AIMS Mathematics, 11(10): 34780–34804.
DOI: 10.3934/math.20261355
Received: 01 July 2026
Revised: 24 August 2026
Accepted: 20 September 2026
Published: 26 October 2026

https://www.aimspress.com/journal/Math

*Research article*

# Improving physics-informed neural networks performance with tree-structured Parzen estimator (TPE)-based Bayesian hyperparameters tuning: Application to nonlinear partial differential equations

**Safeer Hamza Tahir[1], Sadia Saeed[1,*], Reham Alahmadi[2], Saifudin Hafiz Yahaya[3] and Umair Ali[1,*]**

[1] Department of Applied Mathematics and Statistics, Institute of Space Technology, Islamabad, Pakistan
[2] Department of Basic Sciences, College of Sciences and Theoretical Studies, Prince Musaid bin Abdulrahman University, Riyadh, Saudi Arabia
[3] Fakulti Teknologi dan Kejuruteraan Industri dan Pembuatan, Universiti Teknikal Malaysia Melaka, 76100 Durian Tunggal, Melaka

* **Correspondence:** Email: sadia.saeed@ist.edu.pk, umairkhanmath@gmail.com.

**Abstract:** Physics-informed neural networks (PINNs) have become a powerful meshless technique for solving partial differential equations (PDEs). However, the network architecture and the selection of hyperparameters significantly influence their performance. The majority of existing research works are based on heuristic methods for hyperparameter tuning. This often results in lower accuracy and unstable convergence, especially for nonlinear stiff PDEs. In this study, we address this limitation by introducing a comprehensive and systematic tune-first PINN framework. We employ the tree-structured Parzen estimator (TPE) within a Bayesian optimization scheme to fine-tune key hyperparameters, including network depth, width, collocation points, weights associated with PDE and initial conditions, Fourier features, and learning rates. The optimized configuration is subsequently used for high-fidelity PINN training. We validate the proposed methodology using challenging one- and two-dimensional nonlinear, stiff PDEs. Numerical results indicate that systematic tuning of PINN architecture improves performance by up to 85% compared to a standard PINN with manually selected hyperparameters. These findings underscore that hyperparameter optimization provides a reliable and reproducible approach for improving the accuracy and stability of PINNs, and it is adaptable to higher-dimensional problems and other stiff nonlinear PDEs.

**Keywords:** physics-informed neural networks; Bayesian optimization; hyperparameter tuning; tree-structured Parzen estimators; partial differential equations
**Mathematics Subject Classification:** 82B24, 35G20, 35J60,62C10

## 1. Introduction

Numerous physical systems in science and engineering are represented by partial differential equations (PDEs) [1,2]. These equations are often complex, stiff, and nonlinear. This complexity makes finding exact solutions challenging. As a result, researchers frequently employ numerical techniques, such as finite difference methods [3,4], finite volume methods [5], and finite element methods [6], to solve these equations. Although these approaches are effective for approximating solutions to complex PDEs, they also have limitations. In particular, solving high-dimensional problems with these methods is often time-consuming and computationally expensive [7,8].

Recently, deep learning frameworks have emerged as promising alternatives to traditional numerical methods [9–11]. This progress is driven by advances in computational power, particularly GPU acceleration, and the demonstrated capacity of deep learning to approximate complex, nonlinear, high-dimensional mappings [11–13]. As a result, deep neural networks (DNNs) have been successfully applied across various scientific computing domains, including materials science [9], fluid dynamics [11,12], and uncertainty quantification [13]. Deep learning techniques can be classified as supervised and unsupervised learning based on labeled and unlabeled data. In supervised learning convolution neural networks are commonly used for image detection, and graph-based deep learning techniques are used to connect the link between the components [14]. Another category is Physics-Informed Neural Networks (PINNs) [15–17], which incorporate physical constraints into neural network loss functions to address high-dimensional PDEs. Owing to their simplicity and strong experimental performance, PINNs have become a prominent approach in scientific machine learning. Machine-vision-driven PINNs have been utilized for pantograph catenary state detection, whereas graph-MDETR has been designed for unmanned aerial vehicle (UAV) based catenary support component detection [14]. Several extensions to the PINNs framework demonstrate significant potential for addressing challenges such as solving high-dimensional PDEs [18–21], uncertainty quantification [13], autonomous driving[10], and inverse problems [15]. PINNs still exhibit notable weaknesses that limit their practical applicability [22–24]. Training is often complicated by factors such as a rigid optimization landscape, gradient imbalance, and sensitivity to hyperparameters. We may refer to hyperparameters as model settings that are not determined by the training data. Instead, they govern the learning processes and define the model's structure. Typically, PINNs inherently possess several hyperparameters, including the learning rate, the architecture's width and depth, the choice of activation function, and the weights assigned to the loss function. It is important to recognize that additional hyperparameters, including the optimization algorithm, the learning rate (e.g., using a learning rate scheduler), and the distribution of collocation points, can be incorporated. This elevated dimensionality of the hyperparameter (HP) search space poses challenges for identifying suitable configurations—specifically, those that yield satisfactory generalization. The issue of slow training has consistently been a significant topic of discussion within the PINNs community [25–27]. Hyperparameter optimization (HP-tuning) [25–29], is a valuable approach to address these challenges. Various methodologies, including grid search [29] and random search optimization [26], are commonly

employed for hyperparameter selection. However, grid search is limited by the curse of dimensionality, and random search can be inefficient. Each hyperparameter evaluation requires training a deep neural network, underscoring the need to limit the total number of configurations. To overcome these limitations, this study introduces a systematic tuning procedure with three primary objectives: (i) Demonstrating that properly tuned PINNs exhibit superior accuracy and stability compared to untuned/standard models; (ii) Applying proposed methodology to several 0ne- and two-dimensional PDEs to evaluate its effectiveness using key performance metric; (iii) Providing insights into parameter sensitivity to inform the design of more efficient network architectures for nonlinearm PDEs.

This work applies tree-structured Parzen estimator (TPE)-based Bayesian optimization for hyperparameter tuning to address challenges arising from inadequate PINN training. An exhaustive overview is provided for tuning the number of hidden layers, the number of neurons per layer, the number of collocation points, and the weight associated with PDE and initial conditions, Fourier's features, and learning rates. The nonlinear, stiff PDEs are considered for implementation, and the optimized PINN is compared with the standard PINN to show the effectiveness of the proposed methodology.

The structure of this paper is as follows. Section 2 presents an overview of PINN methodology and the fundamental principles of Bayesian optimization, with emphasis on TPE-based Bayesian optimization for tuning PINN hyperparameters to improve accuracy in solving the nonlinear PDEs in one- and two-dimensions. Section 3 presents the results, and Section 4 concludes the paper and discusses directions for future work.

## 2. Methodology

In this section, we first outline the general methodology employed for hyperparameter tuning. After that, TPE-based Bayesian hyperparameter optimization is defined along with specific algorithms relevant to our research problems using the Optuna library in Python [30].

### *2.1. Problem statement*

Let us consider the general form of an nth-order PDE as follows:

$$u_t + \mathcal{N}[u(x,t)] = 0, \qquad x \in \Omega \subset \mathbb{R}^d, t \in (0,T\,]. \tag{2.1}$$

Here, $u(x,t)$ is the unknown solution that we need to determine. $\mathcal{N}(\cdot)$ is a differential operator having derivatives up to the nth order. The associated boundary and initial conditions are given by

$$u(x,t) = g(x,t), \quad (x,t) \in \partial\Omega \tag{2.2}$$

$$u(x,0) = h(x), \qquad x \in \Omega. \tag{2.3}$$

### *2.2. Physics-informed neural networks framework (PINNs)*

PINNs integrate physical laws, such as ordinary or partial differential equations, into the training

of deep neural networks. These equations are incorporated directly into the loss functions, enabling PINNs to adjust network weights, biases, and physics-related parameters during training. Let $u_\theta$ be the neural network approximation, and then the residual of Eq (2.1) can be represented by

$$r_\theta = \frac{\partial \hat{u}_\theta}{\partial \mathrm{t}} + \mathcal{N}[\hat{u}_\theta(x,t)].$$

The loss function typically consists of the following three components:

- MSE on the residual PDE

$$Loss_{pde} = \frac{1}{n_r}\sum_{k=1}^{n_r}\left|\frac{\partial \hat{u}_\theta}{\partial \mathrm{t}} - \mathcal{N}(\hat{u}_\theta(x_k^r, t_k^r)\right|^2; \tag{2.4}$$

- MSE on the initial condition

$$Loss_{ic} = \frac{1}{n_{ic}}\sum_{i=1}^{n_{ic}}\left|\hat{u}_\theta(\mathrm{x_i}^{ic}, 0) - \mathrm{h}(\mathrm{x_i}^{ic})\right|^2; \tag{2.5}$$

- MSE on the boundary conditions

$$Loss_{bc} = \frac{1}{n_{bc}}\sum_{j=1}^{n_{bc}}\left|\hat{u}_\theta(\mathrm{x_j}^{bc}, \mathrm{t_j^{bc}}) - \mathrm{g}(\mathrm{x_j}^{bc}, t_j^{bc})\right|^2. \tag{2.6}$$

Consequently, the overall loss function of the standard PINN is obtained by summing all the previously defined mean-squared error terms in Eqs (2.4)–(2.6):

$$Loss_{total}(\theta) = \lambda_{pde}Loss_{pde}(\theta) + \lambda_{ic}Loss_{ic}(\theta) + \lambda_{bc}Loss_{bc}(\theta). \tag{2.7}$$

Here, $\lambda_{pde}$, $\lambda_{ic}$, and $\lambda_{bc}$ denote the weighting coefficients which correspond to the PDE, initial condition, and boundary conditions loss terms, respectively. The objective of PINN training is to minimize $\theta^* = \arg\min_\theta Loss_{total}(\theta)$.

### *2.3. Tree-structured Parzen-based Bayesian hyperparameter optimization (TPE-based HPO)*

The performance of PINN depends on the selection of appropriate and suitable hyperparameters. A manually selected hyperparameter may perform well for one PDE but may perform poorly when applied to another PDE. Consequently, a systematic approach is applied for selecting appropriate hyperparameters using TPE-based Bayesian hyperparameter tuning. Let us define a vector of hyperparametrs $\Lambda = \{\mathrm{L}, \eta, \lambda_{\mathrm{pde}}, \lambda_{\mathrm{ic}}, \mathrm{n_{coll}}, \mathrm{w}, \ldots.\}$, where, L is numbers of layers, w is number of neurons, $\eta$ denotes learning rate, $\mathrm{n_{coll}}$ are collocation points, $\lambda_{\mathrm{pde}}$ and $\lambda_{\mathrm{ic}}$ are weights associated to pde and initial conditions loss respectively. The objective of Bayesian Optimization (BO) is to identify the minimum value of an objective function defined as

$$\Lambda^* = \arg\min_{\Lambda\in\vartheta} \mathcal{J}(\Lambda), \tag{2.8}$$

where $\vartheta$ denotes the search space of hyperparameters $\Lambda$ and $\mathcal{J}(\Lambda)$ measures the L2-relative error. BO is an efficient sequential approach for optimizing functions without requiring any explicit assumption about their functional form [25,27,30,31]. In BO, unlike grid search [29] or random search [26], previous

evaluation results are used to construct a probabilistic model that relates hyperparameters to the likelihood of achieving a given outcome in the objective function.

We implement BO using the TPE approach presented by [30]. TPE separates the model into a favorable set of hyperparameters for which $\mathcal{J} < \mathcal{J}^*$and unfavorable set of hyperparameters for which $\mathcal{J} \geq \mathcal{J}^*$with $\mathcal{J}^*$ as a threshold defined on the objective function. The corresponding probability densities are defined as $\ell(\Lambda) = p\ (\Lambda \mid \mathcal{J} < \mathcal{J}^*)$ and $m(\Lambda) = p\ (\Lambda \mid \mathcal{J} \geq \mathcal{J}^*$.TPE samples candidate configurations and favors those having a large $\ell(\Lambda)/\mathrm{m}(\Lambda)$ ratio.

Optuna is a Python library that implements a TPE-based Bayesian HPO tuning framework. Each Optuna trial represents a specific set of hyperparameters $\Lambda$ used to construct a PINN with the chosen configuration, which is then trained within computational resources. The resulting error from this process is returned as the objective function. The process of TPE-based BO is illustrated in Figure 1 below:

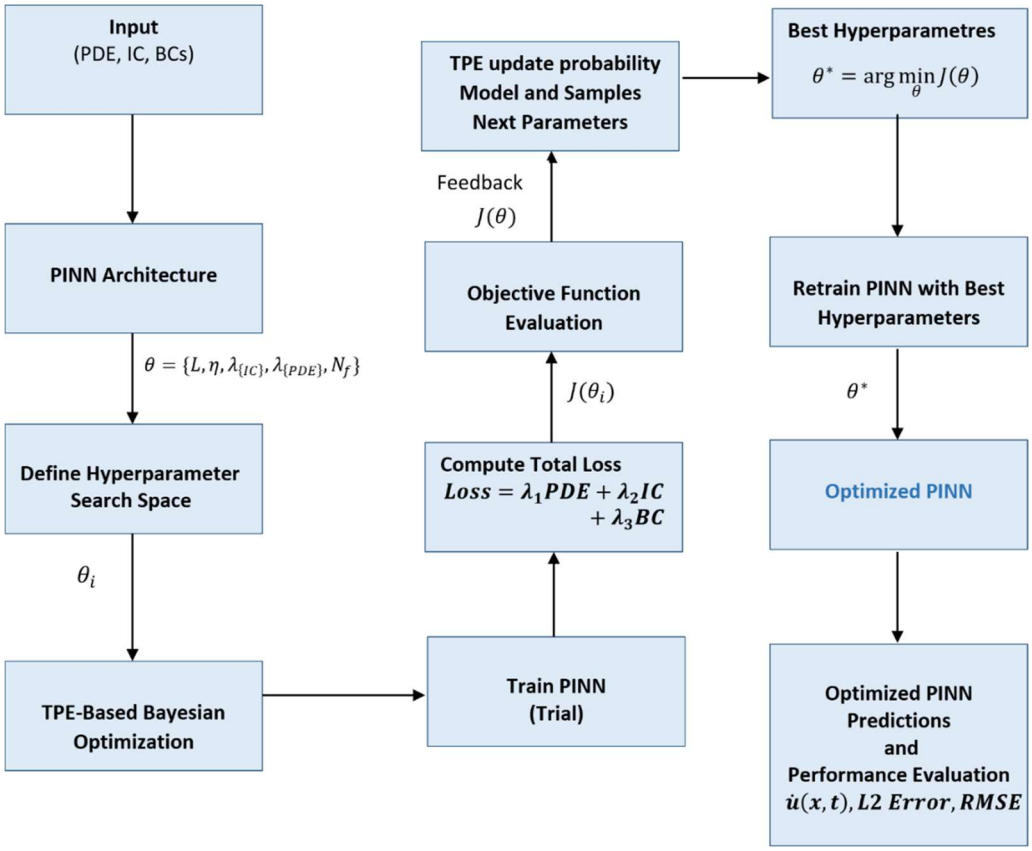



**Figure 1.** TPE-based Bayesian optimization architecture.

## 3. Results

The results of applying the proposed methodology to a series of test cases involving non-linear one-dimensional Burgers' and Allen-Cahn equations, as well as a nonlinear two-dimensional elliptic equation. To ensure a fair comparison, we construct the same neural network architecture for both

standard and optimized PINNs, with the only difference being the selection of hyperparameter methods. The standard PINN relies on manual selection of hyperparameters based on trial and error, whereas the optimized PINN utilizes hyperparameters determined through a tuned TPE-based BO process. Also, the hyperparameters search space for all the test cases was selected according to the ranges commonly used in PINN studies. These values were further constrained to ensure computational feasibility for the current study. All simulations were conducted on T4GPU setup, and code was implemented using pytorch library. For Optuna results, the TPE sampler was initialized with a fixed random seed and utilized multivariate sampling to capture interactions among hyperparameters. A median pruner was implemented to terminate underperforming trials early.

### *3.1 Test case 1: One-dimensional Burgers' equation*

Let us consider the following one-dimensional nonlinear viscous Burgers equation, subject to homogeneous Dirichlet boundary conditions and an initial condition defined as below in Eq (3.1)

$$\begin{aligned} u_t + \alpha u u_x - \nu u_{xx} = 0, \qquad & -1 < x < 1, 0 < t \leq 1 \\ u(-1,t) = u(1,t) = 0, \qquad & \forall\, t \in (0,1) \\ u(x,0) = -sin(\pi x). \qquad & \forall\, t \in (-1,1) \end{aligned} \tag{3.1}$$

where, $\alpha = 1$ and $\nu = 0.01/\pi$ is the kinematic viscosity. The Burgers equation describes phenomena such as turbulence and the propagation of shock waves in viscous flows. For the reference solution, code [32] is employed, utilizing the Hopf-Cole transformation and the Gauss-Hermite quadrature [33]

The standard and optimized PINNs utilized four fully connected hidden layers with 64 neurons each, $tanh$ as the activation function, and 0.001 as the learning rate. The number of training points for the PDE is $n_{pde} = 5000$, while the initial and boundary condition training points are $n_{ic} = 500 = n_{bc}$. Initially, the ADAM optimizer is applied for 5000 epochs, followed by the L-BFGS optimizer for an additional 500 epochs to achieve further fine-tuning. Total training time for the standard PINN was recorded as 6.07 min using an (NVIDIA T4 GPU) setup.

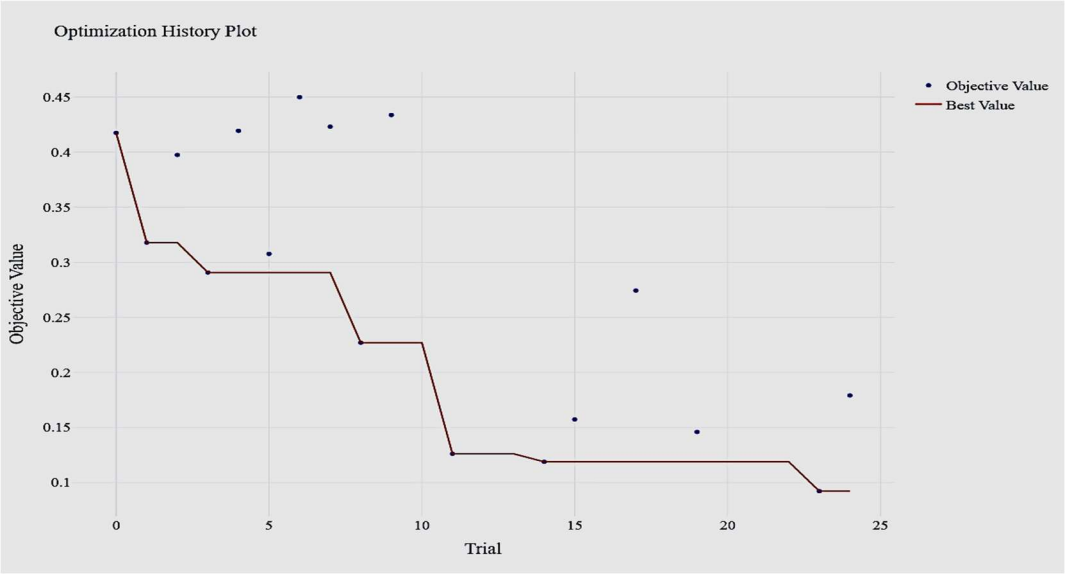
After that, we employ TPE-based Bayesian HPO to seek the systematic selection of optimal PINN configurations. Table 1 shows the range of the network's width and depth, weights associated with the PDE and initial condition, and collocation points for optimization.

**Table 1.** Hyperparameters search space for Burgers equation.

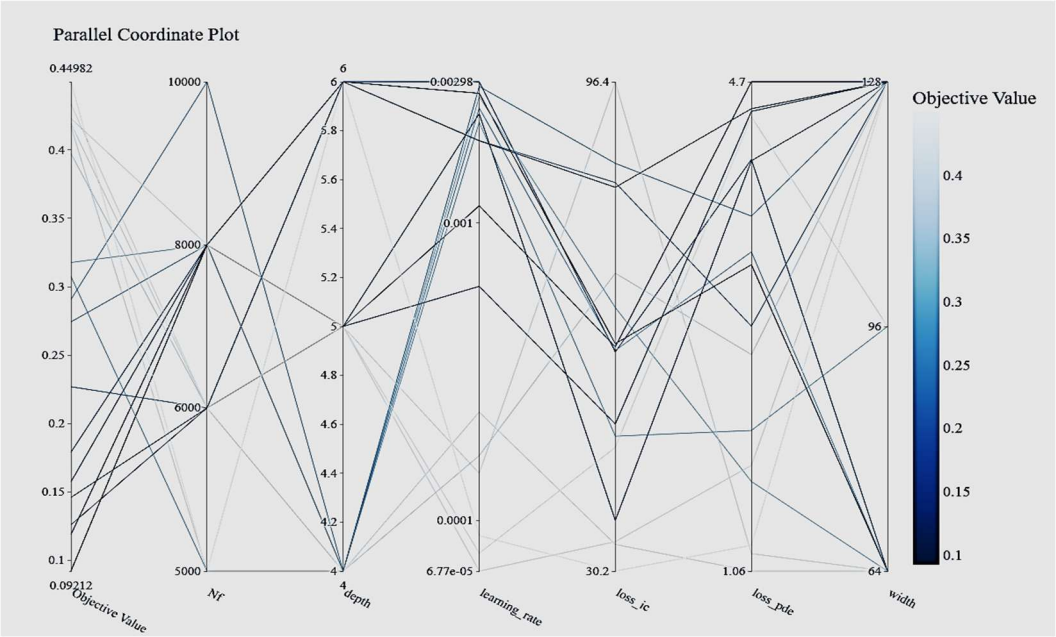
| Hyperparameters | Symbol | Range |
|---|---|---|
| **learning rate** | $\eta$ | $[5e^{-5},\ 3e^{-3}]$ |
| **width** | $w$ | {64, 96, 128} |
| **depth** | $L$ | {4,5 6} |
| **weight PDE** | $\lambda_{pde}$ | [1, 5] |
| **weight initial condition** | $\lambda_{ic}$ | [30, 100] |
| **collocation points** | $n_f$ | {5000,6000,8000,10000} |

Figure 2(a) shows the results of 25 trials against the objective function loss value. We have seen that as iterations proceed, the objective value decreases gradually and the optimal region for hyperparameters lies around trial number 23, where the loss has a minimum value of 0.0921. Figure 2(b)

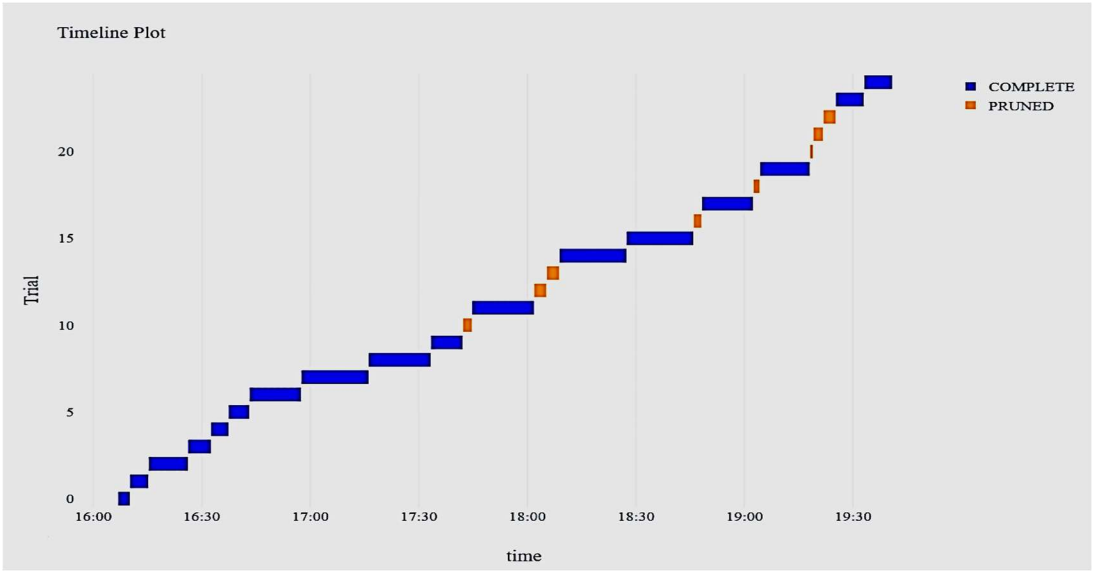
shows the parallel coordinates plots; the optimal values of tuned hyperparameters at trial 23 are $\sigma = 0.00298, w = 64,\ L = 6, n_f = 8000, \lambda_{pde} = 2.6958, \lambda_{ic} = 51.7940$. The computational time for the Optuna trial was recorded as 10.35 min.




(a)



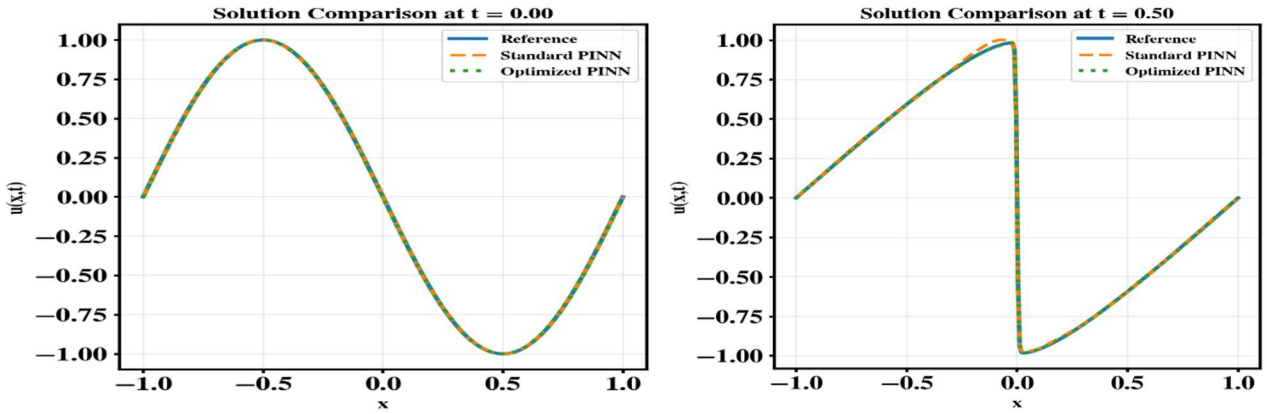

(b)



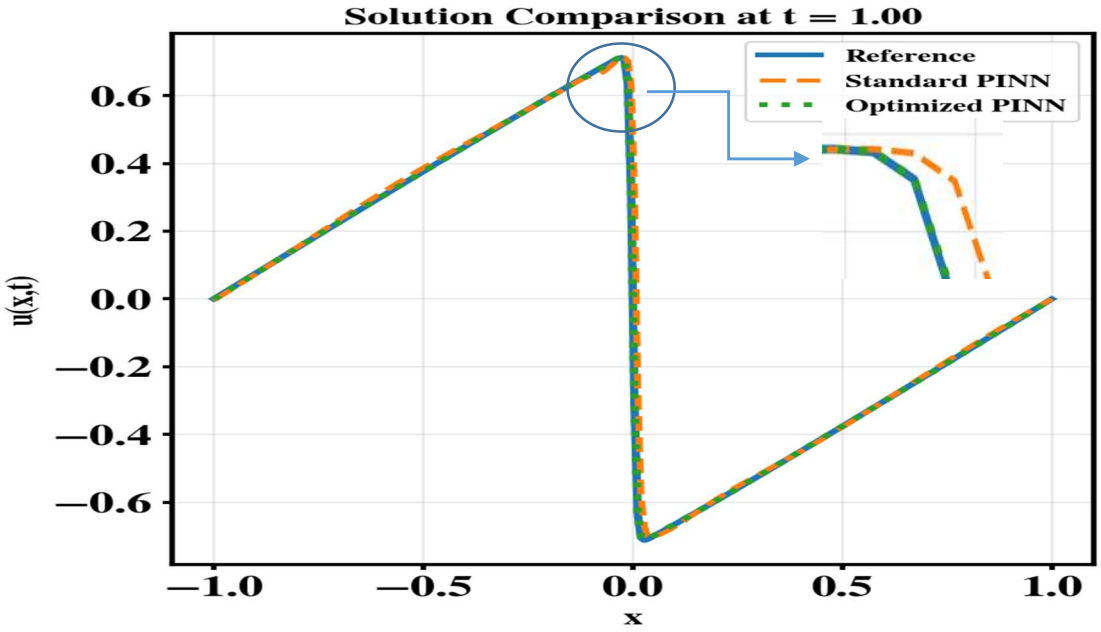

(c)

**Figure 2.** Optuna plots: (a) Optimization history plot, (b) Parallel coordinates plot, (c) Optimization timeline.

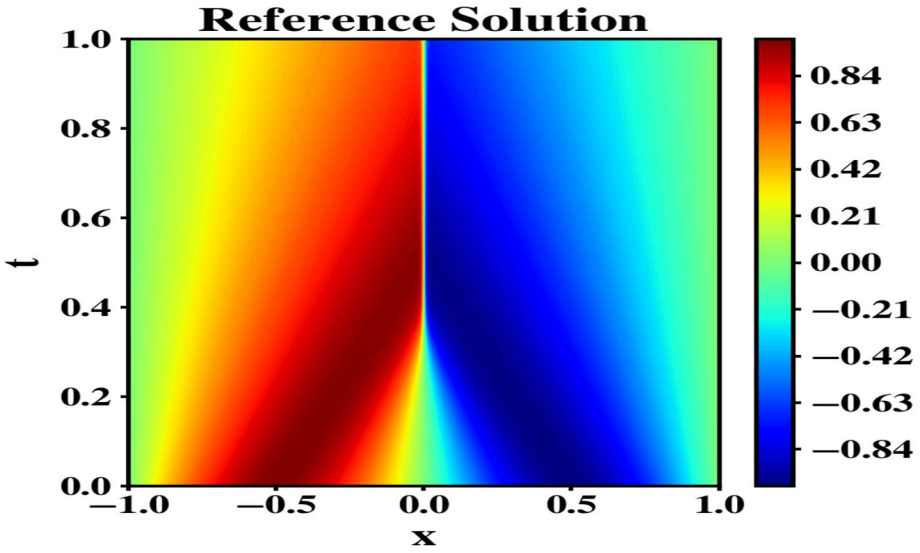
Figure 3(a)–(c) presents the predicted solution profile for both standard and optimized PINNs, alongside the reference solution at $t = 0,\ 0.5,$ and $1.0$. Both standard and optimized PINNs' curves exhibit smooth behavior. However, standard PINN deviate from the reference solution as shown in Figure 3(c). This indicates that systematic selection of hyperparameters with TPE-based HPO produces solutions that are more closely aligned with the reference solution. Figure 3(d)–(f) presents the heatmaps corresponding to the reference, standard, and optimized PINN solution, respectively. Figure 3(g),(h) presents the absolute error plots for the standard and optimized PINN. It can be seen that there is a high error value near t=0 due to discontinuity. The maximum error of the optimized PINN is 0.0135, compared to 0.63 for the standard PINN which is 0.63. These results indicate that the optimized PINN performs better than the standard PINN.




(a) (b)


Solution Comparison at t = 1.00
Reference
Standard PINN
Optimized PINN
u(x,t)
x
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−1.0
−0.5
0.0
0.5
1.0

(c)


Reference Solution
t
x
1.0
0.8
0.6
0.4
0.2
0.0
−1.0
−0.5
0.0
0.5
1.0
0.84
0.63
0.42
0.21
0.00
−0.21
−0.42
−0.63
−0.84

(d)

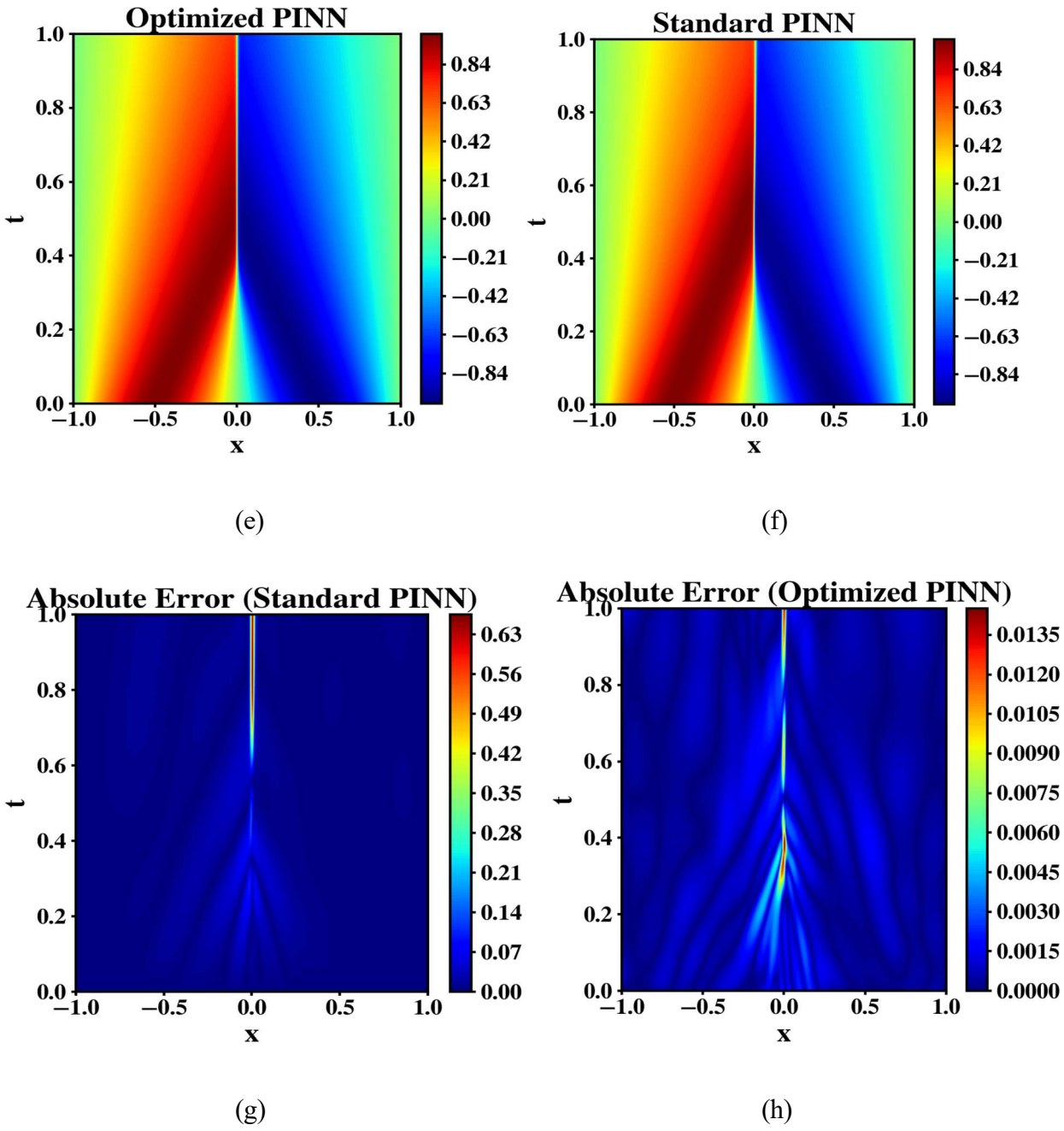
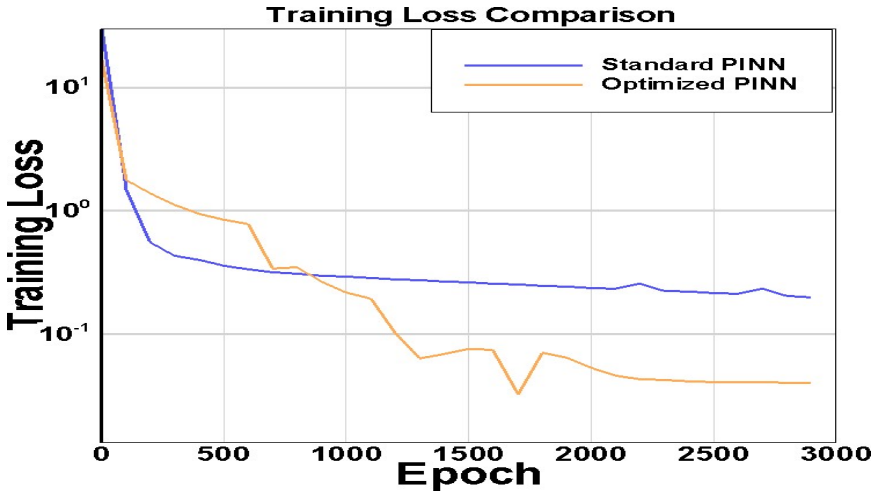


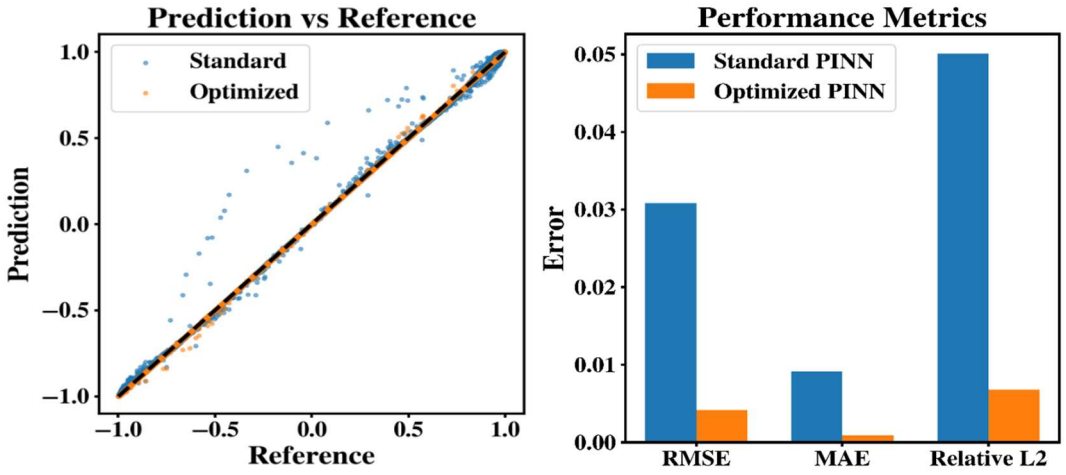

**Figure 3.** (a–c) Solution snapshots of the Burgers' equation at $t = 0, 0.50, 1.0$. Heatmaps of (d) Reference solution, (e) Standard PINN solution, and (f) Optimized PINN solution. (g) Absolute error of the standard PINN. (h) Absolute error of the optimized PINN.

Figure 4(a) presents the training loss curves over 3000 epochs for both standard and optimized PINN. The loss curve of the optimized PINN remains consistently lower than that of the standard PINN. These results show that the hyperparameter tuning using TPE-based HPO enhances the accuracy of the solution. Furthermore, Figure 4(b) presents a comparison between reference and prediction at various time slots. It can be seen that the predicted solution shows good agreement with the reference solution in smooth regions. In contrast, a clear deviation can be seen for the standard PINN. The performance metrics of the optimized PINN are compared with those of the standard PINN in Figure 4(c). Also, Table 2 presents the root mean square error, mean absolute error, and relative L2 error

for both models. The TPE-based optimized PINN demonstrates an 83% improvement in prediction accuracy compared to the standard PINN, despite both models utilizing the same neural network architecture.




(a)




(b) (c)

**Figure 4**. (a) Training loss curves of standard and optimized PINN. (b) Prediction vs reference solution. (c) Performance metrics.

**Table 2.** Model performance metrics.

| Model | Root Mean Square Error (RMSE) | Mean Absolute Error (MAE) | Relative L2 Error |
|---|---|---|---|
| **Standard PINN** | 0.029235424 | 0.007748274 | 0.047519289 |
| **Optimized PINN** | 0.00479796 | 0.001280711 | 0.007798611 |
| **Improvement** | 83.5% | 83.4% | 83.5% |

**Cumulative ablation study**

The importance of each component within the search space is validated through an ablation study as described in Table 3. The effect of incorporating each tuned hyperparameter in TPE-based Bayesian optimized PINN training is assessed by calculating the relative L2 error. We observe that by choosing tuned hyperparameters, the relative L2 decreases from 0.128 to 0.008.

**Table 3.** Ablation study.

| Model | $\lambda_{pde}$ | $\lambda_{ic}$ | $n_f$ | $w$ | $\eta$ | $L$ | L2 error |
|---|---|---|---|---|---|---|---|
| **Standard PINN** | 1 | 50 | 5000 | 64 | 0.001 | 4 | 0.127565 |
| **$\lambda_{pde}$ tuning** | 2.7 | 50 | 5000 | 64 | 0.001 | 4 | 0.018036 |
| **Adding $\lambda_{ic}$ tuning** | 2.7 | 51.8 | 5000 | 64 | 0.001 | 4 | 0.032681 |
| **Adding $n_f$tuning** | 2.7 | 51.8 | 8000 | 64 | 0.001 | 4 | 0.025228 |
| **Adding $w$ tuning** | 2.7 | 51.8 | 6000 | 128 | 0.001 | 4 | 0.010877 |
| **Adding $\eta$ tuning** | 2.7 | 51.8 | 6000 | 128 | 0.002559 | 4 | 0.037595 |
| **Adding $L$= Optimized PINN** | 2.6958 | 51.8 | 6000 | 128 | 0.002559 | 5 | 0.007799 |

*3.2 Test case 2: One-dimensional Allen-Cahn equation*

In the next example, consider the following one-dimensional nonlinear stiff Allen-Cahn equation, subject to periodic boundary conditions and an initial condition defined as below in Eq (3.2).

$$\begin{gathered} u_t = 0.0001u_{xx} - 5(u^3 - u), \quad -1 < x < 1, 0 < t \leq 1 \\ u(-1,t) = u(1,t), \quad u_x(-1,t) = u_x(1,t), \ \forall\, t \in (0,1] \\ u(x,0) = x^2 \cos(\pi x). \quad \forall\, x \in [-1,1]. \end{gathered} \tag{3.2}$$

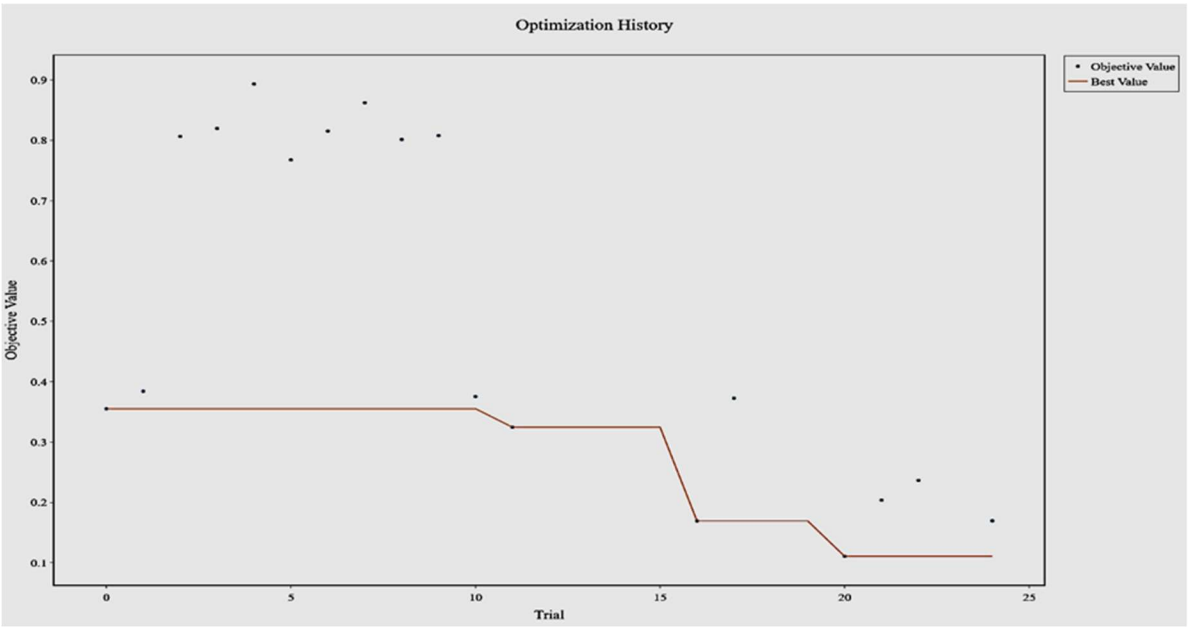
The reference solution of Eq (3.2) is obtained by implementing spectral methods in MATLAB using the Chebfun package [34]. We have considered the standard PINN to solve the test problem given in Eq (3.2). We employ TPE-based Bayesian HPO to seek the optimal PINN configuration. The value of interfacial thickness in Eq (3.2) ($\varepsilon^2 = 0.0001$) is very small, so to handle interfacial dynamics and multiscale spatial features, Fourier features are incorporated as input in the standard PINN and are added to the search space for the optimized PINN. The standard PINN utilized five fully connected hidden layers with 128 neurons each and $tanh$ as the activation function. The number of training points for the PDE is $n_{pde} = 50000$, while the initial and boundary condition training points are $n_{ic} = 512 = n_{bc}$. Further, the weight associated with PDE is 50 and with the initial condition is 100. Initially, the ADAM optimizer is applied for 15000 epochs, followed by the L-BFGS optimizer for an additional 1500 epochs to achieve further fine-tuning. Training time for the standard PINN was recorded as 32.79 min.

We apply TPE-based Bayesian HPO to seek the optimal PINN configuration. Table 4 shows the range of these parameters in the tuning process.

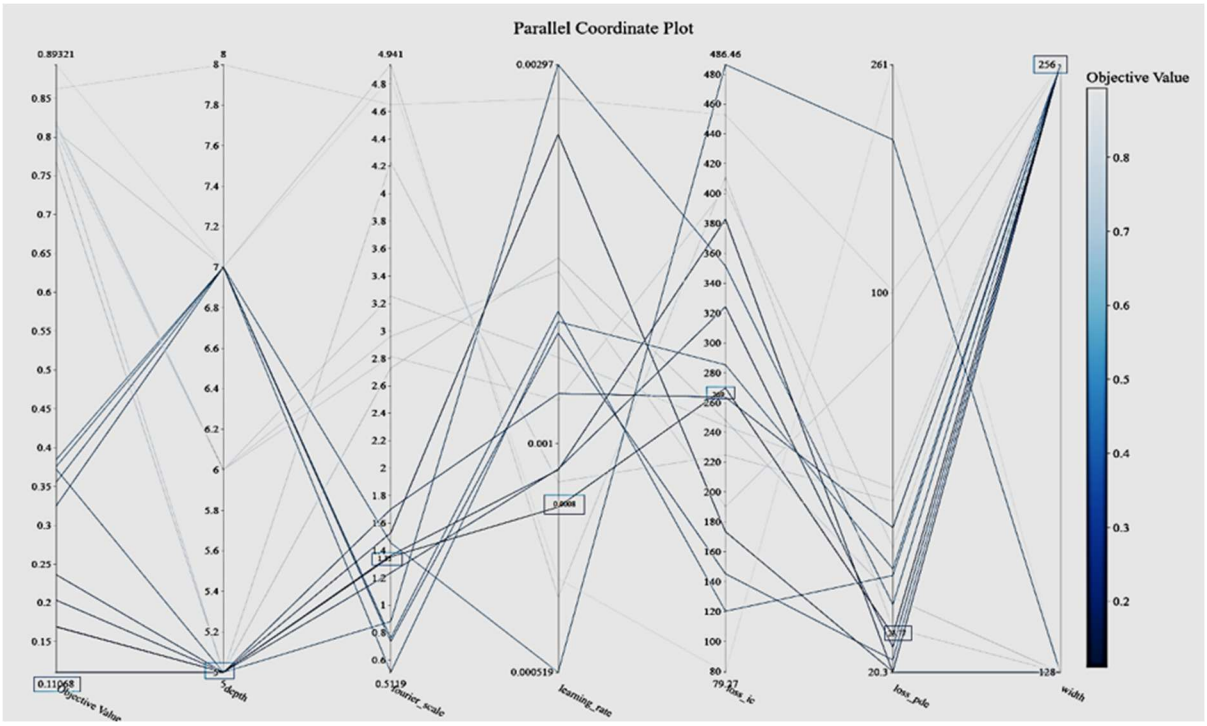
**Table 4.** Hyperparameters search space for the one-dimensional Allen-Cahn equation.

| Hyperparameters | Symbol | Range |
|---|---|---|
| **Learning rate** | $\eta$ | $[5e^{-5},\ 3e^{-3}]$ |
| **width** | $w$ | {64, 128, 256} |
| **Depth** | $L$ | {5,6,7,8, 6} |
| **Weight PDE** | $\lambda_{pde}$ | [20, 300] |
| **Weight initial condition** | $\lambda_{ic}$ | [50, 500] |
| **Fourier scale** | $F_s$ | [0.5, 5.0] |

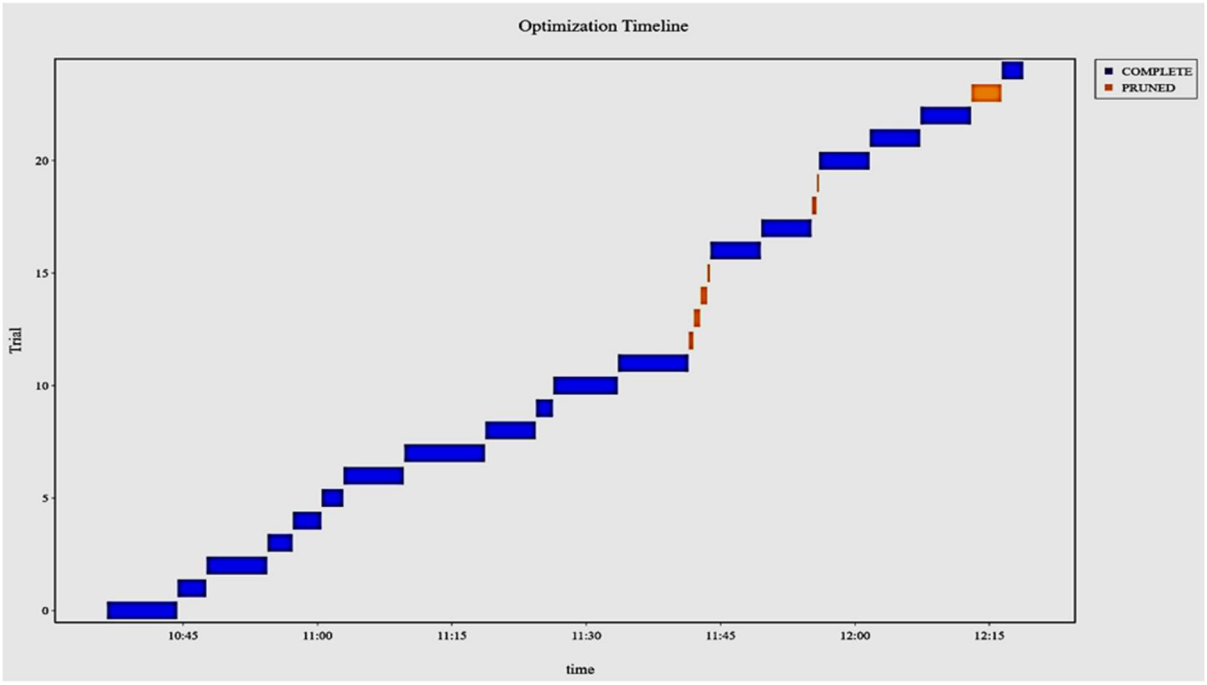
Figure 5(a) shows the results of 25 trials against the objective function loss value in the optimization history plot. We have seen that as iterations proceed, the BO framework balances exploration and exploitation, that will give a decrease in the loss value. From Figure 5(a), the optimal region for hyperparameters lies around trial number 20, where the loss has a minimum value of 0.110677. The relationship between selected hyperparameters and objective function loss is illustrated in the parallel coordinate plots presented in Figure 5(b). Each line represents a distinct trial. Lower loss values are associated at 21 trails with $\eta = 0.0008, w = 256,\ L = 5,\ \lambda_{pde} = 23.77, \lambda_{ic} = 269.77\ and\ F_s = 1.35,$ as shown in Figure 5(b). Optuna trials took 102.29 min to choose the optimal values, as shown in the optimization timeline plots in Figure 5(c).




(a)




(b)

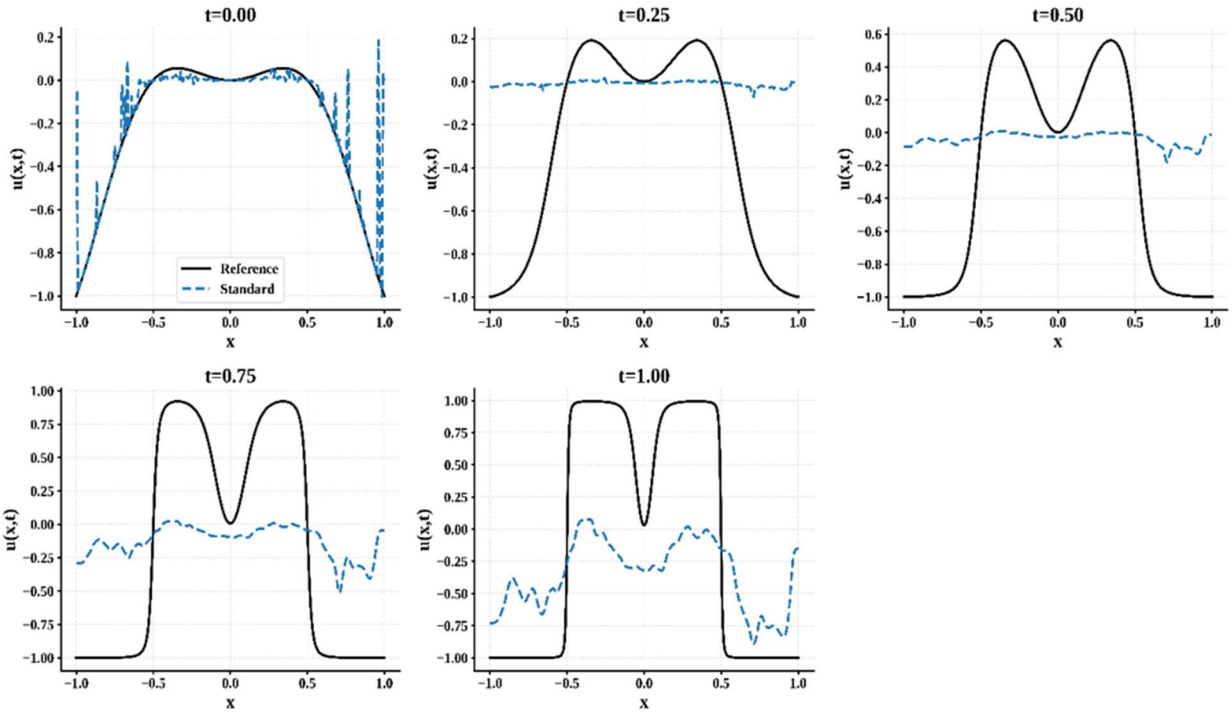



(c)

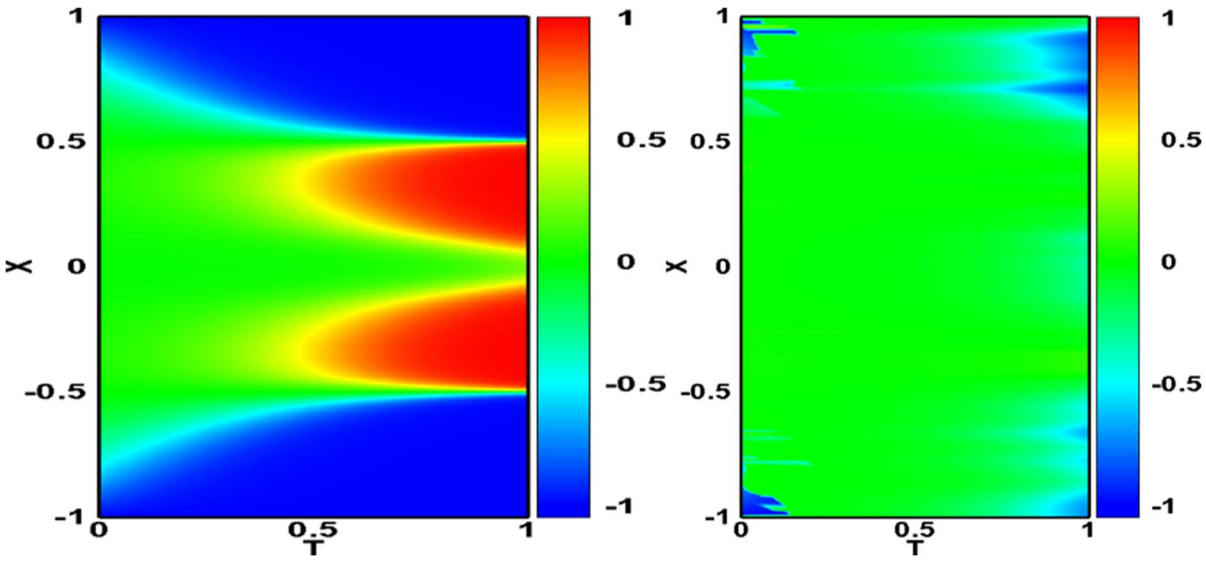
**Figure 5.** Optuna plots for one-dimensional Allen-Cahn equation: (a) Optimization history plot, (b) Hyperparameter importance plot, (c) Optuna completed and pruned timeline plot.

Figure 6(a) presents the solution plots of the Allen-Cahn equation using a standard PINN compared to the reference solution at $t = 0, 0.25, 0.50, 0.75, 1.0$. The standard PINN fails to accurately predict the solution, exhibiting oscillatory behavior at t=0, and degenerating completely for $t > 0$. Both standard and optimized PINN models employ the same network architecture, differing only in hyperparameter tuning. However, the standard PINN does not develop the expected phase separation or sharp interfaces, as reflected in the absolute error plot shown in Figure 6(d). Figure 6(b), (c) show the heatmaps of the reference and standard PINN solutions.

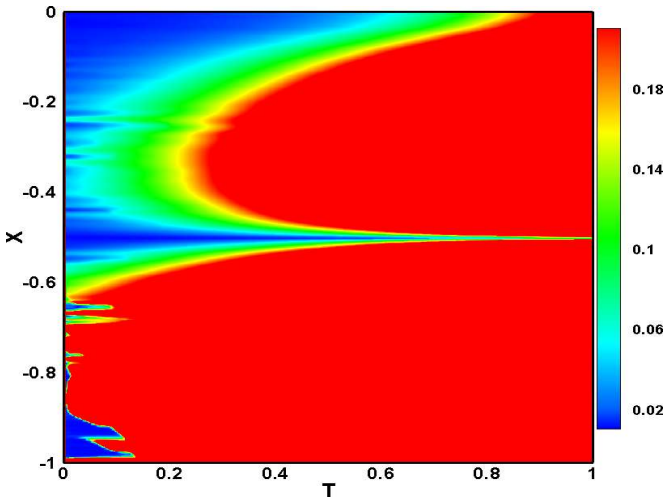
Figure 7(a) represents the solution plots of the Allen-Cahn equation for the optimized PINN compared to the reference solution at t= 0, 0.25, 0.50, 0.75, 1.0. The proposed method demonstrates substantially improved accuracy. This indicates that TPE-based BO enabled the model to solve the Allen-Cahn equation with high numerical accuracy, effectively capturing the spatial and temporal evolution of the order parameter while satisfying all physical, boundary, and initial conditions. Figure 7(d) demonstrates that our model achieved substantially lower errors than standard PINNs, indicating improved predictive performance. Figure 7(b),(c) demonstrate solution accuracy by presenting the heatmaps of both the reference solution and optimized PINN.




(a)

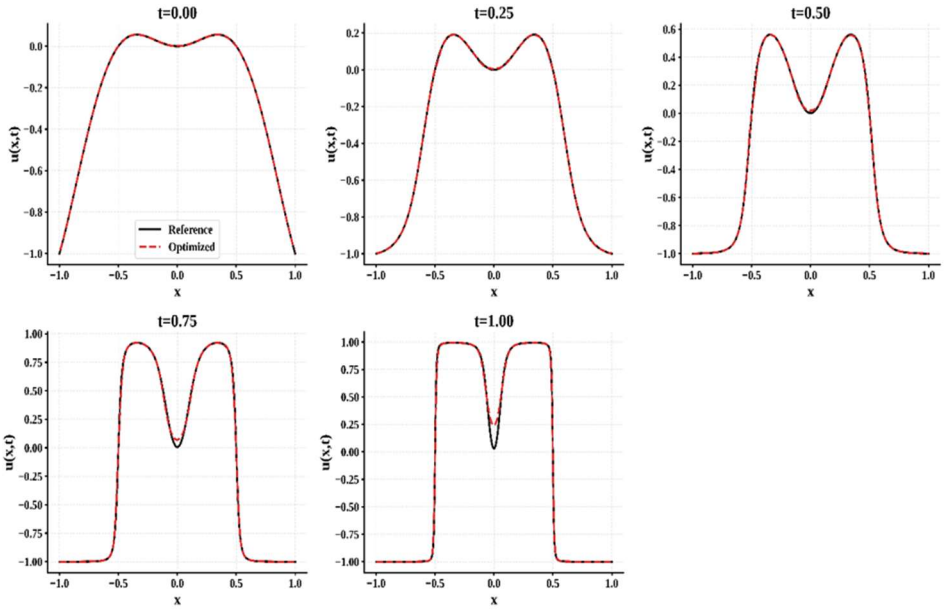



(b) Reference solution. (c) Standard PINN solution.

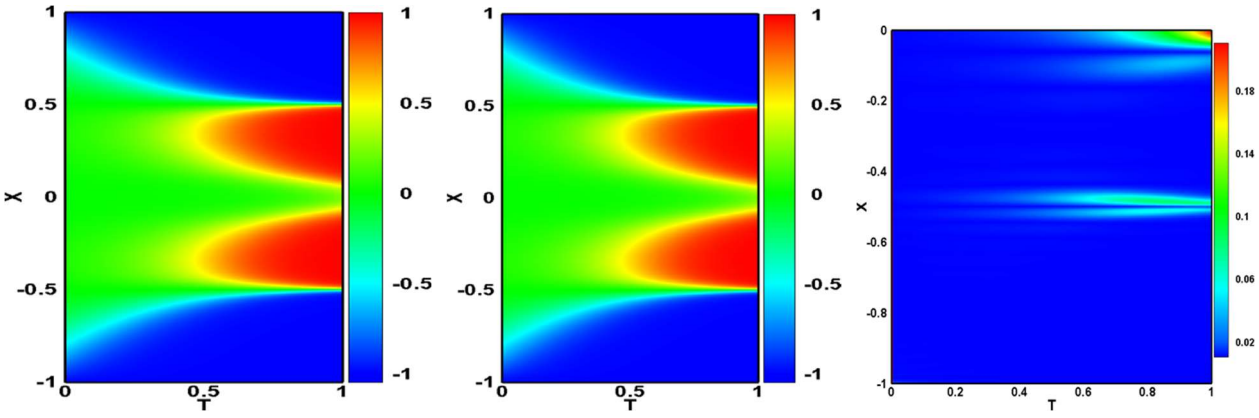



(d) AE standard PINN.

**Figure 6.** (a) Standard PINN solution at t= 0, 0.25, 0.5, 0.75, 1.0. (b) Heatmaps of the reference solution. (c) Heatmaps of the standard PINN solution. (d) Absolute error standard PINN.




(a)

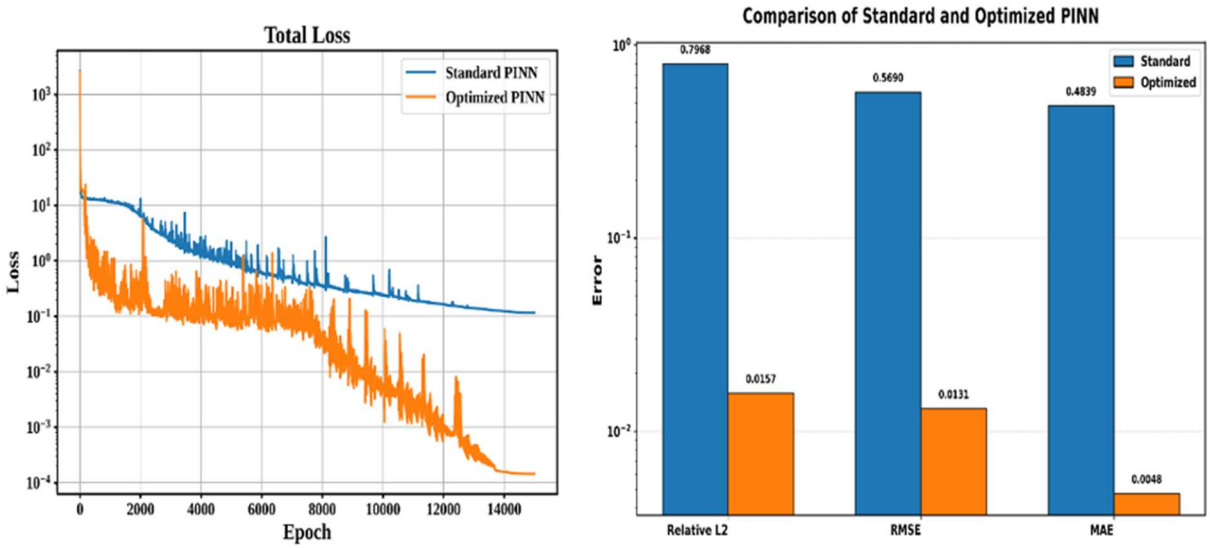



(b) Reference solution. (c) Optimized PINN solution. (d) AE optimized PINN.

**Figure 7.** (a) Optimized PINN solution at t= 0, 0.25, 0.5, 0.75, 1.0. (b) Heatmaps of the reference solution. (c) Heatmaps of the optimized PINN solution. (d) Absolute error optimized PINN.

The training loss curves in Figure 8(a) reveal more stable, accelerated training resulting from automatic hyperparameter tuning, which minimized loss to $10^{-4}$. Figure 8(b) and Table 5 demonstrate the performance metrics of standard and optimized PINNs. It can be seen that by choosing a suitable network architecture, the error is minimized and the performance is optimized.

PINN increases by 98%. Collectively, these results demonstrate that TPE-based BO enhances PINNs' performance by improving accuracy, stability, and convergence efficiency for the Allen-Cahn equation.




(a) (b)

**Figure 8**. (a) Training loss curves of standard and optimized PINN. (b) Performance metrics.

**Table 5.** Model performance metrics.

| Model | Root Mean Square Error (RMSE) | Mean Absolute Error (MAE) | Relative L2 Error |
|---|---|---|---|
| **Standard PINN** | 0.568974 | 0.483915 | 0.7968067 |
| **Optimized PINN** | 0.013090 | 0.004754 | 0.015730 |
| **Improvement** | 97.7% | 99% | 98% |

### *3.3 Test case 3: Two-dimensional elliptic equation*

In our last test case, we consider a two-dimensional nonlinear elliptic equation, subject to boundary conditions defined as below in Eq (3.3)

$$u_{xx} + u_{yy} = f(x,y) + \beta u^3, \qquad 0 < x < 1, 0 < y \leq 1$$

$$u(x,0) = 0 = u(x,1),\ u(0,y) = 0 = u(1,y),\ \forall\, x \in [0,1]. \tag{3.3}$$

Where,

$$f(x,y) = -2\pi^2 \sin(\pi x)\sin(\pi y) - 64\pi^2 \sin(4\pi x)\sin(4\pi y) - \beta[\sin(\pi x)\sin(\pi y) + 2\sin(4\pi x)\sin(4\pi y)]^3.$$

We have constructed $f(x,y)$ from the exact solution for Eq (3.3) given below

$$u(x,y) = \sin(\pi x)\sin(\pi y) + 2\sin(4\pi x)\sin(\pi y).$$

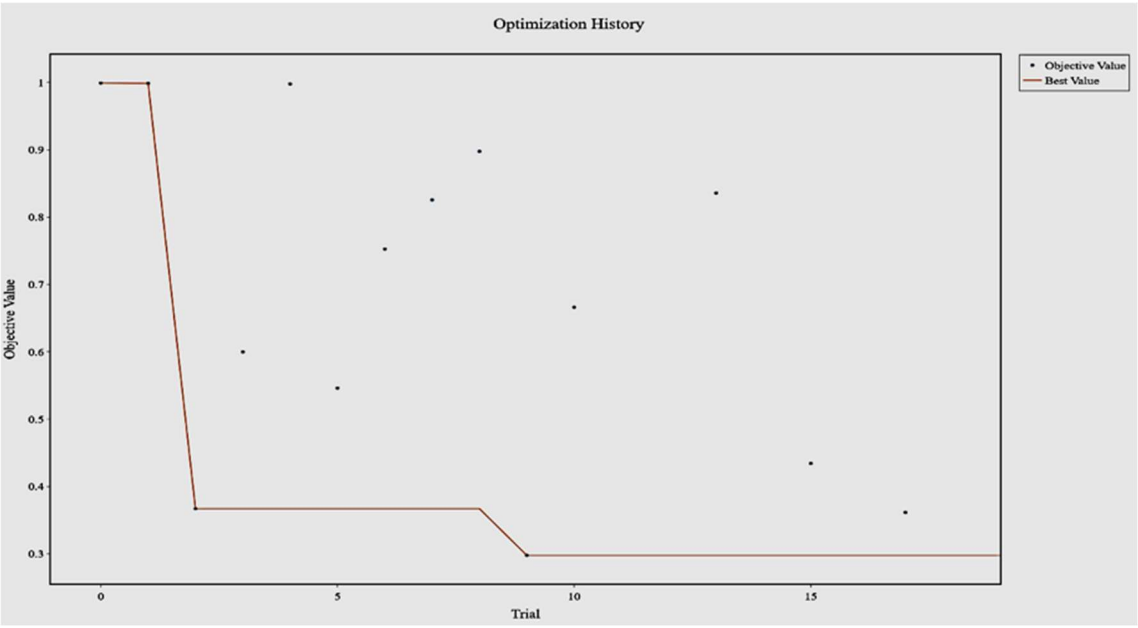
Here $\beta = 20$ controls the degree of nonlinearity. Since the exact solution contain high frequency term so we took Fourier frequencies as inputs in training. Also, we enforced boundary conditions by using hard constraints, so there is no need for a separate boundary loss term for this specific problem. In this example, we implement a standard PINN with five hidden layers, having 28 neurons per layer and $tanh$ as an activation function. The learning rate is 0.001, the number of training samples for the PDE is 5000, and for the boundary condition is 512, respectively. We initially use the ADAM optimizer for 3000 epochs and then use the LBFGS optimizer for finer tuning with 300 epochs. The standard PINN training time is 15.8 min. The TPE-based Bayesian HPO is used to optimize five PINN hyperparameters along with the search space as shown in Table 6.

**Table 6.** Hyperparameters search space for two-dimensional elliptic equation.

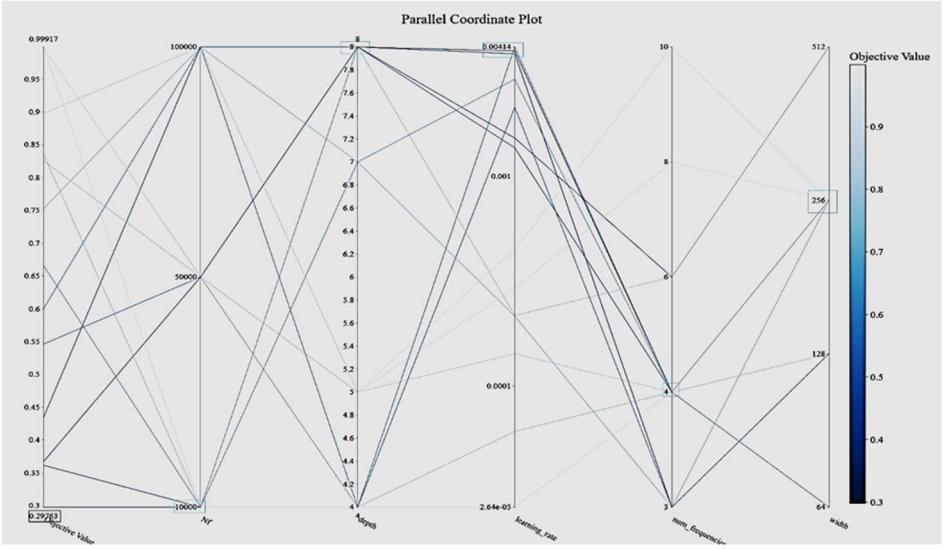
| Hyperparameters | Symbol | Range |
|---|---|---|
| **Learning rate** | $\eta$ | $[e^{-5},\ 3e^{-3}]$ |
| **Network width** | $w$ | {64, 128, 256, 512} |
| **Network depth** | $L$ | {4,5,6,7,8} |
| **Weight PDE** | $\lambda_{pde}$ | [20, 300] |
| **Collocation points** | $n_f$ | {10000,20000,50000,100000} |
| **Fourier frequencies** | $F_f$ | {3, 4, 6, 8,10} |

Optuna trials took 41.16 min, as shown in Figure 9(c). Figure 9(a) shows the results of 20 trials against the objective function loss value. We have seen that as iterations proceed, the BO framework balances exploration and exploitation and gives the optimal region for hyperparameters, which lies around trial number 9, where the loss has a minimum value of 0.29762.

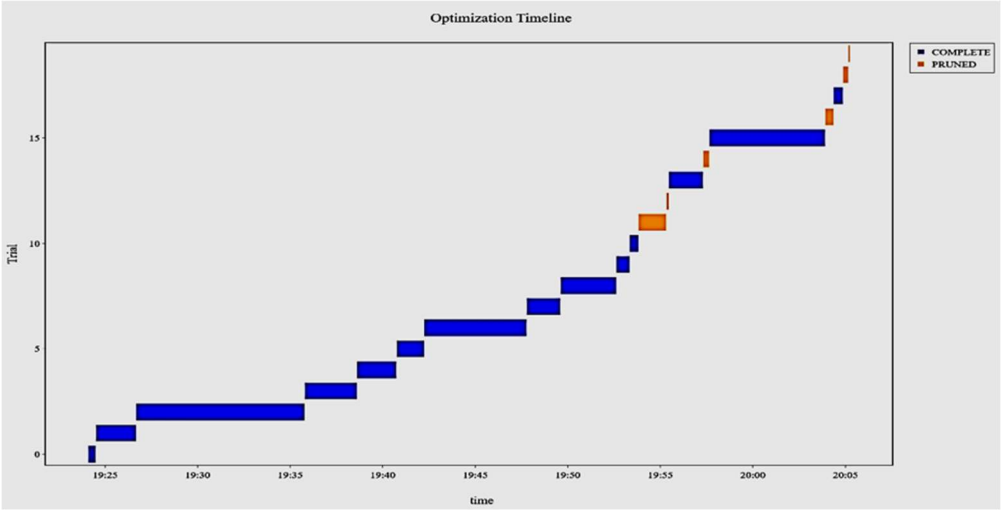
The tuned hyperparameters from Figure 9(b) at trial 9 are $\eta = 0.003959, w = 256,\ L = 8, n_f = 10000, F_s = 4$.




(a)



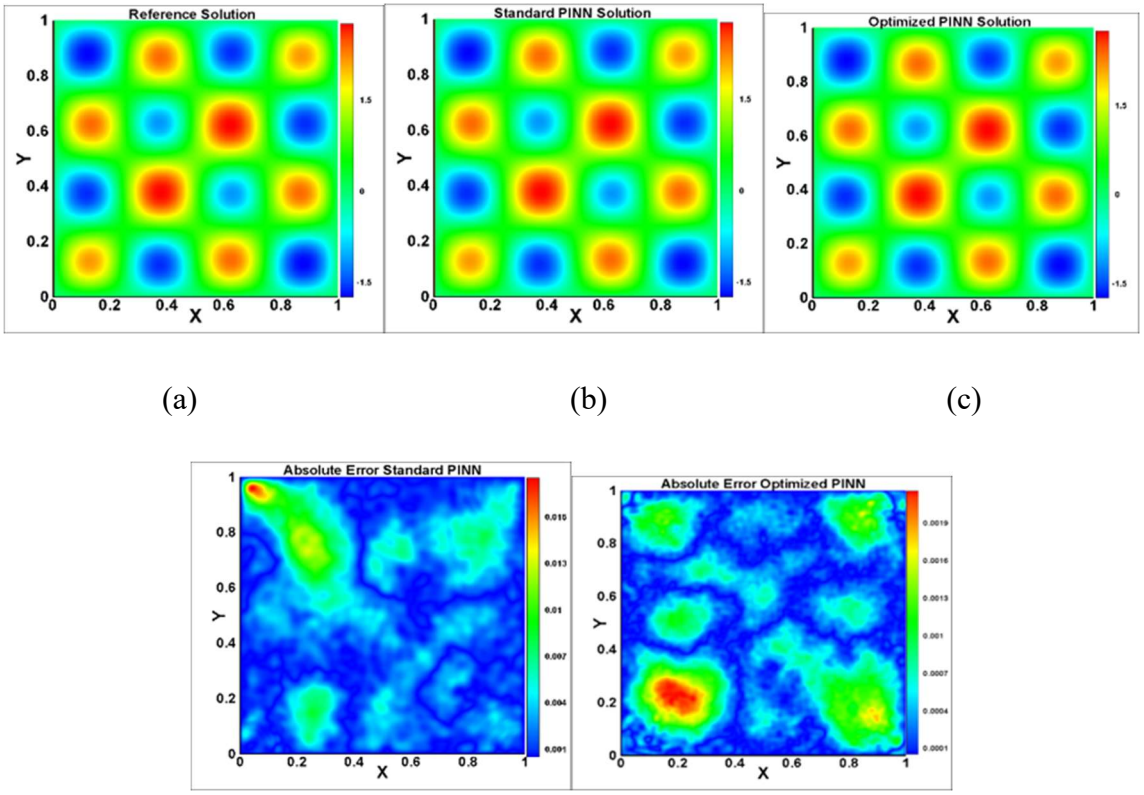

(b)




(c)

**Figure 9.** Optuna plots for 2D elliptic equation (a) Optimization history plot (b) Parallel coordinate plot (c) Optimization timeline.

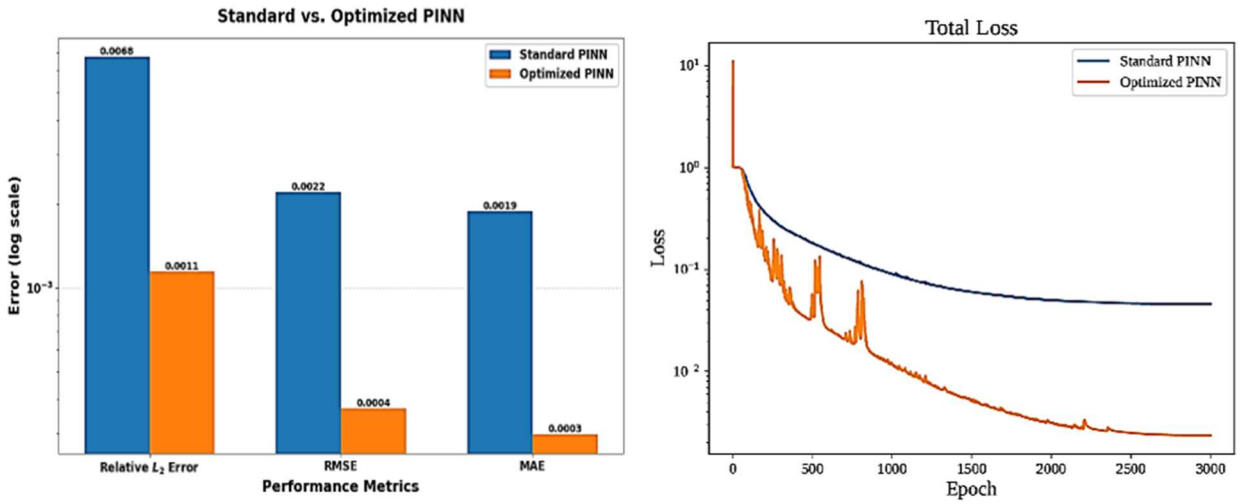
Figure 10(a)–(c) present the contour plots of the predicted solution profile for the nonlinear elliptic equation in two dimensions by using both standard and optimized PINN, alongside the reference solution. Both the reference and optimized PINN solutions closely match each other, whereas the standard PINN shows deviation in the region where the nonlinearity is high. Figure 10(d),(e) present the absolute error plots for the standard and optimized PINN. The performance metrics of the optimized PINN with the standard PINN are shown in Figure 11(a). Table 7 shows that the TPE-based optimized PINN improves the prediction by 83% as compared to the standard PINN, although we apply the same neural network architecture on both models. Figure 11(b) presents the training loss curve of the standard and optimized PINN, indicating that TPE-based Bayesian HPO provides us with a systematic way of tuning hyperparameters.




(a) (b) (c)

(d) (e)

**Figure 10.** Nonlinear elliptic two-dimensional equation: (a) Reference solution, (b) Standard PINN solution, (c) Optimized PINN solution, (d) Absolute error standard PINN, (e) Absolute error optimized PINN.




(a) (b)

**Figure 11.** (a) Performance metrics of standard and optimized PINN (b) Training loss curves of standard and optimized PINN.

**Table 7.** Model performance metrics.

| Model | Root mean square error (RMSE) | Mean absolute error (MAE) | Relative L2 error |
|---|---|---|---|
| **Standard PINN** | 0.0021959 | 0.001878 | 0.003332 |
| **Optimized PINN** | 0.0003689 | 0.000298 | 0.000559 |
| **Improvement** | 83.2% | 84.1% | 83.2% |

## 4. Conclusions

In this work, we employed a TPE-based Bayesian HPO technique. The objective was to improve the ability of PINNs to solve nonlinear, stiff PDEs, thereby increasing the effectiveness of their training. Instead of relying on manual hyperparameter selection, an automated TPE-based Bayesian HPO search was utilized to identify optimal settings for hyperparameters like network width, depth, learning rate, weight associated with PDE and initial condition loss, and Fourier features scale. The result indicates that the systematically tuned PINNs demonstrate greater accuracy and stability, compared to untuned models. Hyperparameter optimization also facilitates the identification of the most influential parameters, thereby informing the design of improved network architectures for nonlinear PDEs. These findings underscore the significance of systematic hyperparameter optimization when applying PINNs to stiff and nonlinear systems. In our future research, we want to extend this approach to evaluate hybrid optimization methods such as combining TPE with early stopping or multi-fidelity search and to compare the current approach with other optimization strategies to reduce the computational cost of repeated training.

## Author contributions

Safeer Hamza Tahir: Conceptualization, methodology, writing original draft, validation, investigation; Sadia Saeed: Conceptualization, software, methodology, validation, visualization, writing review and editing; Reham Alahmadi: Methodology, validation, visualization, review final draft; Saifudin Hafiz Yahaya: Investigation, validation, methodology, visualization, review of the final draft; Umair Ali: Methodology, validation, review final draft, software. All authors read the manuscript and approved it for submission.

## Use of Generative-AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

## Acknowledgments

The authors would like to express their gratitude and appreciation to the Fakulti Teknologi dan Kejuruteraan Industri dan Pembuatan, University Teknikal Malaysia Melaka, 76100 Durian Tunggal, Melaka, for the financial support.

## Conflict of interest

The authors declare they have no conflict of interest.

## References


1. A. H. Ganie, L. H. Sadek, M. M. Tharwat, M. A. Iqbal, M. M. Miah, M. M. Rasid, et al., New investigation of the analytical behaviors for some nonlinear PDEs in mathematical physics and modern engineering, *Part. Differ. Equal. Appl. Math.*, **9** (2024), 100608. https://doi.org/10.1016/j.padiff.2023.100608
2. K. Velten, D. M. Schmidt, K. Kahlen, *Mathematical modeling and simu lation: Introduction for scientists and engineers*, John Wiley & Sons, 2024. https://doi.org/10.1002/9783527849604
3. A. Shah, S. Ayub, M. Sohaib, S. Saeed, S. A. Khan, S. Abbas, et al., An operator splitting scheme for numerical simulation of spinodal decomposition and microstructure evolution of binary alloys, *Heliyon*, **9** (2023). https://doi.org/10.1016/j.heliyon.2023.e16597
4. A. Shah, S. Saeed, S. A. Khan, Numerical investigation of bubbles coales cence in a shear flow with diffuse-interface model, *Heliyon*, **4** (2018). https://doi.org/10.1016/j.heliyon.2018.e01024
5. W. Barsukow, R. Loubere, P. H. Maire, A node-conservative vorticity pre serving finite volume method for linear acoustics on unstructured grids, *Math. Comput.*, **94** (2025), 2299–2343. https://doi.org/10.1090/mcom/4020
6. J. N. Reddy, *An introduction to the finite element method*, In Dynamics of Earth's Fluid System, CRC Press, 2026, 199–226. https://doi.org/10.1201/9781003762133-22
7. S. A. Khan, A. Shah, S. Saeed, Numerical simulation of the interaction between three equal-sized rising bubbles using the phase-field method, *AIP Adv.*, **10** (2020). https://doi.org/10.1063/1.5144963
8. A. Shah, S. Saeed, L. Yuan, An artificial compressibility method for 3D phase-field model and its application to two-phase flows, *Int. J. Comput. Method.*, **14** (2017), 1750059. https://doi.org/10.1142/S0219876217500591
9. Z. Yang, J. Sun, Y. Yang, Y. Chai, Y. Liu, Rapid screening strategy of 2D materials for li-ion battery (lib) electrode based on deep neural networks (DNN), *ACS Appl. Energ. Mater.*, **8** (2025), 3058–3065. https://doi.org/10.1021/acsaem.4c03209
10. Z. Zhou, Y. Wang, G. Zhou, X. Liu, M. Wu, K. Dai, Vehicle lateral dynamics-inspired hybrid model using neural network for parameter identification and error characterization, *IEEE T. Veh. Technol.*, **73** (2024), 16173–16186. https://doi.org/10.1109/TVT.2024.3416317
11. M. D. Kumar, C. S. K. Raju, K. Sajjan, G. Dharmaiah, N. A. Shah, S. J. Yook, Deep neural network-based prediction and computational fluid dynamics analysis of convective heat transfer in dusty fluid flow over heated surface, *Phys. Fluids*, **37** (2025). https://doi.org/10.1063/5.0250396
12. Z. Cai, X. Lin, T. Liu, Physics-informed neural network for estimating surface heat flux from surface temperature measurements, *Int. J. Heat Mass Tran.*, **253** (2025), 127580. https://doi.org/10.1016/j.ijheatmasstransfer.2025.127580
13. D. Zhang, L. Lu, L. Guo, G. E. Karniadakis, Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems, *J. Comput. Phys.*, **397** (2019), 108850. https://doi.org/10.1016/j.jcp.2019.07.048

14. H. Yang, Z. Liu, W. Liu, H. Wang, Y. Zhang, H. Wang, Graph-MDETR: A graph-guided mamba-DETR network for UAV catenary support components detection in electrified railways, *IEEE T. Intell. Transp.*, **27** (2026), 6319–6332. https://doi.org/10.1109/TITS.2026.3680498
15. M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, *J. Comput. Phys.*, **378** (2019), 686–707. https://doi.org/10.1016/j.jcp.2018.10.045
16. R. Mattey, S. Ghosh, A novel sequential method to train physics informed neural networks for Allen Cahn and Cahn Hilliard equations, *Comput. Methods Appl. Mech. Eng.*, **390** (2022), 114474. https://doi.org/10.1016/j.cma.2021.114474
17. H. Sadaf, N. Iftikhar, S. Saeed, Physics-informed neural networks for simulating nonlinear peristaltic flow dynamics: A comparative analysis, *Int. Commun. Heat Mass*, **169** (2025), 109708. https://doi.org/10.1016/j.icheatmasstransfer.2025.109708
18. A. D. Jagtap, G. E. Karniadakis, Extended physics-informed neural networks (XPINNs): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations, *Commun. Comput. Phys.*, **28** (2020), 2002–2041. https://doi.org/10.4208/cicp.OA-2020-0164
19. Z. Zhou, Y. Wang, X. Liu, Z. Li, M. Wu, G. Zhou, Hybrid of neural network and physics-based estimator for vehicle longitudinal dynamics modeling using limited driving data, *IEEE T. Intell. Transp.,* 2025, 1–12. https://doi.org/10.1109/TITS.2025.3585346
20. M. Li, L. Li, Z. Zhang, Q. Zhang, A novel method for generating permutation mass functions using permutations to represent class bias, *Expert Syst. Appl*., **319** (2026), 132004. https://doi.org/10.1016/j.eswa.2026.132004
21. J. Chen, C. Yu, Y. Wang, Z. Zhou, Z. Liu, Hybrid modeling for vehicle lateral dynamics via AGRU with a dual-attention mechanism under limited data, *Control Eng. Pract.*, **151** (2024), 106015. https://doi.org/10.1016/j.conengprac.2024.106015
22. S. Wang, X. Yu, P. Perdikaris, When and why PINNs fail to train: A neural tangent kernel perspective, *J. Comput. Phys.*, **449** (2022), 110768. https://doi.org/10.1016/j.jcp.2021.110768
23. S. Markidis, The old and the new: Can physics-informed deep-learning replace traditional linear solvers?, *Front. Big Data*, **4** (2021), 669097. https://doi.org/10.3389/fdata.2021.669097
24. K. Luo, J. Zhao, Y. Wang, J. Li, J. Wen, J. Liang, et al., Physics informed neural networks for PDE problems: A comprehensive review, *Artif. Intell. Rev.*, **58** (2025), 323. https://doi.org/10.1007/s10462-025-11322-7
25. J. Wu, X. Y. Chen, H. Zhang, L. D. Xiong, H. Lei, S. H. Deng, Hyperparameter optimization for machine learning models based on Bayesian optimization, *J. Electron. Sci. Technol.*, **17** (2019), 26–40. https://doi.org/10.1016/j.jnlest.2020.100007
26. J. Bergstra, Y. Bengio, Random search for hyper-parameter optimization, *J. Mach. Learn. Res.,* **13** (2012), 281305.
27. J. Snoek, H. Larochelle, R. P. Adams, Practical Bayesian optimization of machine learning algorithms, *Adv. Neural Inform. Proc. Syst.*, **25** (2012).
28. C. E. Rasmussen, C. K. I. Williams, *Gaussian processes for machine learning*, MIT Press, 2006. https://doi.org/10.7551/mitpress/3206.001.0001
29. Y. Ramdhani, N. K. Hikmawati, R. T. Prasetio, *Sentiment analysis on the JMO application using random forest with grid search optimization*, In 2025 Tenth International Conference on Informatics and Computing (ICIC), 2025, 1–6. https://doi.org/10.1109/ICIC68054.2025.11309556

30. T. Akiba, S. Sano, T. Yanase, T. Ohta, M. Koyama, *Optuna: A next generation hyperparameter optimization framework*, In Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery data mining, 2019, 2623–2631. https://doi.org/10.1145/3292500.3330701
31. J. Bergstra, R. Bardenet, Y. Bengio, B. Kegl, Algorithms for hyperparameter optimization, *Adv. Neural Inform. Proc. Syst.*, 2011, 24.
32. Y. F. Chen, B. Hosseini, H. Owhadi, A. M. Stuart, Solving and learning nonlinear pdes with gaussian processes, *J. Comput. Phys.*, **447** (2021), 110668. https://doi.org/10.1016/j.jcp.2021.110668
33. T. Ohwada, Cole-Hopf transformation as numerical tool for the Burgers equation, *Appl. Comput. Math*., **8** (2009), 107–113.
34. T. A. Driscoll, N. Hale, L. N. Trefethen, *Chebfun guide*, Pafnuty Publications, Oxford, 2014.




AIMS Press

© 2026 the Author(s), licensee AIMS Press. This is an open access article distributed under the terms of the Creative Commons Attribution License (https://creativecommons.org/licenses/by/4.0)