $u(1, t) = 0$ imposed for $t \in [0, 1]$. The Burgers' equation is given by:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}$$

where $u(x, t)$ is the velocity field, and $\nu$ is the viscosity coefficient.

**Diffusion Equation** The diffusion equation includes a time-dependent source term and is defined over the spatial domain $x \in [-1, 1]$ and temporal interval $t \in [0, 1]$. The initial condition is specified as $u(x, 0) = \sin(\pi x)$, with Dirichlet boundary conditions $u(-1, t) = 0$ and $u(1, t) = 0$. The diffusion equation is expressed as:

$$\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} = e^{-t}\left(\sin(\pi x) - \pi^2 \sin(\pi x)\right),$$

where $u(x, t)$ is the solution.

**Poisson Equation** The Poisson equation is defined over the spatial domain $x \in [0, 1]$ and $y \in [0, 1]$. The Poisson equation is expressed as:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y),$$

where $u(x, y)$ is the solution, and $f(x, y)$ is the source term given by:

$$f(x, y) = 2\pi^2 \sin(\pi x)\sin(\pi y).$$

## C.2 IMPACT OF ACTIVATION FUNCTION ON PINN

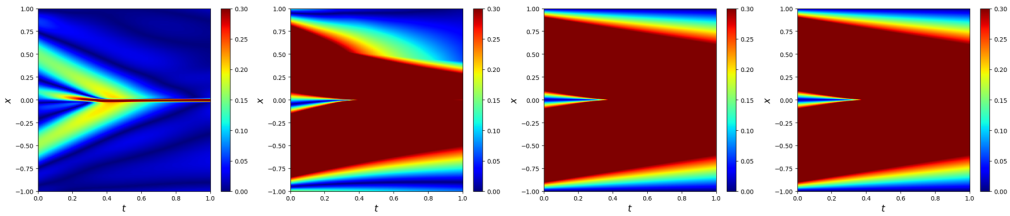
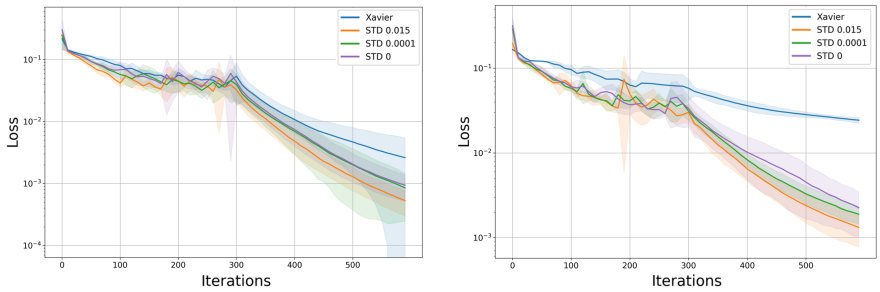


Figure 14: Absolute error for the Burgers' equation with varying activation functions (from left: tanh, ReLU, sigmoid, and swish). The FFNN has 30 hidden layers (32 nodes each) and is trained for 300 iterations using Adam followed by 300 iterations using L-BFGS.

We experimentally demonstrated that the absolute error between the exact solution and the PINN-predicted solution is smaller when using the tanh activation compared to ReLU, sigmoid, and swish activations (Ramachandran et al., 2017) in Figure 14.

### C.3 $\sigma_z$ FOR BURGERS' EQUATION




(a) Same dimension (b) Varying dimensions

Figure 15: Here, STD refers to $\sigma_z$. (a) shows the PINN loss for the Burgers' equation, using an FFNN with 30 layers and 32 nodes in each hidden layer. (b) shows the PINN loss for an FFNN with 30 layers, where the hidden layers alternate between 64 and 32 nodes, repeated 15 times. Each experiment was repeated 10 times with different random seeds.

### C.4 ABSOLUTE ERROR FOR BURGERS' EQUATION

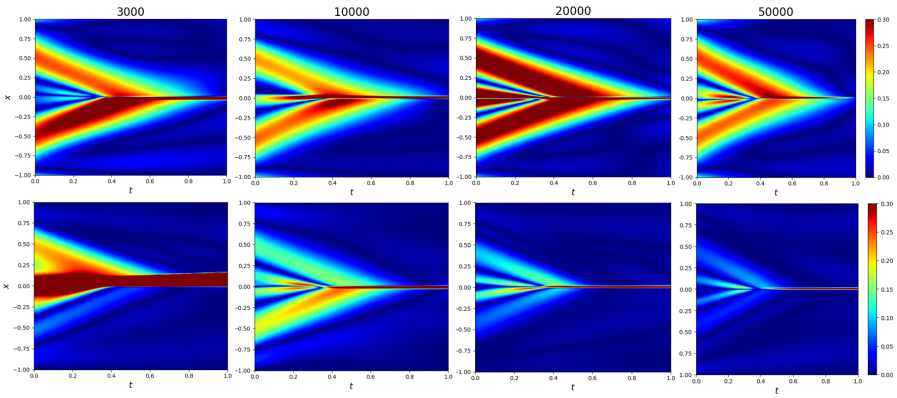



Figure 16: Absolute error between the exact solution and the PINN-predicted solution for the Burgers' equation with varying numbers of collocation points (3000, 10000, 20000, 50000) using **(upper row)** Xavier and **(lower row)** the proposed initialization. The FFNN has 30 hidden layers (32 nodes each) and is trained for 300 iterations using Adam followed by 300 iterations using L-BFGS.

## D EXAMPLES OF THE MATRIX $\mathbf{D}^\ell$

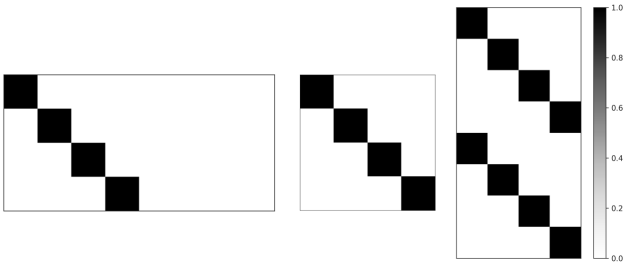


Figure 17: Examples of the matrix $\mathbf{D}^\ell \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$ in Section 3.2 (from left: $N_\ell < N_{\ell-1}$, $N_\ell = N_{\ell-1}$, $N_\ell > N_{\ell-1}$).