On the Training Efficiency of Shallow Architectures for Physics Informed Neural Networks

J Rishi, Azhar Gafoor, Sumanth Kumar, and Deepak Subramani

Department of Computational and Data Sciences, Indian Institute of Science - Bengaluru, India
{rishij,azharctp,ksumanth,deepakns}@iisc.ac.in

Abstract. Physics-informed Neural Networks (PINNs), a class of neural models that are trained by minimizing a combination of the residual of the governing partial differential equation and the initial and boundary data, have gained immense popularity in the natural and engineering sciences. Despite their observed empirical success, an analysis of the training efficiency of residual-driven PINNs at different architecture depths is poorly documented. Usually, neural models used for machine learning tasks such as computer vision and natural language processing have deep architectures, that is, a larger number of hidden layers. In PINNs, we show that for a given trainable parameter count (model size), a shallow network (less layers) converges faster than a deep network (more layers) for the same error characteristics. To illustrate this, we examine the one-dimensional Poisson's equation and evaluate the gradient for residual and boundary loss terms. We show that the characteristics of the gradient of the loss function are such that for residual loss, shallow architectures converge faster. Empirically, we show the implications of our theory through various experiments.

Keywords: PINNs · Partial Differential Equations · Deep Learning.

1 Introduction

Neural networks have gained popularity in various fields such as computer vision, machine translation, and weather prediction [18,25,8,12]. In recent years, a type of deep neural model called Physics-Informed Neural Networks (PINNs) has emerged as a method for solving partial differential equations (PDEs) in a semi-supervised manner [20]. PINNs are trained using a loss function that includes the L2 norm of the residual of the governing PDEs, as well as losses for initial and boundary data. What makes PINNs special is their use of automatic differentiation to evaluate the residual rather than relying on simulation data from traditional PDE solvers. This makes the approach semi-supervised, as only boundary condition data are provided for supervision, while other points utilize the PDE residual loss.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_39

Despite the observed empirical success, the training efficiency of residual-driven PINNs at different network depths (number of layers) has not been well documented. In AI tasks such as computer vision and natural language processing, deep architectures with a larger number of hidden layers are commonly used in neural models. For these tasks, having an insufficient number of layers is shown to require a significantly higher number of parameters compared to deeper networks [2]. However, most PINN models use a shallow network with a modest number of layers (around 5) compared to deep networks that are typically used in AI tasks. Additionally, there is limited research that provides theoretical explanations for hyperparameter tuning in PINNs [29,6].

Our objective is to answer the question “Why do PINN models work well with fewer layers?". To answer this question, we study the properties of the loss function through an analytical solution, a simple network with an analytically tractable number of parameters, and with numerical test cases to show how the gradient of PINN loss behaves for relatively shallower shallow and deep networks with the same parameter count. We find that shallower PINN models have steeper gradients that help them achieve faster convergence, compared to deeper networks with the same parameter count.

In what follows, we first provide a background of PINNs. Next, we analyze the gradient of the residual and boundary losses by considering an analytical solution of Poisson’s equation, for a one-layer and two-layer neural network. Finally, we show the implications of our theory through simulations of the Kovasznay flow, Lid-driven cavity flow, and atmospheric boundary layer flow experiments.

2 Background of PINNs

Consider a general nonlinear PDE,

$$\mathcal{N}(u, x) = f, x \in \Omega\,, \tag{1}$$

$$\mathcal{B}(u(x)) = b, x \in \partial\Omega\,, \tag{2}$$

where $\mathcal{N}$ is a generic nonlinear function, $\mathcal{B}$ are the initial and boundary condition operator, $x \in \Omega$ is the coordinates in the domain Ω, $\partial\Omega$ is the boundary of the domain, b is the boundary condition value and $u \in \mathcal{U}$ is the solution field in a solution space $\mathcal{U}$ that satisfies the above PDE. As a specific example, for 2D fluid flow problems, the vector u consists of the flow velocities (horizontal, vertical) and the pressure. The domain coordinates are (x, y, t) for unsteady flow and (x, y) for steady flows.

Traditionally, finite discretization-based numerical schemes or spectral methods have been used to solve the above PDE. The PINNs approach uses and trains a parametrized neural model $\mathcal{H}$ that approximates the functional map between the input domain Ω and the solution domain $\mathcal{U}$, i.e., $\mathcal{H} : x \times \theta \mapsto u$, such that the differential operator with the initial and boundary conditions are satisfied. Here θ is the set of parameters of the neural network $\mathcal{H}$ that is learned by minimizing a loss function $\mathcal{L}_{PINN}$ that comprises of the PDE-residual term and boundary data loss term, i.e.,

$$\mathcal{L}_{PINN} = \mathcal{L}_{residual} + \mathcal{L}_{boundary} \tag{3}$$

$$\mathcal{L}_{residual} = \frac{1}{N_c}\left[\sum_{i=1}^{N_c}(\mathcal{N}(u_i, x_i) - f)^2\right] \tag{4}$$

$$\mathcal{L}_{boundary} = \frac{1}{N_{bp}}\left[\sum_{i=1}^{N_{bp}}(\mathcal{B}(u(x_i)) - b)^2\right] . \tag{5}$$

Here, N_c represents the total count of location points within the domain and N_{bp} denotes the total number of boundary points. The PDE loss term is computed using the automatic differentiation approach, while the boundary loss term is determined using the mean squared error with provided boundary data for the Dirichlet boundary condition. If the Neumann/Robin condition is specified, the boundary loss is computed with the help of automatic differentiation. The use of automatic differentiation allows to solve the PDE without relying on simulation data from numerical solvers as is done in neural operator methods such as DeepONets [17], and Fourier Neural Operators [15]. Practically, a PINN model is a dense neural network with input as the coordinates and the output is the solution variable of the PDE. For neural models, the words *shallow* and *deep* are relative and context-dependent. Our usage is as follows. For the same number of trainable parameters, a network with fewer layers is called a shallow network, and a network with more layers is called a deep network. Most PINN models in the literature use a shallow dense neural network with about 5 layers [21,18,16,8,19,30,4] compared to typical deep networks used in AI tasks of computer vision and natural language processing.

3 Analysis of the gradient of the residual loss

The key distinguishing factor between PINNs and neural models used for AI tasks is the objective function and training strategy. In PINNs, the goal is to minimize a combined loss function with PDE residual and boundary condition data. Additionally, the residual loss of the PDE is calculated by automatic differentiation. In contrast, traditional machine learning tasks typically involve working with existing datasets, and the objective is to generalize from that data. In PINNs minimizing the objective function is of paramount importance, whereas in usual machine learning tasks typically involve working with existing datasets, and the objective is to generalize from that data.

Gradient descent based algorithms are typically used to train neural networks. The nature of the gradient determines the convergence of the training. As PINNs have two components (residual and boundary loss), the gradients of each term of the loss affect the convergence. We examine the magnitude of the gradients of these two terms with respect to the parameters of the neural network to determine how the training will progress.

To conduct this study, we take a progressive approach. First, we use the analytical solution of the 1D Poisson's equation to gain insights into the difference between the gradients of these two terms with respect to parameters of the PINN model.

The 1D Poisson's equation is

$$\begin{aligned} \frac{\partial^2 u}{\partial x^2} &= -\pi^2 \sin(\pi x)\,, x \in [-0.5, 0.5] \\ u(-0.5) &= -1\,, \\ u(0.5) &= 1\,, \end{aligned}$$

with an analytical solution $u(x) = \sin(\pi x)$.

Consider a trained neural network $f_\theta(x)$ that accurately approximates the solution $u(x)$. Without loss of generality, we may express the approximate solution as $f_\theta(x) = u(x)\epsilon_\theta(x)$, where $\epsilon_\theta(x)$ is defined for $x \in [-0.5, 0.5]$ and $\frac{\partial \epsilon_\theta(x)}{\partial x} < \epsilon$, for some $\epsilon > 0$. The gradient of residual loss and boundary loss of f_θ is

$$\begin{aligned} \frac{\partial \mathcal{L}_{residual}}{\partial \theta} &\approx \pi^4(\epsilon_\theta(x) - 1)\frac{\partial \epsilon_\theta(x)}{\partial \theta}\,, \\ \frac{\partial \mathcal{L}_{boundary}}{\partial \theta} &\approx 4(\epsilon_\theta(x) - 1)\frac{\partial \epsilon_\theta(x)}{\partial \theta}\,. \end{aligned}$$

The full derivation of the above is in Appendix 6.1, attached in supplementary material. We see that the gradient of the residual term with respect to the parameters of the neural network is approximately $\mathcal{O}(10) - -\mathcal{O}(100)$ times the gradient of the boundary loss term with respect to the parameters [28]. This observation holds for a variety of equations as reported in other PINN studies [24,28].

Next, we consider two networks with two neurons and only two trainable parameters w_1, w_2. In the first network, we use only one layer (Fig. 1a); in the second, we use two layers (Fig. 1b). The weights on the remaining edges are assumed to be one for analysis. All hidden neurons use the swish activation function as commonly used in PINNs [22]. An input (x) gives the output (U) for the shallow (one-layer) and deep (two-layer) networks as follows.

$$U_{Shallow} = \frac{w_1 x}{1 + e^{(-w_1 x)}} + \frac{w_2 x}{1 + e^{(-w_2 x)}}\,, \tag{6}$$

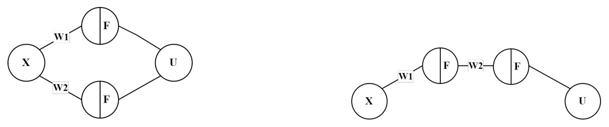
$$U_{Deep} = \frac{\dfrac{w_2 w_1 x}{1 + e^{(-w_1 x)}}}{1 + e^{-\left(\dfrac{w_2 w_1 x}{1 + e^{(-w_1 x)}}\right)}}\,. \tag{7}$$

The efficiency of learning for a neural network through backpropagation depends on the characteristics of the loss function. From our analysis above, we

Table 1: Hyperparameters used in all experiments to study the effect of network depth.

Hyperparameters									
Flows	Kovaszny flow			Lid-driven cavity			ABL		
Hidden layers	5	10	15	5	10	15	5+1[a]	10+1[a]	15+1[a]
Neurons per layer	10	6(5), 7(10)	5(10), 6(5)	300	200	161(5), 160(10)	64(5), 8(1)	36(10), 36(1)	32(15), 24(1)
Learning rate	0.001								
Epochs	2,000			30,000			25,000		
Optimizer	Adam								
Trainable-parameters	500	494	495	363,000	363,000	363,051	19,472	18,936	18,960
Grid points	64*64			50*50			50*500		
Activation function	Swish						Swish + ReLU		
Initializer	Glorot Uniform			Glorot Normal			Glorot Uniform		

[a] One layer of neurons for each output separately. More details in Appendix 6.2.



(a) 1 hidden layer neural network

(b) 2 hidden layer neural network

Fig. 1: The architectures used to analyse the residual loss for the Poisson's equation. Here, the 1 layer network is the "shallow" network and the 2 layer network is the "deep" network.

know that the gradient of the residual loss term is greater. Therefore, we plot the residual loss in terms of w_1 and w_2 to analyze the difference in the loss landscape for a 1-layer (called shallow) and 2-layer (called deep) PINN. Approximately the residual term of the loss function for shallow and deep network is

$$\mathcal{L}_{Shallow} \approx \frac{w_1^2 e^{-w_1 x}}{(1+e^{-w_1 x})^2} + \frac{w_2^2 e^{-w_2 x}}{(1+e^{-w_2 x})^2} ,$$
$$\mathcal{L}_{Deep} \approx \frac{w_1 w_2}{(1+e^{-w_1 x})^2 (1+e^{\frac{-w_1 w_2 x}{1+e^{-w_1 x}}})^2} .$$

This loss is evaluated for a randomly chosen x in the input range and with a grid of weights and visualized in Fig. 2. We see that the loss landscape of the shallow network has a prominent valley that can easily be reached by the gradient descent algorithm, whereas the deep network has a shallow valley that is harder to optimize. Hence, we expect the shallow network to converge faster than the deep network during optimization [14,7].

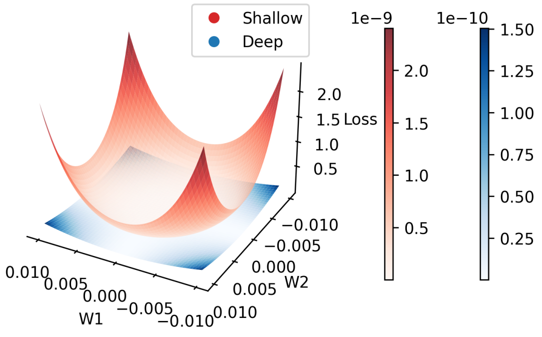


Fig. 2: The residual loss plotted against the parameters of the neural network for shallow (1 layer) and deep (2 layers) networks.

4 Experimental results

To empirically confirm that shallow networks are indeed more efficient to train than deep networks for PINN tasks, we conduct experiments with these three test cases, viz., (i) the Kovasznay flow, (ii) the lid-driven cavity flow, and (iii) the Atmospheric Boundary Layer (ABL) flow. For each case, we train PINN models with 5, 10 and 15 layers and compare the solutions of each to a ground truth obtained either analytically or numerically.

4.1 Kovasznay flow

The Kovasznay flow [10,27] is governed by the steady state Navier-Stokes equations

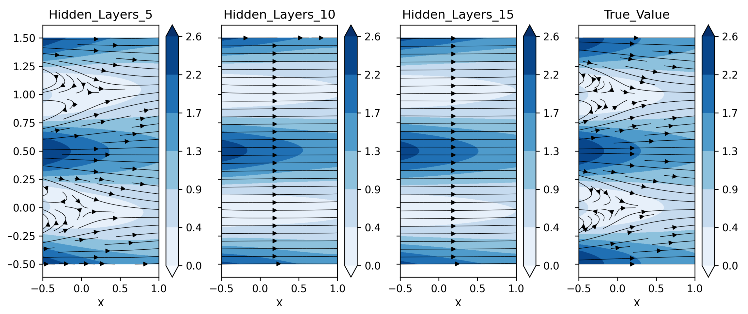


Fig. 3: The solution of PINNs trained with 5, 10 and 15 hidden layers is shown next to the true solution of Kovasznay flow with Re=40. Flow streamlines are overlaid on a background of velocity magnitude.

$$
\begin{aligned}
(\boldsymbol{V}.\nabla)\boldsymbol{V} &= -\nabla p + \frac{1}{Re}(\nabla^2 \boldsymbol{V}) \\
\nabla.\boldsymbol{V} &= 0\,, \\
\boldsymbol{V}|_{bdry} &= g(x_b, y_b)\,,
\end{aligned} \tag{8}
$$

where $\boldsymbol{V}$ is the velocity vector, p is the pressure, Re is the Reynold's number, ∇ is the gradient operator, ∇^2 is the Laplace operator and g is the boundary data function. The Kovasznay flow has an analytical solution given by

$$
\begin{aligned}
\zeta &= \frac{0.5}{\mu} - \sqrt{\frac{1}{4\mu^2} + 4\pi^2}\,, \\
u &= 1 - \exp(\zeta x)\cos(2\pi y)\,, \\
v &= \frac{\zeta}{2\pi}\exp(\zeta x)\sin(2\pi y)\,, \\
p &= 0.5(1 - \exp(2\zeta x))\,,
\end{aligned}
$$

where μ is the viscosity, x and y are the cartesian coordinate system, u and v are horizontal and vertical components of the velocity, and p is pressure. The boundary condition of the Navier-Stokes equation is given by the analytical solution evaluated at the boundary points.

We solve the Kovasznay flow using the PINN approach in a square grid with $x \in [-0.5, 1.0]$, $y \in [-0.5, 1.5]$ and $\mu = 0.025$. The loss function consists of the residual of the PDE in eq. (8) and the boundary loss.

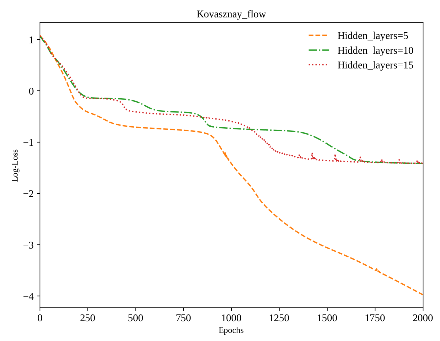


Fig. 4: Log-loss plot during training of the PINN with 5, 10, and 15 hidden layers for the Kovasznay flow test case.

We conducted experiments using PINN models with 5, 10, and 15 hidden layers. To ensure a fair comparison, we use approximately the same number of trainable parameters for these three models and keep all other training algorithm hyperparameters fixed. Consequently, shallower networks have more neurons in each layer compared to deeper networks, as the total number of trainable parameters remains approximately the same. The main hyperparameters used are presented in Table 1. The training process consists of 2000 epochs with a learning rate of 0.001. The regular training of the PINNs is completed and the trained model is then evaluated on a 64×64 regular grid for comparison with the ground truth obtained from the analytical solution. Figure 3 illustrates the solution for the PINN models with 5, 10, and 15 hidden layers, together with the reference ground truth solution. It is observed that only the shallow model with 5 hidden layers successfully trained, while the deeper models failed to accurately capture the physics of the flow. Figure 4 shows the log-loss plot for all three models. The reduction in loss is significant for the model with 5 hidden layers, whereas the loss plateaus for the other two models. This observation directly confirms our hypothesis that shallower networks have a favorable loss landscape for the optimization method to find a minimum for PINNs training dominated by the residual loss. The residual loss gradient is higher compared to the boundary loss gradient with respect to the network parameters for most of the epochs in this case as well, which is consistent with our analysis (Sec. 3) (See also Appendix 6.3)

4.2 Lid-driven cavity flow

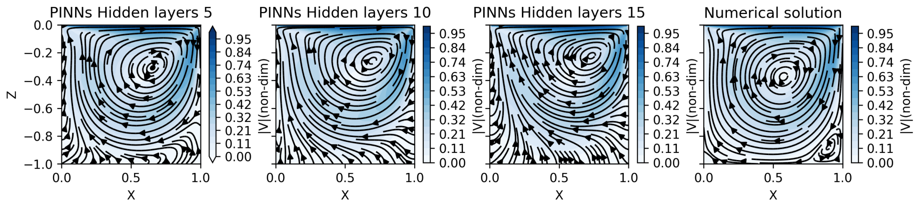
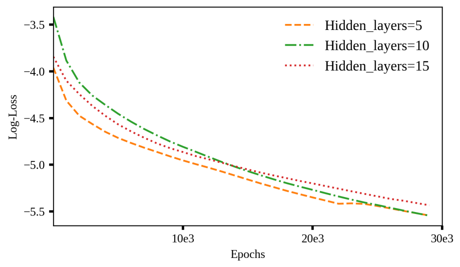


Fig. 5: The solution of PINNs trained with 5, 10 and 15 hidden layers is shown next to the true solution of the lid-driven cavity flow for Re=300. Flow streamlines are overlaid on a background of velocity magnitude.

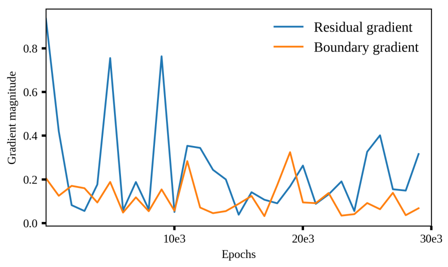
The second test case chosen is the steady-state lid-driven cavity flow problem, which is an idealization of the wind-driven response of the ocean and is of practical relevance. The rightmost panel of Fig. 5 shows the flow setup. A square fluid-filled cavity in the $x - z$ plane is subject to a horizontal motion on the top surface, corresponding to the lid moving to the right at a specified velocity of 1 non-dimensional unit. The PINNs simulation employs the Navier-Stokes equation (Eq. 8) along with the following boundary conditions: $u, v = 0$ for left, bottom, and right, and $u = 1, v = 0$ on top surface. Simulations are compared with the solution obtained from a numerical CFD solver [26].

We train PINNs with 5, 10, and 15 hidden layers, all with approximately the same number of trainable parameters, and other hyperparameters are fixed, similar to the previous test case. The main hyperparameters are shown in Table 1.

Figure 5 shows the flow streamlines and magnitude of the lid-driven cavity problem for Re = 300 for different numbers of hidden layers. All models were trained for 30,000 epochs. The flow simulation using the network with five hidden layers agrees better with the flow simulation using the CFD tool, while those with more layers do not. Figure 6a shows the learning curve during training, i.e., the plot of log-loss versus the number of epochs. The network with five hidden layers has the lowest loss compared to the other networks. Table 2 documents the mean relative error between the PINN and CFD simulations for the different output variables. The relative error and the confidence interval are the lowest for the network with five hidden layers. A noticeable difference in the number of epochs to reach a certain threshold log-loss can also be observed between the models. For example, to reach a log-loss value less than -5.1, models with 5, 10, and 15 hidden layers took nearly 14000, 16000, and 17000 epochs, respectively. Figure 6b shows that the residual loss gradient is higher compared to the boundary loss gradient with respect to network parameters, which aligns with our analysis in Sec. 3.



(a)



(b)

Fig. 6: (a) Log-loss plot for 5, 10 and 15 hidden layers PINN models to simulate the lid-driven cavity flow. (b) L2 norm of the gradient vector of residual loss and boundary loss with respect to neural network parameters for the lid-driven cavity flow problem for 5 hidden layers.

Table 2: Relative error in u, v, and p with their confidence interval for lid-driven cavity flow test case.

Relative error			
Layers	U Error	V Error	P Error
5	0.03±0.001	0.04±0.001	0.015±0.0001
10	0.05±0.002	0.06±0.002	0.02±0.0002
15	0.06±0.003	0.07±0.003	0.04±0.0006

4.3 Atmospheric boundary layer (ABL)

Our third test case is the turbulent atmospheric boundary layer flow simulation, a much more complex test case than the previous two. A horizontally homogeneous atmospheric boundary layer flow, as shown in Figure 7, is simulated in a uniformly rough, flat terrain under neutral stratification. The steady Reynolds-averaged Navier-Stokes (RANS) framework with two-equation turbulence models has been shown to be effective for the simulation of the ABL flow [1,11]. In the present work, we use the standard $k - \epsilon$ model [9] for turbulence closure along with the standard wall functions [13] modified to be consistent with the inlet profiles [23].

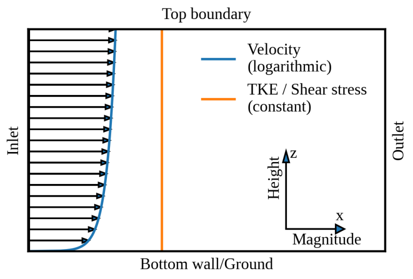


Fig. 7: Illustration of the vertical distribution of horizontal velocity and TKE / shear stress within the Prandtl layer of the atmospheric boundary layer.

RANS governing equations The incompressible steady RANS equations, along with the turbulent kinetic energy (TKE) (k) and the dissipation rate (ϵ) transport equations for the closure in the Cartesian coordinates, are as follows.

$$\text{Continuity equation: } \frac{\partial u_i}{\partial x_i} = 0\,, \tag{9}$$

$$\text{Momentum equation: } \frac{\partial u_i u_j}{\partial x_j} = \frac{\partial}{\partial x_j}\left[(\nu + \nu_t)\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\right] - \frac{1}{\rho}\frac{\partial P}{\partial x_i}\,, \tag{10}$$

$$k \text{ transport equation: } \frac{\partial k u_j}{\partial x_j} = \frac{\partial}{\partial x_j}\left[\left(\nu + \frac{\nu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right] + P_k - \epsilon\,, \tag{11}$$

$$\epsilon \text{ transport equation: } \frac{\partial \epsilon u_j}{\partial x_j} = \frac{\partial}{\partial x_j}\left[\left(\nu + \frac{\nu_t}{\sigma_\epsilon}\right)\frac{\partial \epsilon}{\partial x_j}\right] + C_{1\epsilon}\frac{\epsilon}{k}P_k - C_{2\epsilon}\frac{\epsilon^2}{k}\,, \tag{12}$$

where x_i is the spatial coordinate, u_i is the time-averaged velocity vector (u, w for 2D (x, z) and u, v, w for 3D (x, y, z)), P is the mean pressure, ν is the kinematic viscosity, ν_t is the turbulent viscosity, P_k is the production of turbulent kinetic energy. P_k and ν_t are given by the equations,

$$P_k = \nu_t \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \frac{\partial u_i}{\partial x_j}\,, \qquad\qquad \nu_t = C_\mu \frac{k^2}{\epsilon}\,.$$

Here, the values of the five constants $C_\mu, \sigma_k, \sigma_\epsilon, C_{1\epsilon}$ and $C_{2\epsilon}$ used for simulation of neutral atmospheric flow are 0.033, 1.0, 1.3, 1.176, and 1.92, respectively [5].

Boundary conditions For homogeneous ABL flow simulations in the stream-wise direction, we use a fully developed inlet velocity profile given by [23],

$$u = \frac{u_\tau}{\kappa} \ln \left(\frac{z + z_0}{z_0} \right), \tag{13}$$

where $u_\tau = \sqrt{\frac{\tau_w}{\rho}}$ is the frictional velocity (with τ_w is the wall shear stress, and ρ is the fluid density), $\kappa \approx 0.418$ is the von Kármán constant, z is the height co-ordinate and z_0 is the aerodynamic roughness height. Wall functions are used to achieve sufficiently precise solutions in the region close to the wall/ground, which helps reduce the computational cost and allows the inclusion of empirical information in special cases, such as rough wall conditions, to be used in ABL [13]. Turbulent kinetic energy and dissipation are specified on the bottom wall/ground using the equations,

$$k = \frac{u_\tau^2}{\sqrt{C_\mu}}, \qquad\qquad \epsilon = \frac{u_\tau^3}{\kappa(z + z_0)}\,,$$

with $z = z_p$, where z_p is the height from the wall such that $90 < \frac{z_p u_\tau}{\nu} < 500$ [3]. The velocity is also specified at the wall using eq. 13. At the top boundary, the symmetry boundary condition is used for velocity, pressure, and other turbulent quantities. A pressure outlet boundary condition is specified for the outlet with vanishing stream-wise gradients of other quantities.

ABL simulation result Figure 8 shows plots of the stream-wise velocity, turbulent kinetic energy, and dissipation rate profiles against height obtained from the PINN models with 5, 10 and 15 hidden layers. The inlet profile is also plotted for reference. The hyperparameters used for the PINN models are shown in Table 1. Here, we use a reference velocity (u_{ref}) of $8m/s$ at a reference height (z_{ref}) of $70m$ and at a turbulence intensity of 7%. The domain is $100m$ high and $200m$ long in the stream-wise direction. The frictional velocity, calculated using Eq. 13 with 0.001 as the aerodynamic roughness height, is used to obtain values of other turbulent quantities. A successful simulation will have obtained horizontally homogeneous ABL flow under neutral stratification, that is, the PINN simulations and reference inlet profiles would be identical. The flow is also characterized by constant shear stress over the height, providing a constant profile for the turbulent kinetic energy as seen in Figure 8 panel for TKE.

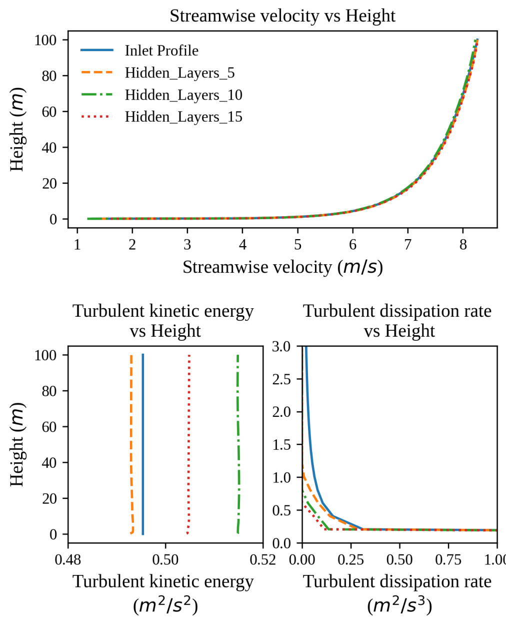


Fig. 8: Stream-wise velocity, turbulent kinetic energy and turbulent dissipation rate profiles of the ABL flow simulated from PINN models with 5, 10 and 15 hidden layers are compared to the reference (inlet profile). For the turbulent dissipation rate, the height is shown only until 3 m for highlighting the differences.

From Figure 8, it can be seen that the velocity profiles simulated by all networks are in good agreement with the inlet profile. However, the deviations in the TKE and dissipation rate profiles from the reference are more evident in models with greater number of hidden layers. The greater difference for 10 and 15 layer PINN models can be attributed to the fact that for TKE and dissipation rate, the Dirichlet condition was provided only at the bottom wall, and PDE losses were used to solve within the domain. This approach made the loss function residual dominated and as per our analysis (Sec 3), we expect shallower networks to perform better here as evidenced by our experimental results. The absolute error between the inlet and outlet profiles simulated using the different PINN models is listed in Table 3. The TKE error in the 5 hidden layer model is almost one order less compared to other models. The velocity values were provided at the inlet and the bottom wall. Thus, all models were able to simulate this profile well, which shows that all models perform well in supervised learning tasks. From the log-loss plot (Figure 9), we see that for the same number of epochs, the model with 5 hidden layers ended the training with the least loss. In our experiments, we have fixed the number of parameters and epochs to be equal for models with different hidden layers. A noticeable difference in the number of epochs to reach a certain threshold log-loss can also be observed between the models. For example, to reach a log-loss value less than 4, models with 5, 10, and 15 hidden layers took nearly 9000, 12000, and 16000 epochs, respectively. Also, it should be noted that the computational time for the model with 5 hidden layers is $\frac{4}{3}$ and $\frac{5}{3}$ times lesser than the models with 10 and 15 hidden layers. The residual loss gradient is higher compared to the boundary loss gradient with respect to the network parameters in this case as well, which is consistent with our analysis (Sec. 3) (See also Appendix 6.4)

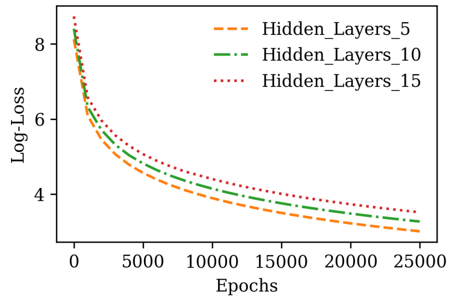


Fig. 9: Log-loss plot for 5, 10, and 15 hidden layers for the ABL flow.

Table 3: Mean absolute error in u, k, and ϵ with their confidence interval for ABL flow.

Mean absolute error			
Hidden Layers	u error	k error	ϵ error
5	$0.016 \pm 5 \times 10^{-5}$	$0.002 \pm 2 \times 10^{-5}$	$0.003 \pm 4 \times 10^{-5}$
10	$0.019 \pm 5 \times 10^{-5}$	$0.020 \pm 8 \times 10^{-8}$	$0.005 \pm 1 \times 10^{-3}$
15	$0.024 \pm 2 \times 10^{-4}$	$0.010 \pm 1 \times 10^{-7}$	$0.004 \pm 2 \times 10^{-4}$

5 Conclusion

The results of our study suggest that shallow neural networks are more beneficial than deeper networks in the training of PINNs. We also provide an explanation for why shallow networks tend to converge better than deep networks, based on an analysis of the gradient of the loss function. To illustrate our findings, we present various test cases. Initially, these results may seem counter-intuitive, particularly when compared to the prevailing knowledge and empirical evidence in the fields of computer vision and natural language processing in AI. However, our analysis of the loss function for PDE residual-driven training clearly demonstrates why relatively shallow networks perform well in PINN model training.In all instances, it was observed that the gradient of the residual loss is higher when compared to the gradient of the boundary loss. While this holds true in the majority of cases, in the future we explore scenarios where this observation may not be accurate.

Supplementary material : Appendix and Codes

Acknowledgment. This research is partially supported by research grants from the Ministry of Earth Sciences (MoES/36/OOIS/Extra/84/2022) and Ministry of Education (MHRD STARS 1/388 SPADE). We are grateful for the PMRF fellowship support to Azhar Gafoor.

References

1. Apsley, D.D., Castro, I.P.: A limited-length-scale k-ε model for the neutral and stably-stratified atmospheric boundary layer. Boundary-layer meteorology **83**, 75–98 (1997)
2. Bengio, Y., et al.: Learning deep architectures for ai. Foundations and trends® in Machine Learning **2**(1), 1–127 (2009)
3. Blocken, B., Stathopoulos, T., Carmeliet, J.: Cfd simulation of the atmospheric boundary layer: wall function problems. Atmospheric environment **41**(2), 238–252 (2007)
4. Cai, S., Wang, Z., Wang, S., Perdikaris, P., Karniadakis, G.E.: Physics-informed neural networks for heat transfer problems. Journal of Heat Transfer **143**(6), 060801 (2021)
5. El Kasmi, A., Masson, C.: An extended k–ϵ model for turbulent flow through horizontal-axis wind turbines. Journal of Wind Engineering and Industrial Aerodynamics **96**(1), 103–122 (Jan 2008). https://doi.org/10.1016/j.jweia.2007.03.007
6. Elsken, T., Metzen, J.H., Hutter, F.: Neural architecture search: A survey. The Journal of Machine Learning Research **20**(1), 1997–2017 (2019)
7. Glorot, X., Bengio, Y.: 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)
8. Jin, X., Cai, S., Li, H., Karniadakis, G.E.: Nsfnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations. Journal of Computational Physics **426**, 109951 (2021)
9. Jones, W.P., Launder, B.E.: The prediction of laminarization with a two-equation model of turbulence. International Journal of Heat and Mass Transfer **15**(2), 301–314 (Feb 1972). https://doi.org/10.1016/0017-9310(72)90076-2
10. Kovasznay, L.I.G.: Laminar flow behind a two-dimensional grid. Mathematical Proceedings of the Cambridge Philosophical Society **44**(1), 58–62 (1948). https://doi.org/10.1017/S0305004100023999
11. van der Laan, M.P., Kelly, M., Floors, R., Peña, A.: Rossby number similarity of an atmospheric rans model using limited-length-scale turbulence closures extended to unstable stratification. Wind Energy Science **5**(1), 355–374 (2020)
12. Lam, R., Sanchez-Gonzalez, A., Willson, M., Wirnsberger, P., Fortunato, M., Alet, F., Ravuri, S., Ewalds, T., Eaton-Rosen, Z., Hu, W., et al.: Learning skillful medium-range global weather forecasting. Science p. eadi2336 (2023)
13. Launder, B.E., Spalding, D.B.: The numerical computation of turbulent flows. Computer Methods in Applied Mechanics and Engineering **3**(2), 269–289 (Mar 1974). https://doi.org/10.1016/0045-7825(74)90029-2
14. Levin, E., Fleisher, M.: Accelerated learning in layered neural networks. Complex systems **2**(625-640), 3 (1988)

15. Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., Anandkumar, A.: Fourier neural operator for parametric partial differential equations. arXiv preprint arXiv:2010.08895 (2020)
16. Lim, K.L., Dutta, R., Rotaru, M.: Physics informed neural network using finite difference method. In: 2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC). pp. 1828–1833. IEEE (2022)
17. Lu, L., Jin, P., Pang, G., Zhang, Z., Karniadakis, G.E.: Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. Nature machine intelligence **3**(3), 218–229 (2021)
18. Mao, Z., Jagtap, A.D., Karniadakis, G.E.: Physics-informed neural networks for high-speed flows. Computer Methods in Applied Mechanics and Engineering **360**, 112789 (2020)
19. Pang, G., Lu, L., Karniadakis, G.E.: fpinns: Fractional physics-informed neural networks. SIAM Journal on Scientific Computing **41**(4), A2603–A2626 (2019)
20. Raissi, M., Perdikaris, P., Karniadakis, G.: 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). https://doi.org/https://doi.org/10.1016/j.jcp.2018.10.045, https://www.sciencedirect.com/science/article/pii/S0021999118307125
21. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational physics **378**, 686–707 (2019)
22. Ramachandran, P., Zoph, B., Le, Q.V.: Searching for activation functions. arXiv preprint arXiv:1710.05941 (2017)
23. Richards, P.J., Hoxey, R.P.: Appropriate boundary conditions for computational wind engineering models using the k-ϵ turbulence model. Journal of Wind Engineering and Industrial Aerodynamics **46–47**, 145–153 (Aug 1993). https://doi.org/10.1016/0167-6105(93)90124-7
24. Sankaran, S., Wang, H., Guilhoto, L.F., Perdikaris, P.: On the impact of larger batch size in the training of physics informed neural networks. In: The Symbiosis of Deep Learning and Differential Equations II (2022)
25. Sun, L., Gao, H., Pan, S., Wang, J.X.: Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. Computer Methods in Applied Mechanics and Engineering **361**, 112732 (2020)
26. Ueckermann, M.P., Lermusiaux, P.F.: 2.29 finite volume matlab framework documentation. MSEAS report **14** (2012)
27. Wang, C.: Exact solutions of the steady-state navier-stokes equations. Annual Review of Fluid Mechanics **23**(1), 159–177 (1991)
28. Wang, S., Teng, Y., Perdikaris, P.: Understanding and mitigating gradient flow pathologies in physics-informed neural networks. SIAM Journal on Scientific Computing **43**(5), A3055–A3081 (2021)
29. Wang, Y., Han, X., Chang, C.Y., Zha, D., Braga-Neto, U., Hu, X.: Auto-pinn: understanding and optimizing physics-informed neural architecture. arXiv preprint arXiv:2205.13748 (2022)
30. Yang, L., Meng, X., Karniadakis, G.E.: B-pinns: Bayesian physics-informed neural networks for forward and inverse pde problems with noisy data. Journal of Computational Physics **425**, 109913 (2021)