IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 30 November 2023, accepted 4 January 2024, date of publication 15 January 2024, date of current version 20 February 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3354058

RESEARCH ARTICLE

An Improved Method for Physics-Informed Neural Networks That Accelerates Convergence

LIANGLIANG YAN[1], YOU ZHOU[1], HUAN LIU[2], AND LINGQI LIU[1]

[1]Planetary Science Research Center and School of Computer and Security, Chengdu University of Technology, Chengdu 610059, China
[2]College of Electronic and Information Engineering, Jinggangshan University, Ji'an 343009, China

Corresponding authors: You Zhou (zhouyou06@cdut.edu.cn) and Huan Liu (liuhuan816618@163.com)

This work was supported in part by the National Natural Science Foundation of China Grant 42241142, Grant 42061055, and Grant 41973063; in part by the Sichuan Provincial Natural Science Foundation under Grant 2023NSFSC0278; in part by the Jiangxi Provincial Natural Science Foundation, China, under Grant 20202BABL202047; and in part by the Science and Technology Research Projects of the Jiangxi Province Education Department under Grant GJJ2201608.

ABSTRACT Physics-Informed Neural Networks (PINNs) have proven highly effective for solving high-dimensional Partial Differential Equations (PDEs), having demonstrated tremendous potential in a variety of challenging scenarios. However, traditional PINNs (vanilla PINNs), typically based on fully connected neural networks (FCNN), often face issues with convergence and parameter redundancy. This paper proposes a novel approach that utilizes a multi-input residual network, incorporating a multi-step training paradigm to facilitate unsupervised training. This improved method, which we named MultiInNet PINNs, can enhance the convergence speed and the stability of traditional PINNs. Our experiments demonstrate that MultiInNet PINNs achieve better convergence with fewer parameters than other networks like FCNN, ResNet, and UNet. Specifically, the multi-step training increases convergence speed by approximately 45%, while the MultiInNet enhancement contributes an additional 50%, leading to a total improvement of about 70%. This accelerated convergence speed allows PINNs to lower computational costs by achieving faster convergence. Moreover, our MultiInNet PINNs provides a potential method for handling initial and boundary conditions (I/BCs) separately within PINNs.

INDEX TERMS Physics-informed neural network, partial differential equations, multi-input residual network, convergence speed, unsupervised learning.

I. INTRODUCTION

Various complex natural phenomena are often described by partial differential equations (PDEs) that lack straightforward analytical solutions. To tackle this, numerical methods like Finite Difference Method (FDM) [1], Finite Volume Method (FVM) [2], Smooth Particle Hydrodynamics (SPH) [3], and others have been developed. Regardless of the approach chosen, the core challenge lies in discretizing these equations to solve [4], [5], [6], a process that is time-intensive and significantly affects accuracy. Advances in technology, particularly in GPUs, have enabled the use of finer meshes and smaller particle scales, facilitating more complex discretization schemes. However, the computational workload for intricate simulations remains substantial. Thus, computational efficiency remains a crucial consideration, especially for complex simulations [7].

Deep learning, a transformative paradigm, has displayed remarkable efficacy across diverse domains, including real-time object detection (e.g., YOLO [8]) and image generation (e.g., diffusion models [9]). Its pervasive influence extends beyond computer vision domains, progressively infiltrating disciplines like molecular dynamics and, notably, the realm of differential equations. For instance, Dissanayake and Phan-Thien employed fully connected neural networks to address both the linear poisson equation and the nonlinear

The associate editor coordinating the review of this manuscript and approving it for publication was Tao Huang.

© 2024 The Authors. This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

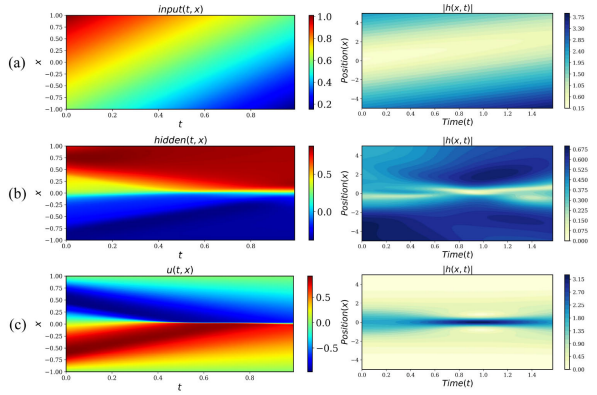


FIGURE 1. PINNs solves the Burger/Schrödinger equation. (a) is the visualization of the shallow PINNs neural network, (b) is the visualization of the neural network after several layers, (c) is the prediction of vanilla PINNs.

heat generation problem [10]. Concurrently, Lagaris et al. have used decomposition strategies to ensure solutions adhere to initial and boundary conditions, effectively using neural networks to address differential equation challenges [11]. However, earlier efforts primarily focused on ubiquitous problems or those abundant in readily available data. Advancements in deep learning, facilitated by frameworks like PyTorch and TensorFlow, alongside advances in automatic differentiation techniques [12], has spawned innovative network architectures [13], [14]. Physics-Informed Neural Network (PINNs) [15] is a prominent example, lauded for its prowess in tackling Partial Differential Equations (PDEs). PINNs' popularity has spurred the development of advanced models [16], [17], [18], [19], [20], including DeepXDE [21], an open-source toolkit by Lu et al. This comprehensive library extends PINNs' capabilities for solving diverse PDE challenges. The continuous refinement of PINNs-inspired strategies underscores their potential to transform PDE problem-solving, evolving the field beyond common or data-rich problems.

Despite the evident advantages of PINNs, they face challenges in training dynamics, notably slow convergence [22]. This issue often stems from intricate interactions among diverse loss terms contributing to the collective training loss. To thoroughly investigate the learning dynamics of network parameters at different depths, we systematically truncate the output of PINNs networks with varying depths. This strategic approach provides insights into their learning trends. For instance, when applying PINNs to solve the Burger equation (as shown in Figure 1 left panel), the analysis of truncated outputs unveils limited boundary learning tendencies in shallower neural network layers. Conversely, deeper layers exhibit progressively enhanced boundary awareness. Expanding our investigation to the Schrödinger equation, as shown in Figure 1 (right panel), confirms the consistent learning trends observed in PINNs. Based on these outcomes, it is plausible to suggest leveraging insights derived from shallower layers to enhance the learning trajectory of deeper layers.

In contrast to the recourse of adopting specialized activation functions to expedite convergence of PINNs [23], [24], [25], our focus extends to the holistic backbone network structure and training methodology. Thus, we introduce multi-input residual network (MultiInNet), inspired by the residual network [26]. This architecture is designed to facilitate enhanced cross-layer information exchange, foster collaborative learning, and tackle the challenge of slow convergence encountered in solving partial differential equations.

The specific contributions we have made can be summarized in the following points:

- We explored the parameter learning of PINNs at various depths through the visualization of truncated predictions from distinct layers within the network architecture. The findings suggest that the deeper layers of the network exhibit a more pronounced capability in effectively assimilating initial and boundary information
- In contrast to PINNs' one-step training that involves all network parameters, we propose a multi-step training approach with fixed parameters. This novel strategy has demonstrated an accelerated convergence rate for PINNs.
- We introduce a novel residual-based backbone network termed the ''multi-input residual network.'' This innovation accelerates the convergence rate of PINNs and concurrently reduces the network parameters compared to the fully connected backbone.

The paper is structured as follows. Section II briefly outlines the fundamental principles of PINNs. In Section III, we provide an overview of the related works pertaining to the convergence and initial/boundary conditions of PINNs. Section IV presents numerical results and comparative experiments of various backbone network PINNs to solve different PDEs. Finally, in Section V, the efficacy of our approach is verified through ablation experiments. Finally, in Section VI, we outline our methodology and briefly address any remaining concerns, highlighting possible avenues for future research. All related code and data are available on https://github.com/konanl/MultiIn-net_PINNs.

II. RELATED WORKS

A. INFLUENCE OF INITIAL/BOUNDARY CONDITIONS(I/BCs)

Physics-Informed Neural Networks (PINNs) is significantly affected by constraints imposed by initial/boundary conditions (I/BCs), often resulting in unstable or slow convergence. Two main strategies have been employed to address this issue. The first strategy involves optimizing weight parameters in the loss function [22], [27], [28], [29]. Typically, priority is given to optimizing PDE residuals during PINNs training. To address this, practitioners often increase the relative weight of the loss associated with I/BCs. This strategic weighting emphasizes I/BCs during backpropagation and can lead to favorable training outcomes. However, determining the precise weighting parameters remains a challenge.

Wang et al. proposed a dynamic adjustment method that monitors the ratio of residuals to I/BCs loss components and updates weights accordingly [22]. The second strategy introduces a regularization term within the loss function to prevent overfitting during training. This regularization component penalizes PDE residuals, preventing excessive optimization that could hinder other loss components during gradient descent. For example, to counteract density-fitted instability in one-dimensional shock tube problems using vanilla PINNs, Liu et al. introduced a velocity gradient penalty term within the loss function's residual segment [30].

B. ACCELERATING THE CONVERGENCE OF PINNs

Efforts to expedite the convergence of PINNs can be grouped into three primary directions. Firstly, research aims to enhance PINNs' initial states by optimizing data sampling methods and initializing network parameters effectively. For instance, Nabian et al. investigated the efficiency of importance sampling [31], while Liu et al. introduced the NRPINN framework, which leverages New Reptile initialization for accelerated training and improved accuracy [32]. Secondly, there is a focus on optimizing the network architecture of PINNs, including designing activation functions and backbone networks to improve convergence speed and overall effectiveness. Lastly, fine-tuning the multi-task loss function constitutes the third approach, often involving careful weighting of various loss components [28] or the use of tailored training strategies to facilitate enhanced convergence [33]. Moreover, Recent studies have demonstrated the application of Bayesian optimization in solving physical systems [39], and numerous studies have utilized Bayesian optimization techniques in optimizing the parameter sets of neural networks [40], thereby striving for accelerated convergence.

III. METHODOLOGY

A. PHYSICS-INFORMED NEURAL NETWORKS

Physics-informed neural network (PINNs) [15] is a high-performance network paradigm for solving a wide range of PDEs. Utilizing advanced deep learning framework differentiation techniques [12], we can systematically construct a problem-specific loss function through automatic differentiation. This approach indirectly encapsulates the underlying functional relationship between the fitted data within defined boundaries, consequently, solving PDEs is transformed into an optimization challenge of network parameters. This transformation enables the indirect solution of PDEs by optimizing neural networks. We consider the general form of a partial differential equation with the following initial and boundary conditions:

$$\begin{aligned} u_t + \mathcal{N}(u_t(x,t);\lambda) &= 0, \qquad x \in \Omega, t \in [0,T] \qquad (1)\\ u(x,0) &= f^{ic}(x,0), \qquad x \in \Omega \\ u(x,t) &= f^{bc}(x,t), \qquad x \in \partial\Omega, t \in [0,T] \qquad (2) \end{aligned}$$

where x and t denote the coordinates in space and time, respectively. Subscripts are employed to indicate partial differentiations, while superscripted delineate distinct functions. The operator $\mathcal{N}[\cdot]$ represents either linear or nonlinear differential operations. Furthermore, Ω signifies a subset within $\mathbb{R}^D$, and $\partial\Omega$ pertains the boundary of Ω.

The primary goal of PINNs is to utilize the neural network denoted as $f(x,t;\Theta)$ to effectively approximate the theoretical solution $u(x,t)$ for a given set of PDEs. This involves the formulation of distinct loss functions rooted in the existing PDEs and I/BCs, and these loss functions, in turn, serve as pivotal optimization targets within the PINNs. Subsequently, under specific I/BCs, PINNs can be used to deliver practical solutions. The residual form of Equation (1) is as follows:

$$\mathcal{R}_\theta(x,t) := \frac{\partial}{\partial t} f(x,t;\theta) + \mathcal{N}[f(x,t;\theta)] \qquad (3)$$

where $\mathcal{R}_\theta(x,t)$ is referred to as the PDEs residual, and this value quantifies the deviation from the actual PDEs, so the aspiration is for its value to approach zero. The training of the PINNs' backbone network is executed through gradient descent, utilizing a general composite loss function formulated as follows:

$$\mathcal{L}(\theta) = \mathcal{L}_r(\theta) + \sum_{i=1}^{N} \lambda_i \mathcal{L}_i(\theta) \qquad (4)$$

where $\mathcal{L}_r(\theta)$ is the loss term to penalize the PDEs residual, and $\mathcal{L}_i(\theta)$ denotes the supervision loss, commonly instantiated through initial conditions, boundary conditions, or supervised data loss (supervised data loss can only be implemented if we have labeled data). In this paper, we exclusively focus on training the model in an unsupervised manner, meaning the absence of labeled datasets. Consequently, our consideration solely pertains to the loss originating from I/BCs, with the general form of these respective loss functions as follows:

$$\begin{aligned} \mathcal{L}_{ic}(\theta) &= \frac{1}{N_{ic}} \sum_{i=1}^{N_{ic}} (\mathcal{I}(x_{ic}^i, t_{ic}^i))^2, \\ \mathcal{I}(x_{ic}^i, t_{ic}^i) &= u(x_{ic}^i, 0) - f^{ic}(x_i, 0) \qquad (5)\\ \mathcal{L}_{bc}(\theta) &= \frac{1}{N_{bc}} \sum_{i=1}^{N_{bc}} (\mathcal{B}(x_{bc}^i, t_{bc}^i))^2, \\ \mathcal{B}(x_{bc}^i, t_{bc}^i) &= u(x_{bc}^i, t_{bc}^i) - f^{bc}(x_i, t_i) \qquad (6) \end{aligned}$$

for the PDEs residual loss, the form is as follows:

$$\mathcal{L}_r(\theta) = \frac{1}{N_r} \sum_{i=1}^{N_r} (\mathcal{R}(x_r^i, t_r^i))^2 \qquad (7)$$

where $\{(x_{ic}^i, 0)\}_{i=0}^{N_{ic}}$,$\{(x_{bc}^i, t_{bc}^i)\}_{i=0}^{N_{bc}}$ denote the sampling points for initial and boundary conditions, respectively. Furthermore, $\{(x_r^i, t_r^i)\}_{i=0}^{N_r}$ represents randomly selected training points within the domain Ω. Notably, utilizing these sampled points, Equation (5) (6) (7) correspond to the initial condition

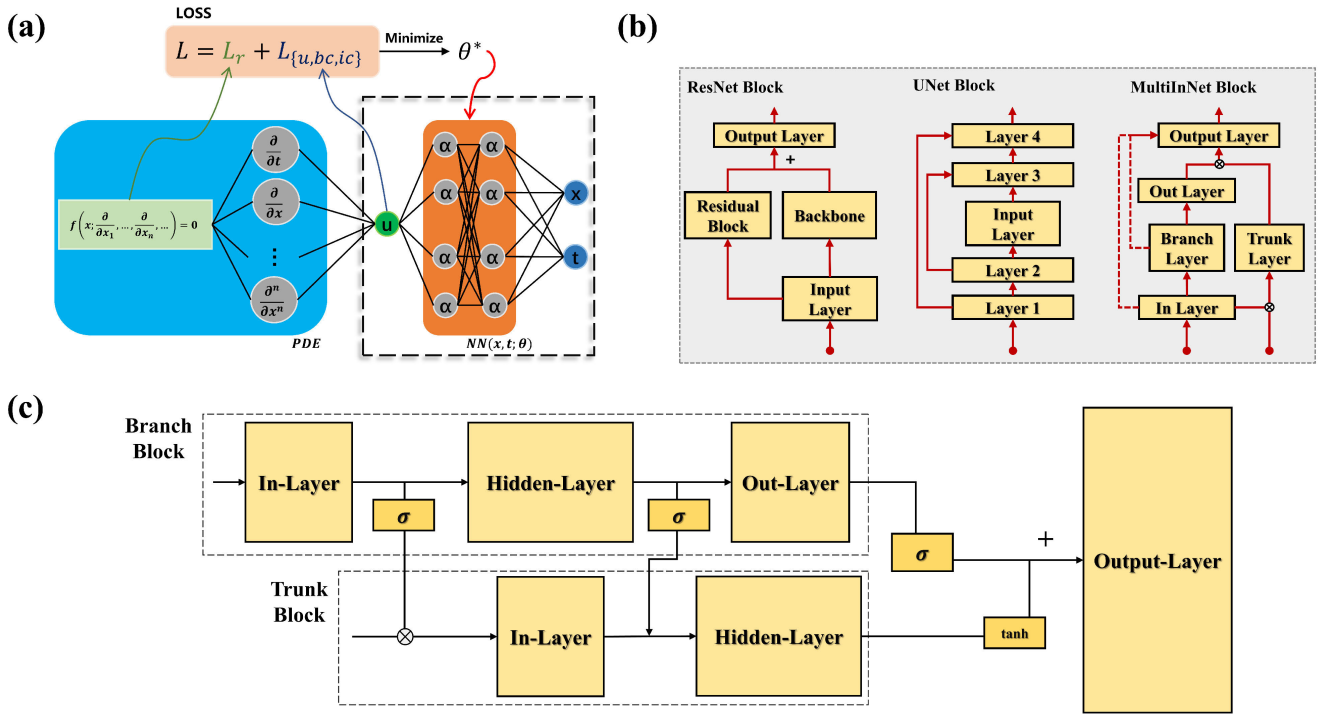


FIGURE 2. (a) is PINNs network model schematic, (b) is three types of backbone network diagrams of PINNs, left to right: Resnet, UNet, MultiIn-net, (c) is the backbone network diagrams of MultiIn-net with correction.

loss, boundary condition, and PDEs residual, respectively. Ultimately, in the training process of the PINNs, the objective is to minimize the composite loss as much as possible, which takes the following form:

$$\mathcal{L}(\theta) = w_r * \mathcal{L}_r(\theta) + w_{ic} * \mathcal{L}_{ic}(\theta) + w_{bc} * \mathcal{L}_{bc}(\theta) \quad (8)$$

where w_r, w_{ic}, w_{bc} symbolize the weights attributed to the three aforementioned loss functions, respectively. Common practice often entails judiciously enhancing the weights of I/BCs loss functions, but this approach is eschewed in this paper, where the weights are uniformly maintained at one, with the exception of vanilla PINNs. The overarching conceptual framework of PINNs is illustrated in Figure 2 (a).

B. BACKBONE NETWORK OF PINNS

Currently, most PINNs employ fully connected networks as backbone, risking overfitting [34] and parameter redundancy. In response, alternative backbone architectures, such as ResNet, have been explored. We propose that crucial initial and boundary conditions information is primarily captured in deeper layers. Utilizing this insight, we enhance shallower layers' parameter learning, aiming to optimize convergence and overall performance.

In this paper, we compare our proposed PINNs backbone network with Resnet and UNet, focusing on parameters and convergence speed. As depicted in Figure 2 (b), we illustrate the network structure of ResNet, UNet, and our MultiInNet, which inspired by residual networks and stood out with dual or multiple input interfaces. This design aims to allow information from deeper layers to influence shallower ones, akin to an inverted residual architecture. To meet PDE targets using both collocation and initial/boundary condition samples [31], we employ multiple pairs of residual connections, differing from ResNet's single connection, this strategy integrates information from various deep-layer network parameters. Furthermore, further modified to filter invalid information, ensuring information exchange promotes gradient descent, we modify the network and further screen residual connections, and the specific structure is shown in Figure 2 (c).

C. TRAINING STRATEGY OF PINNS

In the initial stages of PINNs training, achieving a balanced weighting between loss terms is crucial to prevent instability and hindered convergence, especially with complex I/BCs [35]. Several methodologies in related studies employ specific training methods, notably greedy algorithms [36], especially for training shallow networks, as opposed to the conventional gradient descent algorithms. It's important to note that this paper exclusively employs the Adam optimization algorithm [37] across all methodologies, without delving into the exploration of specialized training algorithms. Moreover, traditional PINNs training involves optimizing all network parameters simultaneously. However, we introduce a modification in this paper. We employ a multi-step strategy where subsets of network parameters are optimized at different stages. For instance, deep network modules are pre-trained early in the process, followed by individual optimization of distinct network layers according

to network type. Finally, we fine-tune the overarching parameters.

IV. NUMERICAL EXPERIMENTS

In this section, we perform numerical experiments to validate our method against alternative approaches. For vanilla PINNs, default training settings were utilized, comprising 8000 training steps, Adam optimizer with a learning rate of 10e-3, and a network structure of 4*20 (layers*neurons). The training sampling points were systematically generated using the Sobol sequence. Specifically, 1000 points were sampled for the residual points, whereas the initial and boundary conditions were set at half the scale of the sampling points. From a fundamental perspective, the rationale behind having fewer sampling points for the initial boundary conditions compared to the residual points is justified. All three network architectures (ResNet, UNet, and our enhanced MultiInNet) follow the multi-step training strategy with consistent training hyperparameters across various examples. Notably, all experiments were conducted utilizing the GPU RTX 3060.

We rigorously assessed the training complexity of diverse network architectures by calculating both the average step training time and the total training time. This evaluation aimed to substantiate the validation of assessing convergence speed based on the convergence step. The results of these assessments are graphically depicted in Figure 3. To ensure the robustness and reliability of our findings, we conducted ten training iterations. Within the presented Figure 3, the solid line represents the total training time, while the dashed line illustrates the duration of each individual training step.

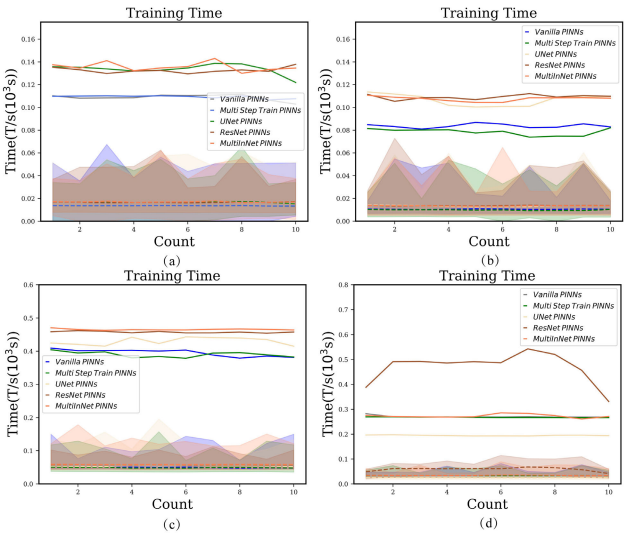


FIGURE 3. Statistical curve of training time for experiments: (a), (b), (c), (d) are poisson equation, advection equation, convention-diffusion equation and helmholtz equation respectively.

Moreover, with the exception of the corrected version of MultiInNet, all examples employ the $tanh(\cdot)$ activation function, which is widely used within the domain of PINNs. In our experimental setup, a larger number of sample points are allocated to the test set as compared to the training set. The evaluation of the model is undertaken with the mean square error (MSE) and the L_2-error, both of which serve as metrics for gauging the performance of the trained model. The expressions for these metrics are as follows:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}|u(x_i,t_i)-\hat{u}(x_i,t_i)|^2 \tag{9}$$

$$L_2-error = \frac{\sqrt{\sum_{i=1}^{N}|u(x_i,t_i)-\hat{u}(x_i,t_i)|^2}}{\sqrt{\sum_{i=1}^{N}|u(x_i,t_i)|^2}} \tag{10}$$

where u signifies the reference solution, $\hat{u}$ denotes the predictions of neural networks. The Mean MSE and L_2-error of all examples are presented in Table 1. Moreover, we compare the magnitudes of various network parameters and the outcomes of which are detailed in Table 2.

A. POISSON EQUATION

For the first example, consider the Poisson equation [38], [41], which has the following general form,

$$-\frac{\mathrm{d}^2u}{\mathrm{d}x^2} = f(x), \qquad x \in [-L, L] \tag{11}$$

and the boundary conditions as defined as $u(-L) = u(L) = 0$. To maintain simplicity, we set the exact solution as $u(x) = sin(x^2)$.

In this example, we exclusively consider the PDE residual and boundary condition loss. The vanilla PINNs use a weight ratio $\frac{w_r}{w_{bc}}$ of 0.1, while others maintain a ratio of 1. Regarding the training strategy, exclusively the deep network parameters are engaged in training until reaching 2000 epochs. Subsequently, solely the shallow network parameters are involved in training from 2000 to 4000 epochs, their loss functions' convergence is shown in the left side of Figure 4. Early-stage instability in the boundary condition loss may stem from freezing shallow network parameters, but overall loss convergence accelerates. Additionally, we modify PINNs backbone networks into UNet, ResNet, and MultiInNet, achieving notable convergence benefits compared to vanilla PINNs' fully connected networks, with ResNet and MultiInNet showing convergence tendencies around 1000 epochs.

B. ADVECTION EQUATION

Advection stands as a critical process within atmospheric motion, with the governing equations incorporating an advection term. We consider advection equation as follows:

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = 0, \qquad x \in [0,1], t \in [0, 0.5] \tag{12}$$

and its I/BCs are the following:

$$\begin{aligned} u(0,x) &= 2sin(\pi x) \\ u(t,0) &= -2sin(\pi t),\ u(t,1) = 2sin(\pi t) \end{aligned} \tag{13}$$

TABLE 1. Statistics of mean MSE of different PINNs.

Mean MSE	Vanilla	Multi-Train	UNet	ResNet	MultiInNet
Poisson equation	3.27E-03	5.72E-04	4.73E-04	4.93E-04	$\mathbf{9.73E-05}$
Advection equation	2.87E-05	9.30E-06	2.50E-06	4.48E-06	$\mathbf{1.41E-06}$
Convection-diffusion equation	1.97E-02	1.920E-02	1.85E-02	5.97E-03	$\mathbf{5.14E-03}$
Helmholtz equation	1.74E-02	1.790E-02	1.19E-02	6.11E-02	$\mathbf{5.57E-03}$

TABLE 2. Statistics of L_2-error of different PINNs.

L_2-error	Vanilla	Multi-Train	UNet	ResNet	MultiInNet
Poisson equation	8.65E-02	3.62E-02	3.58E-02	3.38E-02	$\mathbf{1.49E-02}$
Advection equation	3.79E-03	2.12E-03	1.12E-03	1.50E-03	$\mathbf{8.42E-04}$
Convection-diffusion equation	9.84E-01	9.72E-01	9.52E-01	5.40E-01	$\mathbf{5.01E-01}$
Helmholtz equation	2.67E-01	2.69E-01	2.49E-01	4.99E-01	$\mathbf{1.51E-01}$

TABLE 3. Statistics of network parameters.

Numbers of Params	Vanilla	UNet	ResNet	MultiInNet
Poisson equation	1341	1723	1341	**582**
Advection equation	1341	951	1341	**663**
Convection-diffusion equation	1341	1221	1341	**1083**
Helmholtz equation	5301	1531	1761	**663**

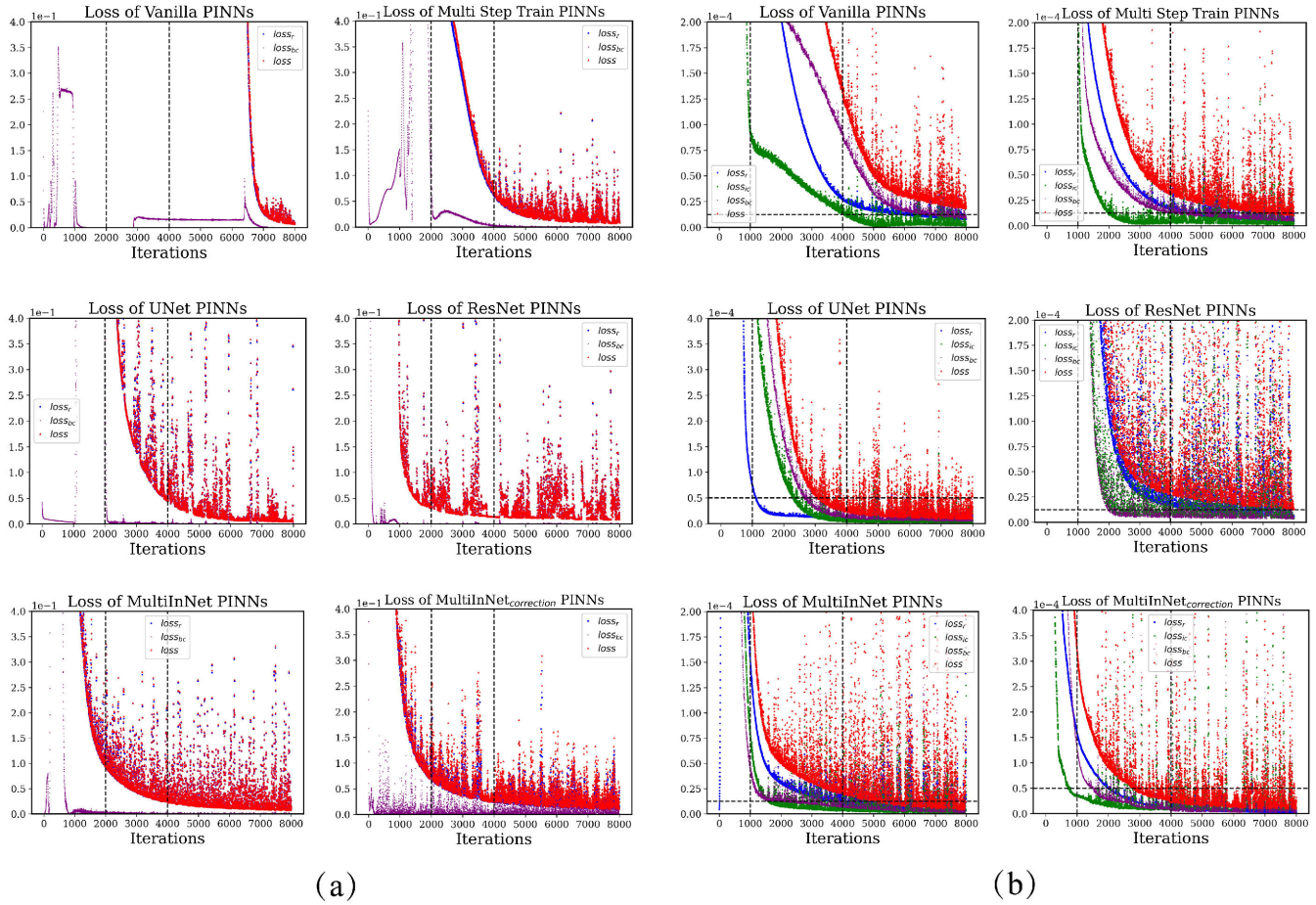


FIGURE 4. Loss decay curve for different PINNs to solve the Poisson/Advection equation: (a) shows the loss decay curve of the Poisson equation, (b) shows the loss decay curve of Advection equation.

and the exact solution of the advection equation in this example is $u(t,x) = 2sin(\pi(x-t))$.

We need to consider three loss components (PDE residual, initial condition, boundary condition) with weight ratios of $w_r : w_{ic} : w_{bc} = 1 : 10 : 10$ for vanilla PINNs, while others maintain weights at 1. The loss convergence is depicted in the right side of Figure 4. Notably, the significant total loss convergence starts when the I/BCs loss converge to a certain extent, underscoring their crucial role in the PINNs backbone network's convergence. Similarly, our backbone's

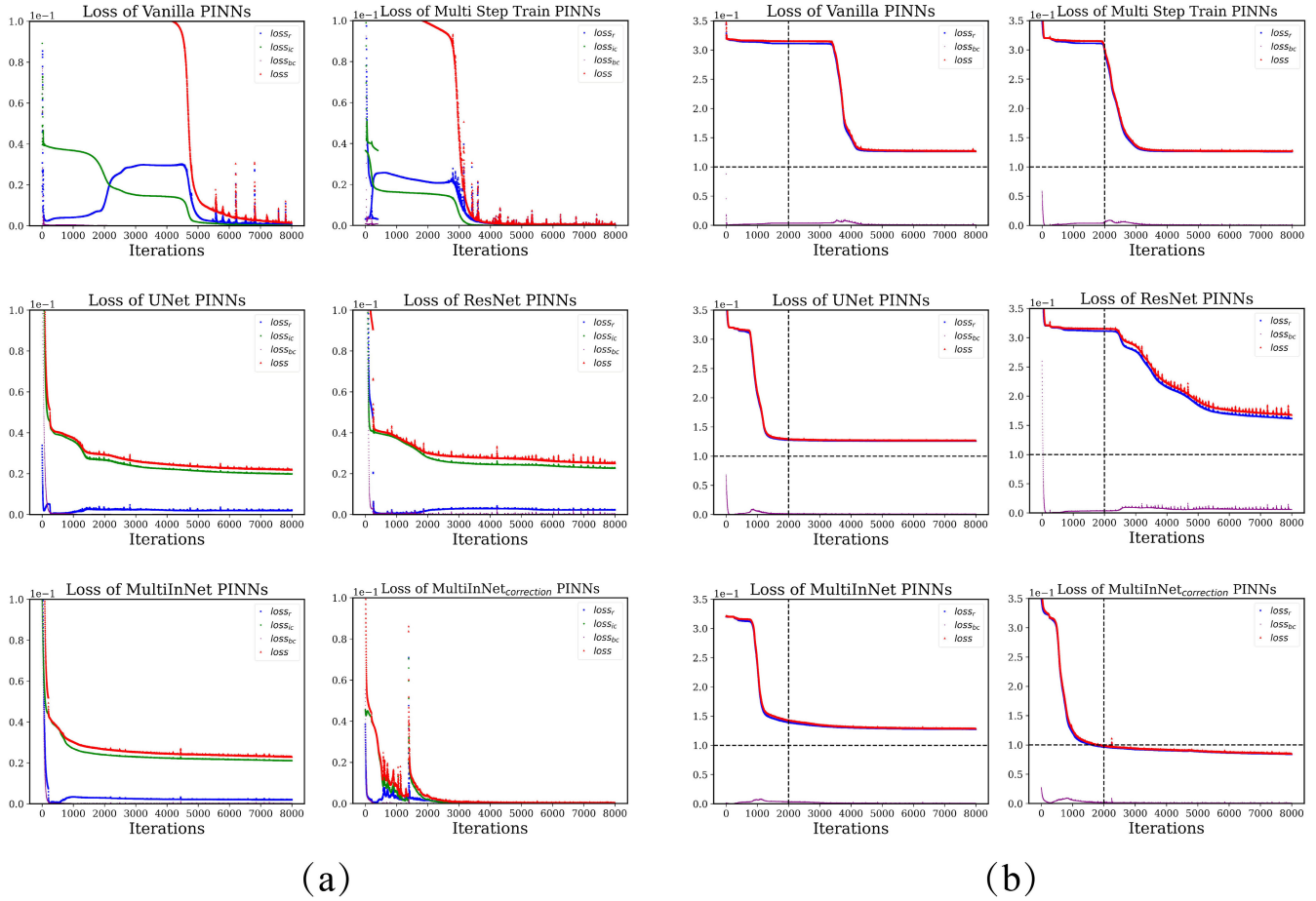


FIGURE 5. Loss decay curve for different PINNs to solve the Convention-diffusion/Helmholtz equation: (a) shows the loss decay curve of the Convention-diffusion equation, (b) shows the loss decay curve of Helmholtz equation.

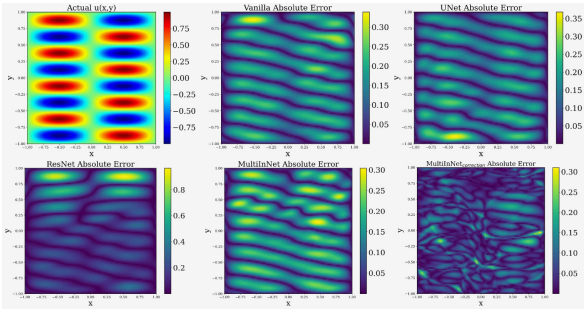


FIGURE 6. Comparison of absolute error in the Helmholtz equation when k is 10.

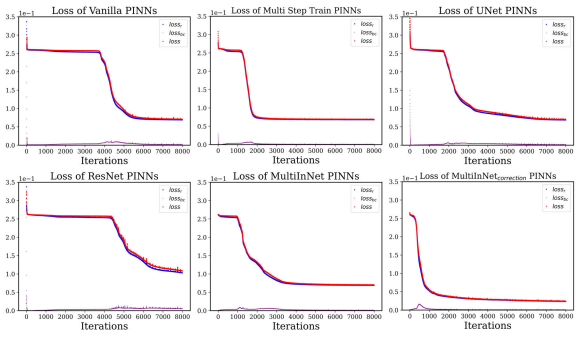


FIGURE 7. Comparison of loss convergence in the Helmholtz equation when k is 10.

I/BCs loss exhibits faster convergence, contributing to the overall accelerated convergence.

C. CONVENTION-DIFFUSION EQUATION

The convection-diffusion equation, a fundamental equation of motion with wide applicability in various domains, represents a linear partial differential equation. Here, we address the convection-diffusion equation in the following form,

$$\frac{\partial u}{\partial t} + c \cdot \frac{\partial u}{\partial x} = \mu \cdot \frac{\partial^2 u}{\partial x^2}, \qquad x \in [-L, L], t \in [0, T] \quad (14)$$

furthermore, we consider the following I/BCs,

$$u(0, x) = \frac{0.1}{\sqrt{0.1 \cdot \mu}} exp(-\frac{(x+2)^2}{4 \cdot 0.1 \cdot \mu})$$
$$u(t, -L) = u(t, L) = 0 \quad (15)$$

and we use the following parameter configuration $c = 4.0, \mu = 0.05, L = 4, T = 1$.

The initial conditions in this instance exhibit greater complexity compared to prior cases, potentially leading to more challenging convergence. The results illustrate a

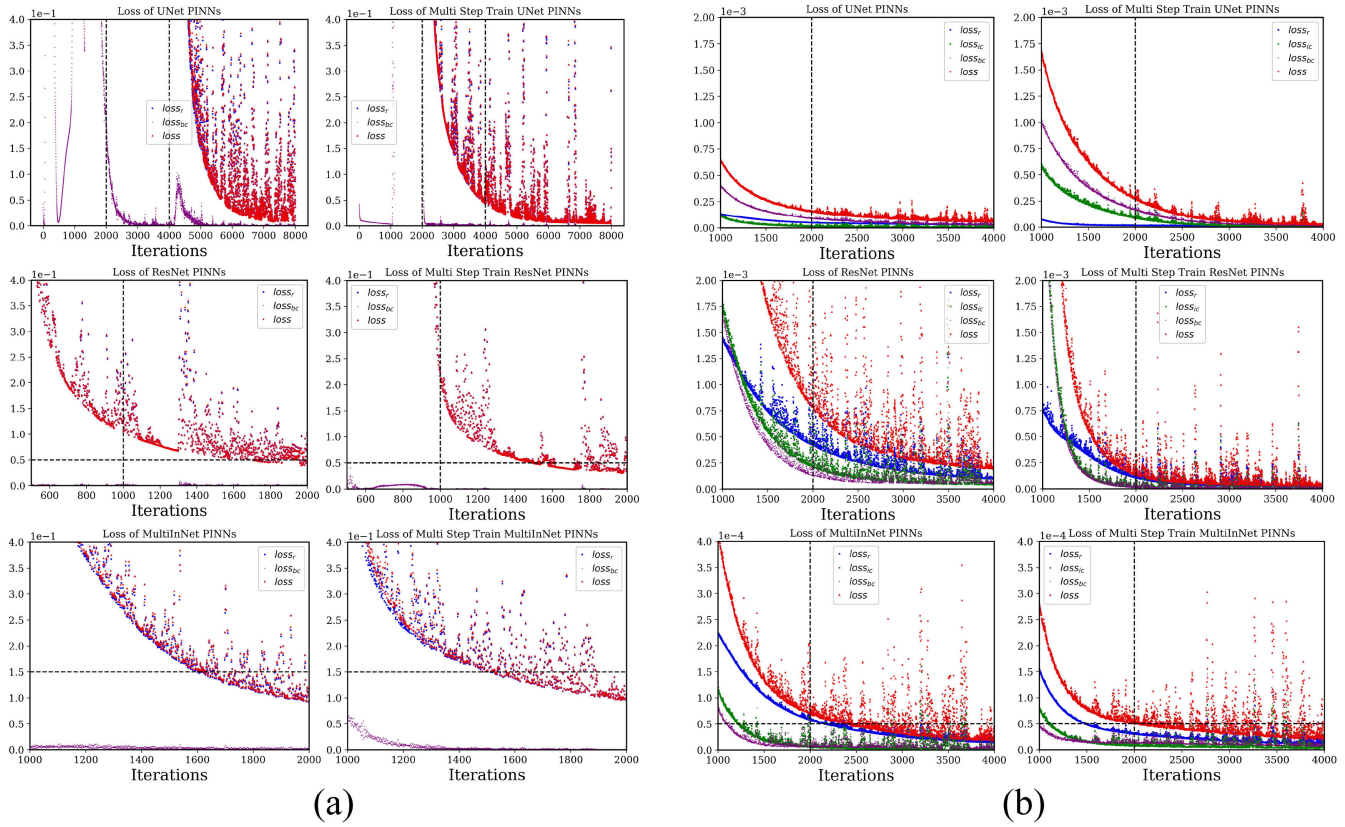


FIGURE 8. One step train (left) vs. multi step train (right) for loss convergence of the Poisson/Advection equation ((a) is poisson equation, and (b) is advection equation).

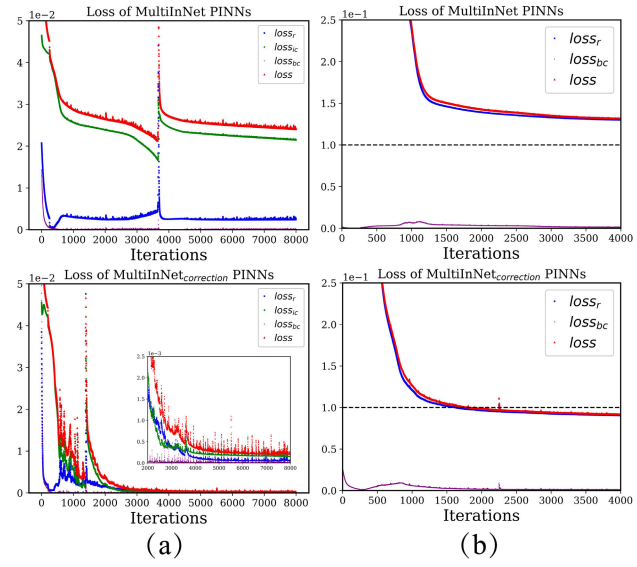


FIGURE 9. MultiInNet PINNs (up) vs. its corrected version (down) for loss convergence of the Convention-diffusion/Helmhoz equation.

dynamic interplay between the initial conditions and PDE residuals during early convergence stages. However, our tailored training approach and selected backbone network are designed to address these complexities, as evidenced in left side of Figure 5. Our training strategy mitigates initial hurdles, such as balancing the significance of the initial condition loss and PDE loss. Conversely, UNet and ResNet exhibit earlier convergence stagnation. Our corrected MultiInNet surpasses these issues, achieving convergence around 2500 iterations.

D. HELMHOLTZ EQUATION

In our final example, we consider the Helmholtz equation [42], a fundamental elliptic partial differential equation that characterizes the behavior of electromagnetic waves. The equation is presented in the following form within this paper:

$$\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} + \kappa^2 u - q(x,y) = 0, \qquad x, y \in [-L, L] \quad (16)$$

and we set $L = 1.0$, meanwhile $q(x, y)$ as follows:

$$\begin{aligned} q(x,y) = &-(a_1\pi)^2 sin(a_1\pi x) sin(a_2\pi y) \\ &-(a_2\pi)^2 sin(a_1\pi x) sin(a_2\pi y) \\ &+\kappa^2 sin(a_1\pi x) sin(a_2\pi y)\omega \end{aligned} \quad (17)$$

and for simplicity we take $\kappa = \omega = 1.0$, boundary conditions as follows:

$$u(-L, y) = u(L, y) = u(x, -L) = u(x, L) = 0 \quad (18)$$

Similarly, for vanilla PINNs we set the weight ratio $w_r : w_{ic} : w_{bc} = 1 : 5 : 5$, and all other ratios are equal to 1. As before, the boundary conditions are complicated, which means that the final level of convergence is often limited by the convergence of the boundary conditions, as can be seen

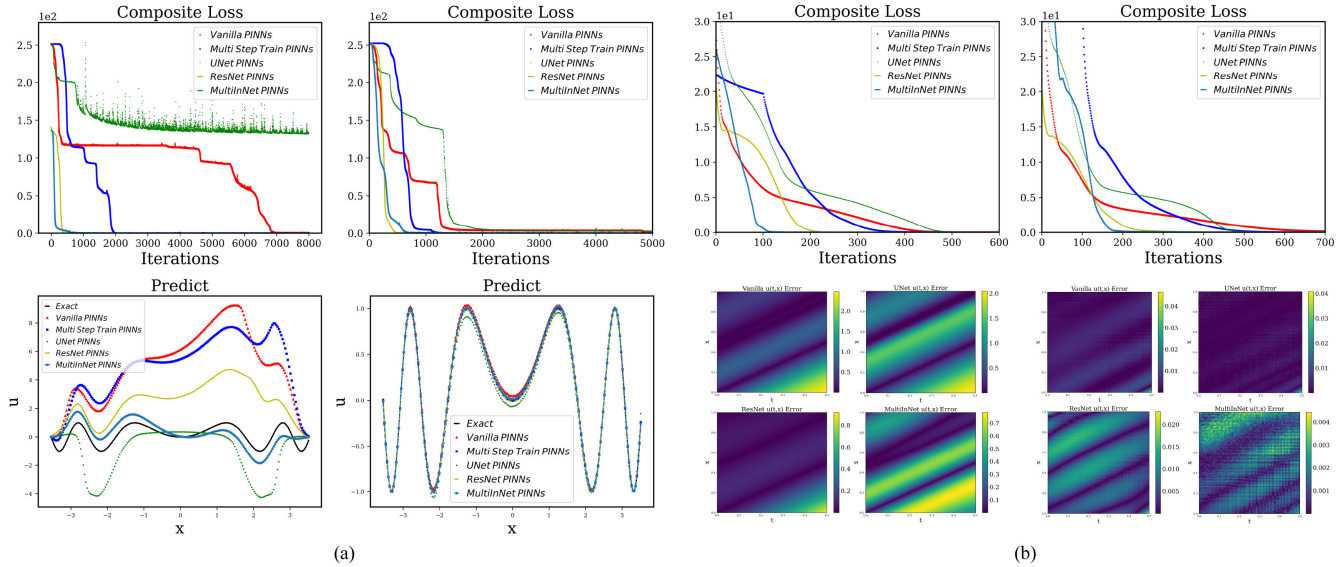


FIGURE 10. The loss convergence and prediction(absolute error) of the Poisson/Advection equation: (a) is Poisson equation, and (b) is Advection equation. The left side of (a)/(b) is the first scenario(20 sample points), the right side is the second(200 sample points).

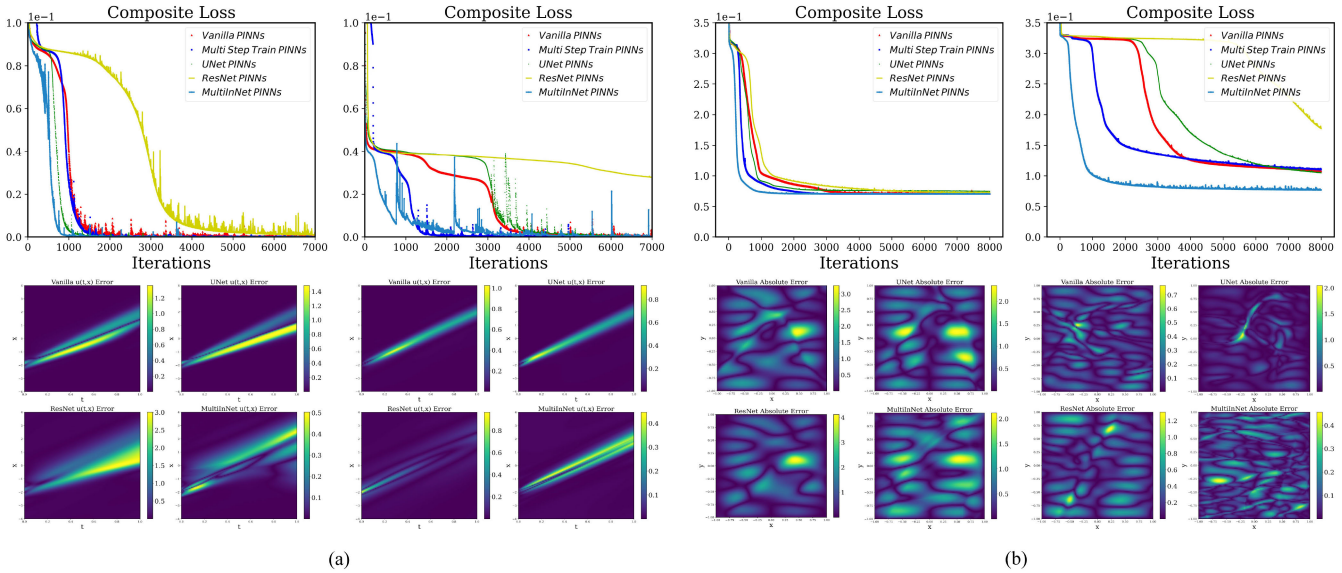


FIGURE 11. The loss convergence and absolute error of the Convention-diffusion/Helmholtz equation: (a) is Convention-diffusion, and (b) is Helmholtz equation. The left side of (a)/(b) is the first scenario(20 sample points), the right side is the second(200 sample points).

in right side of Figure 5, however our corrected MultiInNet converge to a better level.

Furthermore, to assess the model's robustness and the experiments' generality, we incorporated a comparative analysis of prediction and convergence across distinct network structures in the scenario where $\kappa = 10$. In this example, no multi-step training strategy is considered for training, except for the vanilla PINNs. The outcomes of these comparisons are represented in Figure 6 and Figure 7.

V. ABLATION EXPERIMENTS

To validate our training method's effectiveness, we conduct ablation experiments based on the previous section. The first two experiments maintain the same hyperparameters as in the previous section, except for the training method. For UNet, multi-step training yields noticeable improvements. However, for ResNet and MultiInNet (with corrections), due to the presence of residual connections, we opt to fix the first and last layers while employing multi-step training. Achieving more precise subnetwork parameters might have been possible but was not pursued. The next two experiments investigate the correction's impact on MultiInNet, and the concluding experiment was conducted to investigate the influence of diverse sample sizes on the convergence tendencies exhibited by various network structures.

A. COMPARISON OF TRAINING METHODS

The comparison of loss convergence using one step training method and using multi step training method for solving poisson/advection equation is shown in Figure 8.

B. INFLUENCE OF CORRECTION TO THE MultiInNet

The loss convergence comparison between MultiInNet PINNs and its corrected version for solving the convection-diffusion/Helmholtz equation is depicted in Figure 9 (without employing any training strategy). Obviously, the modified version shows improved convergence with a lower level of loss compared to the original MultiInNet.

C. INFLUENCE OF SAMPLE SCALE

Furthermore, we scrutinize the effectiveness of the network and training methods across diverse configurations of training sample points. To elucidate the convergence of training loss, our experimental design encompasses two distinct scenarios. The first scenario involves an examination with a lesser number of sample points(20 points), while the second scenario explores a more substantial set of training sample points(200 points). The loss convergence and prediction(absolute error) of poisson/advection equation is shown as Figure 10, and convention-diffusion/helmholtz equation is shown as Figure 11.

VI. CONCLUSION AND FUTURE WORK

In this study, we have observed significant boundary information primarily present in deeper networks. Consequently, we aimed to implement novel training methodologies and backbone networks within PINNs, and this adaptation allows the PINNs backbone to access initial and boundary information from the outset of training, theoretically expediting convergence. Our experimental findings affirm the effectiveness of our approach, demonstrating that our devised training method and network model (MultiInNet PINNs) notably accelerate the convergence speed of PINNs. Nevertheless, in scenarios where the problem entails exceedingly intricate initial and boundary conditions, Resnet and MultiInNet will struggle to converge, therefore, we've adapted MultiInNet to circumvent the propagation of erroneous gradient information, and this modification distinctly mitigates convergence stagnation. In summary, this paper aims to provide a robust training strategy and recommended network model for PINNs to enhance their convergence efficiency. Effectively utilizing initial and boundary conditions to facilitate PINNs training stands as a pivotal aspect and plays a fundamental role in the generalization capabilities of PINNs. There are many models, such as FNO (Fourier Neural Operator) [43] and DeepONet (Deep Operator Networks) [44], exhibit inherent generalization advantages when dealing with solving Partial Differential Equations (PDEs). Related work has also been done combining them with PINNs to enhance generalization and overall performance [45], [46]. In our prospective investigations, our primary focus will be on enhancing the generalization capabilities of Physics-Informed Neural Networks (PINNs). Strengthening their ability to generalize is crucial for increasing their applicability in industrial settings and unlocking their broader potential.

ACKNOWLEDGMENT

This study was generously funded by multiple organizations. The authors would like to thank the National Supercomputing Center in Chengdu for providing the necessary equipment for this study.

REFERENCES

[1] S. K. Godunov and I. Bohachevsky, "Finite difference method for numerical computation of discontinuous solutions of the equations of fluid dynamics," *Sbornik Math.*, vol. 47, no. 3, pp. 271–306, Jul. 1959.
[2] R. Eymard, T. Gallouët, and R. Herbin, "Finite volume methods," in *Handbook of Numerical Analysis*, vol. 7. New York, NA, USA: Elsevier, 2000, pp. 713–1018.
[3] J. J. Monaghan, "An introduction to SPH," *Comput. Phys. Commun.*, vol. 48, no. 1, pp. 89–96, Jan. 1988.
[4] L. Li and Y. Xie, "Rogue wave solutions of the generalized (3+1)-dimensional Kadomtsev–Petviashvili equation," *Chaos Solitons Fractals*, vol. 147, Jun. 2021, Art. no. 110935.
[5] Y. Xie and L. Li, "Multiple-order breathers for a generalized (3+1)-dimensional Kadomtsev–Petviashvili Benjamin–Bona–Mahony equation near the offshore structure," *Math. Comput. Simul.*, vol. 193, p. 1931, Mar. 2022.
[6] L. Li, Y. Nie, M. Zhu, and Y. Xie, "Variable separation solution and multi-valued soliton of an extended (3+1)-dimensional B-type Kadomtsev–Petviashvili equation," *Nonlinear Dyn.*, vol. 111, no. 5, pp. 4723–4736, Mar. 2023.
[7] G. E. Karniadakis et al., "Physics-informed machine learning," *Nat. Rev. Phys.*, vol. 3, no. 6, pp. 422–440, May 2021.
[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.
[9] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Proc. Adv. Neural. Inf. Process. Syst.*, vol. 33, Dec. 2020, pp. 6840–6851.
[10] M. W. M. G. Dissanayake and N. Phan-Thien, "Neural-network-based approximations for solving partial differential equations," *Commun. Numer. Methods Eng.*, vol. 10, no. 3, pp. 195–201, Mar. 1994.
[11] I. E. Lagaris, A. Likas, and D. I. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," *IEEE Trans. Neural Netw.*, vol. 9, no. 5, pp. 987–1000, Sep. 1998.
[12] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic differentiation in machine learning: A survey," *J. Mach. Learn. Res.*, vol. 18, p. 143, Nov. 2018.
[13] T. Fan, K. Xu, J. Pathak, and E. Darve, "Solving inverse problems in steady-state Navier–Stokes equations using deep neural networks," Aug. 2020, *arXiv:2008.13074*.
[14] W. Wang, G. A. McMechan, and J. Ma, "Elastic isotropic and anisotropic full-waveform inversions using automatic differentiation for gradient calculations in a framework of recurrent neural networks," *Geophysics*, vol. 86, no. 6, pp. R795–R810, Nov. 2021.
[15] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *J. Comput. Phys.*, vol. 378, pp. 686–707, Feb. 2019.
[16] S. Cai, Z. Mao, Z. Wang, M. Yin, and G. E. Karniadakis, "Physics-informed neural networks (PINNs) for fluid mechanics: A review," *Acta Mechanica Sinica*, vol. 37, no. 12, pp. 1727–1738, Dec. 2021.
[17] R. Mojgani, M. Balajewicz, and P. Hassanzadeh, "Lagrangian PINNs: A causality-conforming solution to failure modes of physics-informed neural networks," May 2022, *arXiv:2205.02902*.
[18] M. Penwarden, S. Zhe, A. Narayan, and R. M. Kirby, "Multifidelity modeling for physics-informed neural networks (PINNs)," *J. Comput. Phys.*, vol. 451, Feb. 2022, Art. no. 110844.
[19] P. Kopyt, B. Salski, P. Zagrajek, D. Janczak, M. Sloma, M. Jakubowska, M. Olszewska-Placha, and W. Gwarek, "Electric properties of graphene-based conductive layers from DC up to terahertz range," *IEEE Trans. THz Sci. Technol.*, vol. 6, no. 3, pp. 480–490, Apr. 2016, doi: 10.1109/TTHZ.2016.2544142.
[20] L. Yang, X. Meng, and G. E. Karniadakis, "B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data," *J. Comput. Phys.*, vol. 425, Jan. 2021, Art. no. 109913.

[21] L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis, "DeepXDE: A deep learning library for solving differential equations," *SIAM Rev.*, vol. 63, no. 1, pp. 208–228, Jan. 2021.

[22] S. Wang, Y. Teng, and P. Perdikaris, "Understanding and mitigating gradient flow pathologies in physics-informed neural networks," *SIAM J. Sci. Comput.*, vol. 43, no. 5, pp. A3055–A3081, Jan. 2021.

[23] J. Abbasi and P. Ø. Andersen, "Physical activation functions (PAFs): An approach for more efficient induction of physics into physics-informed neural networks (PINNs)," May 2022, *arXiv:2205.14630*.

[24] R. Gnanasambandam, B. Shen, J. Chung, X. Yue, and Z. Kong, "Self-scalable tanh (Stan): Faster convergence and better generalization in physics-informed neural networks," Apr. 2022, *arXiv:2204.12589*.

[25] A. D. Jagtap, K. Kawaguchi, and G. E. Karniadakis, "Adaptive activation functions accelerate convergence in deep and physics-informed neural networks," *J. Comput. Phys.*, vol. 404, Mar. 2020, Art. no. 109136.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[27] S. Wang, X. Yu, and P. Perdikaris, "When and why PINNs fail to train: A neural tangent kernel perspective," *J. Comput. Phys.*, vol. 449, Jan. 2022, Art. no. 110768.

[28] Z. Xiang, W. Peng, X. Liu, and W. Yao, "Self-adaptive loss balanced physics-informed neural networks," *Neurocomputing*, vol. 496, pp. 11–34, Jul. 2022.

[29] K. Zubov, Z. McCarthy, Y. Ma, F. Calisto, V. Pagliarino, S. Azeglio, L. Bottero, E. Luján, V. Sulzer, A. Bharambe, N. Vinchhi, K. Balakrishnan, D. Upadhyay, and C. Rackauckas, "NeuralPDE: Automating physics-informed neural networks (PINNs) with error approximations," 2021, *arXiv:2107.09443*.

[30] L. Liu, S. Liu, H. Xie, F. Xiong, T. Yu, M. Xiao, L. Liu, and H. Yong, "Discontinuity computing using physics-informed neural network," Jun. 2022, *arXiv:2206.03864*.

[31] M. A. Nabian, R. J. Gladstone, and H. Meidani, "Efficient training of physics-informed neural networks via importance sampling," *Comput.-Aided Civil Infrastruct. Eng.*, vol. 36, no. 8, pp. 962–977, Apr. 2021.

[32] X. Liu, X. Zhang, W. Peng, W. Zhou, and W. Yao, "A novel meta-learning initialization method for physics-informed neural networks," *Neural Comput. Appl.*, vol. 34, no. 17, pp. 14511–14534, May 2022.

[33] R. Sharma and V. Shankar, "Accelerated training of physics-informed neural networks (PINNs) using meshless discretizations," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, Dec. 2022, pp. 1034–1046.

[34] X. Ying, "An overview of overfitting and its solutions," *J. Phys., Conf. Ser.*, vol. 1168, Mar. 2019, Art. no. 022022, doi: 10.1088/1742-6596/1168/2/022022.

[35] L. Sun, H. Gao, S. Pan, and J.-X. Wang, "Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data," *Comput. Methods Appl. Mech. Eng.*, vol. 361, Apr. 2020, Art. no. 112732.

[36] J. W. Siegel, Q. Hong, X. Jin, W. Hao, and J. Xu, "Greedy training algorithms for neural networks and applications to PDEs," *J. Comput. Phys.*, vol. 484, Jul. 2023, Art. no. 112084.

[37] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[38] A. G. Özbay, A. Hamzehloo, S. Laizet, P. Tzirakis, G. Rizos, and B. Schuller, "Poisson CNN: Convolutional neural networks for the solution of the Poisson equation on a Cartesian mesh," *Data-Centric Eng.*, vol. 2, p. e6, Jun. 2021, doi: 10.1017/dce.2021.7.

[39] M. F. Lazin, C. R. Shelton, S. N. Sandhofer, and B. M. Wong, "High-dimensional multi-fidelity Bayesian optimization for quantum control," *Mach. Learn., Sci. Technol.*, vol. 4, no. 4, Oct. 2023, Art. no. 045014.

[40] S. Stock, J. Stiasny, D. Babazadeh, C. Becker, and S. Chatzivasileiadis, "Bayesian physics-informed neural networks for robust system identification of power systems," in *Proc. IEEE Belgrade PowerTech*, Belgrade, Serbia, Jun. 2023, pp. 1–6, doi: 10.1109/PowerTech55446.2023.10202692.

[41] B. Shan, Y. Li, and S. Huang, "VI-PINNs: Variance-involved physics-informed neural networks for fast and accurate prediction of partial differential equations," Nov. 2022, *arXiv:2211.16753*.

[42] Y. A. Erlangga, C. Vuik, and C. W. Oosterlee, "On a class of preconditioners for solving the Helmholtz equation," *Appl. Numer. Math.*, vol. 50, nos. 3–4, pp. 409–425, Sep. 2004.

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

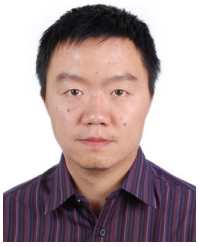
[44] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis, "Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators," *Nature Mach. Intell.*, vol. 3, no. 3, pp. 218–229, Mar. 2021.

[45] Z. Li, H. Zheng, N. Kovachki, D. Jin, H. Chen, B. Liu, K. Azizzadenesheli, and A. Anandkumar, "Physics-informed neural operator for learning partial differential equations," Jul. 2021, *arXiv:2111.03794*.

[46] S. Wang, H. Wang, and P. Perdikaris, "Learning the solution operator of parametric partial differential equations with physics-informed DeepONets," *Sci. Adv.*, vol. 7, no. 40, p. eabi8605, Sep. 2021.

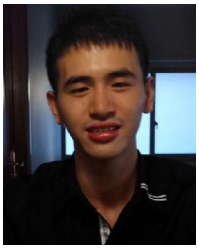
LIANGLIANG YAN received the B.Eng. degree in internetworking from the Tianjin University of Technology and Education. He is currently pursuing the M.Eng. degree with the School of Computer and Network Security, Chengdu University of Technology, Chengdu, China.

His research interests include machine learning, deep learning, and AI for science.



YOU ZHOU is currently an Associate Professor with the Planetary Science Research Center, Chengdu University of Technology. He is also an Associate Researcher with the CAS Center for Excellence in Comparative Planetology. His primary research interests include giant impact simulations using the smoothed particle hydrodynamics (SPH) method and in enhancing fluid dynamics through machine learning.

HUAN LIU received the B.S. degree in computer science from the Nanjing Institute of Technology, Nanjing, Jiangsu, China, in 2004, the M.S. degree from Jiangxi Normal University, Nanchang, Jiangxi, China, in 2008, and the Ph.D. degree in pattern recognition and intelligent system from Donghua University, Shanghai, China, in 2014. She is currently an Associate Professor with the College of Electric and Information Engineering, Jinggangshan University, Jiangxi. Her research interests include image processing, pattern recognition, and machine vision.



LINGQI LIU received the B.Eng. degree in digital media technology from the Guilin University of Electronic Technology. He is currently pursuing the M.Eng. degree with the School of Computer and Network Security, Chengdu University of Technology, Chengdu, China. His research interests include deep learning and physics informed machine learning.