

Commun. Theor. Phys. **76** (2024) 025005 (11pp) https://doi.org/10.1088/1572-9494/ad1a0e

Physical informed memory networks for solving PDEs: implementation and applications

Jiuyun Sun, Huanhe Dong and Yong Fang*

College of Mathematics and Systems Science, Shandong University of Science and Technology, Qingdao 266590, China

E-mail: fangyong@sdust.edu.cn

Received 30 June 2023, revised 15 November 2023
Accepted for publication 3 January 2024
Published 7 February 2024

Abstract

With the advent of physics informed neural networks (PINNs), deep learning has gained interest for solving nonlinear partial differential equations (PDEs) in recent years. In this paper, physics informed memory networks (PIMNs) are proposed as a new approach to solving PDEs by using physical laws and dynamic behavior of PDEs. Unlike the fully connected structure of the PINNs, the PIMNs construct the long-term dependence of the dynamics behavior with the help of the long short-term memory network. Meanwhile, the PDEs residuals are approximated using difference schemes in the form of convolution filter, which avoids information loss at the neighborhood of the sampling points. Finally, the performance of the PIMNs is assessed by solving the KdV equation and the nonlinear Schrödinger equation, and the effects of difference schemes, boundary conditions, network structure and mesh size on the solutions are discussed. Experiments show that the PIMNs are insensitive to boundary conditions and have excellent solution accuracy even with only the initial conditions.

Keywords: nonlinear partial differential equations, physics informed memory networks, physics informed neural networks, numerical solution

(Some figures may appear in colour only in the online journal)

1. Introduction

Partial differential equations (PDEs) are widely used to describe nonlinear phenomena in nature [1–3]. And, solving PDEs is helpful in understanding the physical laws behind these nonlinear phenomena [4–7]. However, analytical solutions of PDEs are often very difficult to obtain [8]. Accordingly, numerical methods have been proposed and promoted the study of PDEs [9, 10]. Due to the large computational demands of these methods, the accuracy and efficiency of solving PDEs are difficult to acquire simultaneously.

In recent years, deep learning methods have been extended from natural language recognition and machine translation to scientific computing, and have provided new ideas for solving PDEs [11–13]. According to the universal approximation theorem, a multilayered feed forward network containing a sufficient number of hidden neurons can approximate any continuous function with arbitrary accuracy [14, 15], which provides the theoretical support for deep learning to solve PDEs. At this stage, there are two types of deep learning methods for solving PDEs [16]. The first type keep the same learning approach as the original deep learning methods. The basic theory is constructing neural operators by learning mappings from function parameter dependence to solutions, such as Deeponet, Fourier neural operator, so on [17, 18]. This type needs to be trained only once to handle different initial value problems, but requires a large amount of data with high fidelity. The second type combines deep learning with physical laws. In second type, the physical laws and a small amount the initial and boundary data of PDEs are used to constrain the network training instead of lots of labeled data. The representative is physics informed neural

* Author to whom any correspondence should be addressed.

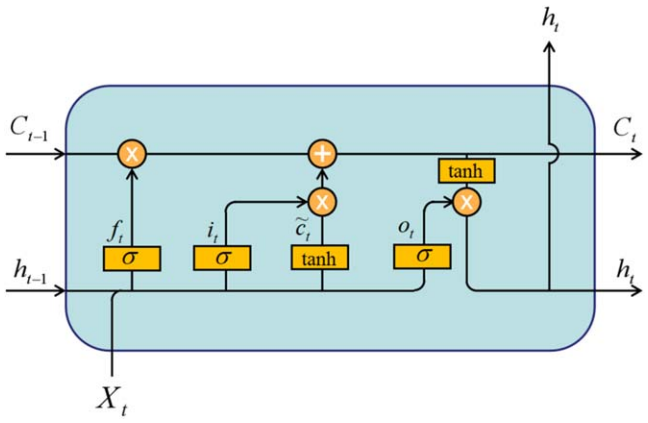

Figure 1. The general structure of LSTM.

networks (PINNs) that can solve both nonlinear PDEs and corresponding inverse problems [19]. Based on the original PINNs, many improved versions were proposed [20–26]. Ameya *et al* set a scalable hyper-parameter in the activation function and proposed an adaptive activation function with better learning capabilities and convergence speed [22]. Lin and Chen devise a two-stage PINNs method based on conserved quantities, which better exploits the properties of PDEs [23]. In addition to the physical laws, the variational residual of PDEs is also considered as a loss term, such as deep Ritz method, deep Galerkin method and so on [27–29]. As the intensive study of the PINNs and its variants, these algorithms are applied to many fields, such as biomedical problems [30], continuum micromechanics [31], stiff chemical kinetics [32]. Inspired by the PINNs, some deep learning solver for non-fully connected structures are proposed. Zhu *et al* constructed a physics-constrained convolutional encoding-decoding structure for stochastic PDEs [33]. Based on the work of Zhu *et al*, physics-informed convolutional-recurrent networks combined with long short-term memory network (LSTM) were proposed, while the initial and boundary conditions were hard-encoded into the network [34]. Mohan *et al* used an extended convolutional LSTM to model turbulence [35]. Based on this, Stevens *et al* exploited the temporal structure [36]. In general, existing deep learning solvers with LSTM structures can approximate the dynamic behavior of the solution without any labeled data, but the implementations rely on feature extraction of convolutional structures and are complex.

The aim of this paper is to build a flexible deep learning solver based on physical laws and temporal structures. Therefore, physics informed memory networks (PIMNs) based on LSTM framework is proposed as a new method for solving PDEs. In the PIMNs, the differential operator is approximated using difference schemes rather than automatic differentiation (AD). AD is flexible and ingenious, but loses information about the neighbors of the sampling points [37]. The differential schemes are implemented as convolution filter, and the convolution filter is only used to calculate the physical residuals and does not change with network training. This is different from existing solvers with convolutional structures. Numerical experiments on the KdV equation and the nonlinear Schrödinger equation show that the PIMNs can achieve excellent accuracy and are insensitive to the boundary conditions.

The rest of the paper is organized as follows. In section 2, the general principle and network architectures of the PIMNs are elaborated. In section 3, two sets of numerical experiments are given and the effects of various influencing factors on the learned solution are discussed. Conclusion is given in last section.

2. Physics informed memory networks

2.1. Problem setup

In general, the form of PDEs that can be solved by physical informed deep learning is as follows:

$$\begin{aligned} &u_t + \mathcal{N}[u] = 0,\ x \in [x_0, x_1],\ t \in [0, T], \\ &u(x, 0) = u_0(x), \\ &u(x_0, t) = u_1(t), \\ &u(x_1, t) = u_2(t), \end{aligned} \tag{1}$$

where $u(x, t)$ is the solution of PDEs and $\mathcal{N}[u]$ is a nonlinear differential operator. $u_0(x)$ and $u_1(t)$, $u_2(t)$ are initial and boundary functions, respectively. And, the basic theory of physical informed deep learning is approximating the solution $u(x, t)$ through the constraints of the physical laws [15]. Therefore, the PDEs residual $f(x, t)$ is defined as

$$f(x, t) := u_t + \mathcal{N}[u]. \tag{2}$$

The keys of the PIMNs are the calculation of physical residuals $f(x, t)$ using the difference schemes and the establishment of the corresponding long-term dependence.

2.2. Physics informed memory networks

In this part, the framework and principles of the PIMNs are given. As shown in figure 2, the basic unit of the PIMNs is LSTM. LSTM inherits the capability of the recurrent neural network for long sequence data and can avoid the problem of vanishing gradient [38].

The structure of the LSTM unit is shown in figure 1. X_t, h_t, c_t, $\tilde{c}_t$, f_t, i_t, o_t are the input, the hidden state, the cell state, the internal cell state, the forget gate, the input gate, the output gate, respectively. f_t and i_t control the information forgotten and added to c_t. $\tilde{c}_t$ is the information added to c_t. The output is determined jointly by o_t and c_t. The mathematical expression of LSTM is shown as follows:

$$\begin{aligned} f_t &= \sigma(W_{Xf}X_t + W_{hf}h_{t-1} + b_f), \\ i_t &= \sigma(W_{Xi}X_t + W_{hi}h_{t-1} + b_i), \\ \tilde{c}_t &= \tanh(W_{Xc}X_t + W_{hc}h_{t-1} + b_c), \\ c_t &= f_t \circ c_{t-1} + i_t \circ \tilde{c}_t, \\ o_t &= \sigma(W_{Xo}X_t + W_{ho}h_{t-1} + W_{co} \circ c_t + b_o), \\ h_t &= o_t \circ \tanh(c_t). \end{aligned} \tag{3}$$

Here, W, b are the network parameters, and $\circ$ represents the Hadamard product.

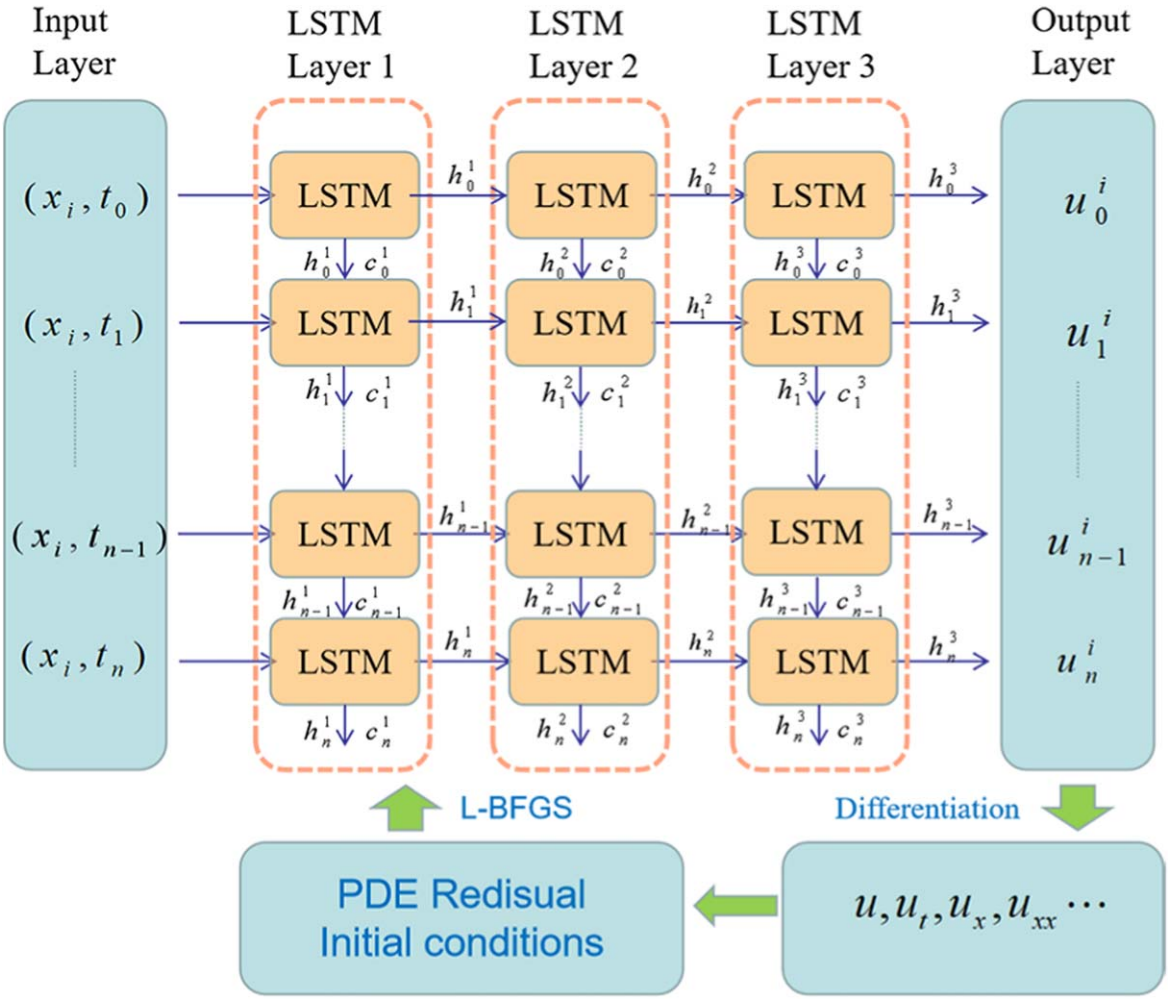

Figure 2. The general structure of the PIMNs.

In the PIMNs, subscripts and superscripts of h_t^i and c_t^i refer to the number of time series and layers, respectively. LSTM unit imports the output h_t^i of the current moment t to the next moment $t+1$, which links all moments of the same spatial point. In addition, h_t^i is also used as input to moment t for the next LSTM layer, which strengthens the connection between different moments. It should be noted that the hidden nodes of the last LSTM are fixed. In fact, we use the last LSTM to control the dimensionality of the output.

As shown in figure 2, the inputs to the PIMNs are the coordinates of the grid points in the region $[x_0, x_1] \times [0, T]$. The region $[x_0, x_1] \times [0, T]$ is meshed into $(m+1) \times (n+1)$ grid points. Each set of inputs to the PIMNs are coordinate values (x_i, t_0), (x_i, t_1), $\cdots$,(x_i, t_n) at the same location. The outputs are the corresponding u_0^i, u_1^i,$\cdots$,u_n^i (corresponding to each row of the left panel of figure 3). Based on the output u_j^i of the PIMNs, their loss functions can be constructed. The loss function of includes three components:

$$\mathrm{MSE} = \mathrm{MSE}_f + \mathrm{MSE}_I + \mathrm{MSE}_B, \tag{4}$$

where

$$\mathrm{MSE}_f = \frac{1}{(m+1)(n+1)} \sum_{i=1}^{m+1} \sum_{j=1}^{n+1} |f_j^i|^2, \tag{5}$$

$$\mathrm{MSE}_I = \frac{1}{m+1} \sum_{i=1}^{m+1} |u_0^i - u(x_i, t_0)|^2, \tag{6}$$

$$\mathrm{MSE}_B = \frac{1}{n+1} \sum_{j=1}^{n+1} |u_j^0 - u(x_0, t_j)|^2 + \frac{1}{n+1} \sum_{j=1}^{n+1} |u_j^m - u(x_m, t_j)|^2. \tag{7}$$

Here, u_j^i and $u(x_i, t_j)$ are the numerical solution and exact solution for the grid points (x_i, t_j), respectively. f_j^i is the physical residual for the grid points (x_i, t_j). u_j^i is the output of the PIMNs and f_j^i is obtained by taking the terms u_t, u_x, etc, constructed based on u_j^i into equation (2). MSE_f, MSE_I and MSE_B correspond to the physical residuals, initial and boundary losses, respectively. MSE_I and MSE_B are obtained by the learned solution and the corresponding known data. The terms u_t, u_x, etc for constructing MSE_f are obtained from the difference schemes and are computed as (in the case of the second-order center difference):

$$u_{j,x}^i = \frac{u_j^{i+1} - u_j^{i-1}}{2\delta x}, \tag{8}$$

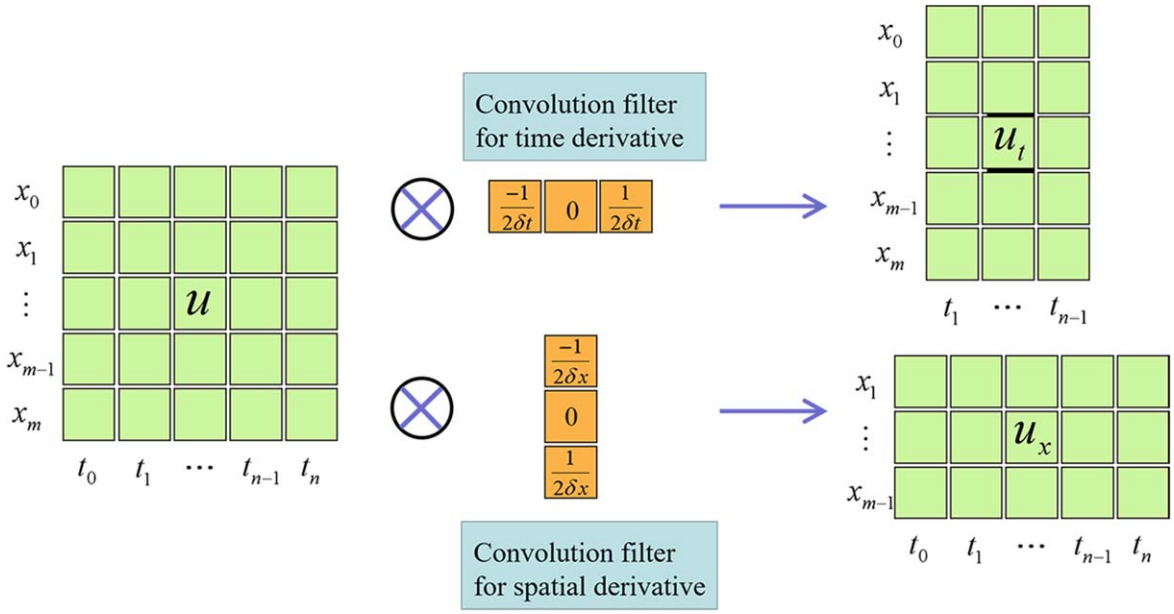

Figure 3. The convolution process of difference schemes.

$$u_{j,t}^{i} = \frac{u_{j+1}^{i} - u_{j-1}^{i}}{2\delta t}. \tag{9}$$

Here, δx and δt represent spatial and temporal intervals, respectively. To accelerate the training, the difference schemes are implemented by convolution operations. Taking second-order central difference schemes as an example, the convolution filters are:

$$K_x = [-1,0,1]^T \times \frac{1}{2\delta x}, \tag{10}$$

$$K_t = [-1,\ 0,\ 1] \times \frac{1}{2\delta t}. \tag{11}$$

Figure 3 illustrates the computation of u_x and u_t. Higher order derivatives can be obtained by performing a difference operation on u_x and u_t, such as

$$u_{j,xx}^{i} = \frac{u_{j,x}^{i+1} - u_{j,x}^{i-1}}{2\delta x}. \tag{12}$$

As shown in figure 3, the padding is not done to avoid unnecessary errors. This leads to the fact that u_t at $t = t_0$, $t = t_n$ and u_x at $x = x_0$, $x = x_m$ cannot be computed (in terms of the second-order central difference). In other words, $f_j{}^i$ cannot be computed at $t = t_0$ (i.e. the initial condition) due to the lack of u_{-1}^{i}. A similar problem arises with the boundary conditions. Therefore, $f_j{}^i$ at $t = t_0$, $x = x_0$ and $x = x_m$ will not be used to compute the loss. When higher order derivative terms are included in $f_j{}^i$, the region in which losses are not computed is correspondingly enlarged. In fact, since the spatial and temporal intervals we obtain are usually small and the boundaries of the domain can be bounded by the initial and boundary conditions, the numerical solutions of the PDEs can be approximated even without computing $f_j{}^i$ near the boundaries of the domain. And, the boundary conditions are not necessary, which will be tested in the numerical experiments. Correspondingly, the total loss excludes the boundary loss when the boundary condition is missing. In addition, the spatial derivatives at t_0 are constructed by known initial conditions. Therefore, except for being the target for the initial loss, the initial conditions are also used indirectly for the PDEs residuals.

3. Numerical experiments

In this section, the PIMNs are applied to solve the KdV equation and the nonlinear Schrödinger equation. Specifically, the effects of different difference schemes on the solving ability of the PIMNs are discussed, and the network structure and mesh size that minimize the error are investigated. Moreover, the performance of the PIMNs and the PINNs are compared.

In the numerical experiments, the implementation of the PIMNs is based on Python 3.7 and Tensorflow 1.15. The loss function is chosen as the mean squared loss function, and L-BFGS is used to optimize the loss function. All numerical examples reported here are run on a Lenovo Y7000P 2020H computer with 2.60 GHz 6-core Intel(R) Core(TM) i7-10750H CPU and 16 GB memory. In addition, the relative L_2 error is used to measure the difference between the predicted and true values and is calculated as follows:

$$\text{Error}_{L_2} = \frac{\sum_{i=1}^{m+1}\sum_{j=1}^{n+1}|u_j^i - u(x_i,\ t_j)|^2}{\sum_{i=1}^{m+1}\sum_{j=1}^{n+1}|u(x_i,\ t_j)|^2}. \tag{13}$$

Here, u_j^i and $u(x_i,\ t_j)$ are the numerical solution and exact solution for the grid points $(x_i,\ t_j)$, respectively. Thus, the relative L_2 error is obtained by calculating the numerical and exact solutions for all grid points.

Table 1. The KdV equation: relative L_2 errors for different difference schemes.

Difference schemes	Forward difference	Backward difference	Central difference
1	$4.355\,988 \times 10^{-3}$	$4.638\,906 \times 10^{-3}$	$2.034\,928 \times 10^{-3}$
2	$4.058\,963 \times 10^{-3}$	$4.640\,369 \times 10^{-3}$	$1.317\,186 \times 10^{-3}$
3	$4.884\,505 \times 10^{-3}$	$4.459\,566 \times 10^{-3}$	$1.124\,485 \times 10^{-3}$
4	$4.580\,123 \times 10^{-3}$	$4.348\,073 \times 10^{-3}$	$9.399\,773 \times 10^{-4}$

3.1. Case 1: The KdV equation

The KdV equation is a classical governing model for the propagation of shallow water waves and has important applications in many areas of physics, such as fluid dynamics, plasma [39–41]. In general, the KdV equation is given by

$$\begin{aligned} q_t + 6qq_x + q_{xxx} = 0,\ x \in [x_0, x_1],\ t \in [t_0, t_1], \\ q(x, t_0) = q_0(x), \\ q(x_0, t) = q_1(x), \\ q(x_1, t) = q_2(x), \end{aligned} \tag{14}$$

where $q_0(x)$ is an initial function, and $q_1(x)$ and $q_2(x)$ are boundary functions. x_0 and x_1 are arbitrary real constants. The PDEs residual $f(x, t)$ corresponding to the KdV equation is:

$$f(x, t) \coloneqq q_t + 6qq_x + q_{xxx}. \tag{15}$$

And, the existence theory of equation (14) can be referred to in the [42]. Here, the traveling wave solution is simulated:

$$q(x, t) = a + 2k^2 \operatorname{sech}^2\{k[x - (4k^2 + 6u_0)t + x_0]\}, \tag{16}$$

where $k^2 = \frac{1}{2}(c - a)$, a, c are real constants and $c > a$. Taking $a = 0.1$, $c = 0.8$, $x_0 = 0$, equation (16) is reduced to

$$q(x, t) = 0.1 + 0.7 \operatorname{sech}^2\left[\frac{\sqrt{7}}{2\sqrt{5}}(x - 2t)\right]. \tag{17}$$

Taking $[x_0, x_1]$ and $[t_0, t_1]$ as $[-10, 10]$ and $[0, 1]$, the corresponding initial and boundary functions are obtained.

3.1.1. Comparison of different difference schemes for solving the KdV equation. In section 2.2, we constructed the PDEs residuals by the second-order central difference. However, it is important to discuss which difference scheme is suitable for constructing time derivatives for establishing the long-term dependence of PDEs. Here, forward difference, backward difference and central difference are used to computer the temporal derivatives, respectively. The space $[-10, 10]$ is divided into 1000 points and the time $[0, 1]$ is divided into 100 points. Two LSTM layers are used with the number of nodes 30 and 1, respectively. Since the initialization of the network parameters is based on random number seeds, 4 sets of experiments based on different random number seeds (i.e. 1, 2, 3, 4 in table 1) were set up in order to avoid the influence of chance. The relative errors for four sets of numerical experiments are given in table 1.

From the data in table 1, the PIMNs can solve the KdV equation with very high accuracy for all three difference methods. The bolded data are the lowest relative L_2 errors produced by the same random number. It can be clearly seen that the relative L_2 errors produced by the central difference is significantly lower than that of the forward and backward difference when using the same network architecture and training data. This indicates that the temporal structure constructed by the central difference is most suitable for solving the KdV equation.

3.1.2. The effect of boundary conditions on solving the KdV equation. In the PIMNs, the boundary conditions are not necessary due to the establishment of long-term dependencies. Next, we analyze the influence of boundary conditions on the training and results. The experimental setup are consistent with the previous subsection. And, four sets of experiments with different seeds of random numbers were also set up.

Figure 4 shows the loss curve, and subplots a–d and e–h correspond to the cases without boundary conditions and with boundary conditions, respectively. From figure 4, it can be seen that the total loss with and without boundary conditions can converge to the same level. Although the boundary loss without boundary conditions is more curved than the boundary loss with boundary conditions, the difference is not significant. Meanwhile, the red line shows that the influence of boundary loss is limited. In terms of the number of iterations, the boundary conditions do not accelerate the convergence of the network, but rather lead to a certain increase in iterations.

Table 2 gives the relative errors for the four sets of numerical experiments with and without boundary condition losses corresponding to figure 4. These two cases have very similar errors. the accuracy of the solution is not affected by boundary conditions. In general, since the influence of boundary conditions on both the training process and the relative L_2 errors is limited, the PIMNs is insensitive to the boundary conditions. But this does not mean that the boundary conditions are not important for solving PDEs, it only shows that the PIMNs can solve initial value problems for PDEs.

3.1.3. The effect of network structure for solving the KdV equation. In this part, based on the same training data, we investigated the effect of network structure on solving the KdV equation by setting different numbers of LSTM layers and hidden nodes. Complex networks tend to be more expressive, but also more difficult to train. The relative L_2 errors for different network structures are given in table 3. Among them, the number of hidden nodes does not include the last LSTM layer (the last LSTM hidden node is 1).

Table 3 shows experimental results of different network structures. When increasing the number of LSTM layers and

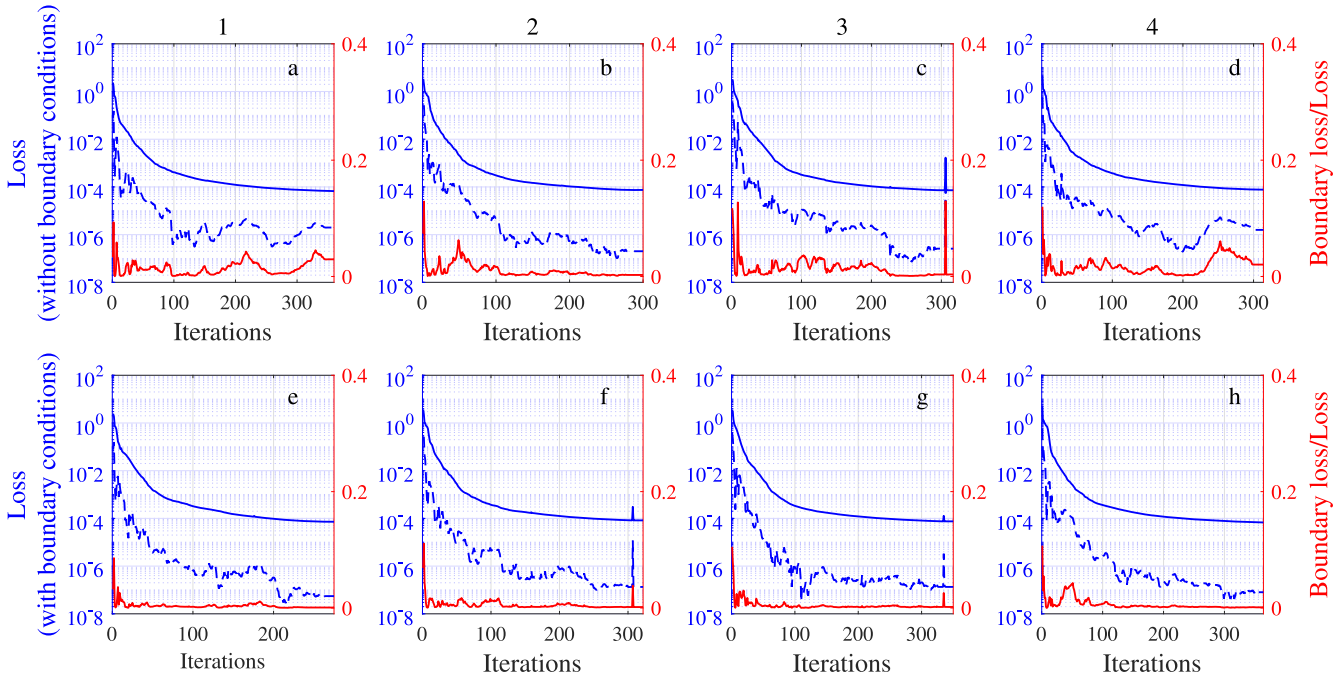

Figure 4. The loss curve: a–d are the loss curves without boundary conditions and e–h are the loss curves with boundary conditions. The blue solid line and the blue dashed line are MSE and MSE_B (the left y-axis), respectively, and the red line is the ratio of MSE_B to MSE (the right y-axis).

Table 2. The KdV equation: relative L_2 errors with and without boundary conditions.

	Relative L_2 errors (without boundary conditions)	Relative L_2 errors (with boundary conditions)
1	$1.690\,036 \times 10^{-3}$	$9.738\,059 \times 10^{-4}$
2	$8.357\,186 \times 10^{-4}$	$1.799\,449 \times 10^{-3}$
3	$1.661\,154 \times 10^{-3}$	$1.724\,861 \times 10^{-3}$
4	$1.835\,814 \times 10^{-3}$	$1.678\,390 \times 10^{-3}$

hidden nodes, the error shows a tendency to decrease. Although not all data fit this trend, it can still be argued that a complex network structure is beneficial to increase accuracy.

3.1.4. The effect of mesh size on solving the KdV equation. In this part, the impact of mesh size on errors is studied when the region is fixed as $[-10, 10] \times [0, 1]$. More temporal and spatial points represent smaller temporal and spatial steps and finer grids. In general, a finer grid produces a smaller truncation error in the difference schemes. This means numerical solutions with smaller errors. But, a fine grid also represents a large amount of training data, which is more demanding to train the model. The number of spatial points is set to 500, 1000, 1500 and 2000. The number of time points is set to 50, 100, 150 and 200. The network structure is chosen with 3 LSTM layers and the first 2 layers have 50 hidden nodes.

The errors at different mesh sizes are given in table 4. It can be observed that in the case of all time nodes, the error is minimum when the spatial node is 500. However, the change of time node does not have a regular effect on the error. To sum up, excessively increasing the grid number and decreasing grid size does not improve the accuracy of the solution for a fixed region.

Figure 5 shows the dynamic behavior of learned solution and the error density diagram when the time points are 150 and the spatial points are 500. The number of iterations is 622 and the training time is 91 s. The error density diagram shows that the error has remained very low and has not changed significantly over time. Therefore, PIMNs can still solve the KdV equation with high accuracy when only the initial conditions are available.

3.1.5. Comparison of the PINNs and the PIMNs for the KdV equation. In this part, the KdV equation is solved by the PINNs and the PIMNs with and without boundary conditions, respectively. To effectively compare these two methods, three sets of comparison experiments were set up and the number of parameters of the three cases is set close to each other. The number of hidden layers of the PINNs are 5, 7, 9 and the single layer neuron is 50. The numbers of parameters are 10 401, 15 501, 20 601. The number of initial and boundary points and collocations points are 100, 10 000, respectively. The structure of the PIMNs is two LSTM layers, and the first layer is 50, 60, 70 nodes, respectively. The numbers of parameters are 10 808, 15 368, 20 728. Table 5 shows the relative errors and number of parameters for the PINNs with boundary condition losses and the PIMNs with and without boundary condition losses.

In terms of the relative errors, all three cases are able to solve the KdV equation with high accuracy. Both the errors of the PIMNs with and without boundary

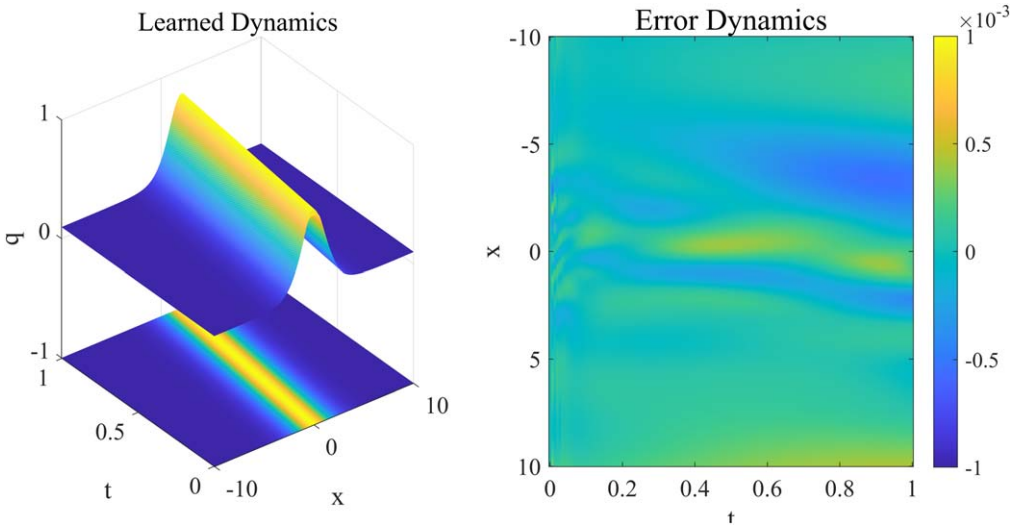

Figure 5. The traveling wave solution of the KdV equation: the dynamic behavior of learned solution and the error density diagram.

Table 3. The KdV equation: relative L_2 errors for different network structures.

Neurons /layers	2	3	4	5
10	$1.795\,948 \times 10^{-3}$	$1.375\,186 \times 10^{-3}$	$1.483\,687 \times 10^{-3}$	$1.562\,139 \times 10^{-3}$
30	$9.063\,210 \times 10^{-4}$	$1.149\,931 \times 10^{-3}$	$1.229\,779 \times 10^{-3}$	$1.364\,718 \times 10^{-3}$
50	$1.162\,169 \times 10^{-3}$	$5.524\,451 \times 10^{-4}$	$1.071\,582 \times 10^{-3}$	$1.361\,776 \times 10^{-3}$
70	$1.129\,416 \times 10^{-3}$	$1.078\,128 \times 10^{-3}$	$1.045\,698 \times 10^{-3}$	$8.195\,887 \times 10^{-4}$

Table 4. The KdV equation: relative L_2 errors for different mesh size.

Spatial points /time points	50	100	150	200
500	$9.534\,459 \times 10^{-4}$	$5.069\,412 \times 10^{-4}$	$5.014\,334 \times 10^{-4}$	$5.628\,230 \times 10^{-4}$
1000	$7.947\,627 \times 10^{-4}$	$8.739\,067 \times 10^{-4}$	$1.447\,893 \times 10^{-3}$	$8.306\,117 \times 10^{-3}$
1500	$1.884\,557 \times 10^{-3}$	$2.239\,517 \times 10^{-3}$	$1.932\,972 \times 10^{-3}$	$3.545\,602 \times 10^{-3}$
2000	$1.442\,287 \times 10^{-2}$	$9.473\,998 \times 10^{-3}$	$1.581\,677 \times 10^{-2}$	$1.531\,911 \times 10^{-2}$

Table 5. The KdV equation: relative L_2 errors for the PINNs and the PIMNs.

	PINNs		PIMNs without boundary conditions		PIMNs with boundary conditions	
	Relative L_2 error	Parameters	Relative L_2 error	Parameters	Relative L_2 error	Parameters
1	$1.977\,253 \times 10^{-3}$	10 401	$7.569\,623 \times 10^{-4}$	10 808	$5.168\,963 \times 10^{-4}$	10 808
2	$2.700\,721 \times 10^{-3}$	15 501	$6.007\,163 \times 10^{-4}$	15 368	$6.046\,472 \times 10^{-4}$	15 368
3	$1.685\,079 \times 10^{-3}$	20 601	$5.175\,313 \times 10^{-4}$	20 728	$5.448\,766 \times 10^{-4}$	20 728

conditions is lower than that of the PINNs when the number of parameters is close. This indicates that the structure of the PIMNs is more advantageous when reconstructing the solutions of the KdV equation. Also, consistent with the 3.1.2 subsection, the PIMNs with boundary conditions does not show a significant advantage over the PIMNs without boundary conditions. In summary, the PIMNs can simulate the solution of the KdV equation with only initial conditions, and even have higher accuracy than the PINNs.

Table 6. The nonlinear Schrödinger equation: relative L_2 errors for different difference schemes.

Difference schemes	Forward difference	Backward difference	Central difference
1	$\mathbf{2.604\,364 \times 10^{-3}}$	$3.259\,175 \times 10^{-3}$	$1.330\,293 \times 10^{-2}$
2	$3.527\,666 \times 10^{-3}$	$3.842\,570 \times 10^{-3}$	$\mathbf{2.265\,705 \times 10^{-3}}$
3	$2.735\,975 \times 10^{-3}$	$1.453\,902 \times 10^{-2}$	$\mathbf{2.476\,216 \times 10^{-3}}$
4	$1.529\,878 \times 10^{-2}$	$\mathbf{4.381\,561 \times 10^{-3}}$	$1.333\,741 \times 10^{-2}$

3.2. Case 2: Nonlinear Schrödinger equation

To test the ability of the PIMNs to handle complex PDEs, the nonlinear Schrödinger equation is solved. The nonlinear Schrödinger equation is often used to describe quantum behavior in quantum mechanics and plays an important role in the physical fields such as plasma, fluid mechanics, and Bose–Einstein condensates [43, 44]. The nonlinear Schrödinger equation is given by

$$\begin{aligned} \mathrm{i}q_t = q_{xx} + 2|q|^2 q,\ x \in [x_0, x_1],\ t \in [t_0, t_1], \\ q(x, t_0) = q_0(x), \\ q(x_0, t) = q_1(t), \\ q(x_1, t) = q_2(t), \end{aligned} \tag{18}$$

where q are complex-valued solutions, $q_0(x)$ is an initial function, and $q_1(x)$ and $q_2(x)$ are boundary functions. The existence theory of equation (18) can be referred to in the [45]. The complex value solution q is formulated as $q = u + \mathrm{i}v$, and $u(x, t)$ and $v(x, t)$ are real-valued functions of x, t. The equation (18) can be converted into

$$\begin{aligned} u_t &= v_{xx} + 2|u^2 + v^2|v, \\ v_t &= -u_{xx} - 2|u^2 + v^2|u. \end{aligned} \tag{19}$$

The residuals of equation (18) can be defined as

$$\begin{aligned} f_u &:= u_t - v_{xx} - 2|u^2 + v^2|v, \\ f_v &:= v_t + u_{xx} + 2|u^2 + v^2|u. \end{aligned} \tag{20}$$

Here, the traveling wave solution is simulated by the PIMNs and formed as

$$q(x, t) = \sqrt{c}\,\mathrm{sech}(\sqrt{c}\,(x - vt))\mathrm{e}^{\mathrm{i}(-\frac{v}{2})(x - vt) + \mathrm{i}\omega t}, \tag{21}$$

where $c = -\omega - v^2/4 > 0$. Taking $c = 0.8$, $v = 1.5$, the traveling wave solution equation (21) is reduced to

$$q(x, t) = \frac{2\sqrt{5}}{5}\,\mathrm{sech}\left(\frac{2\sqrt{5}}{5}(x - t)\right)\mathrm{e}^{-\frac{3}{4}\mathrm{i}x - \frac{49}{80}\mathrm{i}t}. \tag{22}$$

Taking $[x_0, x_1]$, $[t_0, t_1]$ as $[-10, 10]$, $[0, 1]$, the corresponding initial and boundary functions are obtained.

3.2.1. Comparison of different difference schemes for solving the nonlinear Schrödinger equation. Similar to the KdV equation, it is first discussed that which difference schemes should be used to calculate the temporal derivatives of the solution q. The space $[-10, 10]$ is divided into 1000 points and the time $[0, 1]$ is divided into 100 points. Two LSTM layers are used, and the number of nodes is 30 and 2, respectively. Table 6 shows the results generated by the four sets of random numbers.

In table 6, the bolded data are the lowest relative L_2 errors produced by the same random number. It can be seen that the temporal derivatives constructed by all three difference schemes can successfully solve the nonlinear Schrödinger equation with very small relative L_2 errors. And, result generated by the central difference performs better compared to the other two ways. Therefore, the central difference is used to calculate the time derivative in the subsequent subsections.

3.2.2. The effect of boundary conditions on solving the nonlinear Schrödinger equation. To investigate the effect of boundary conditions on the solution in the nonlinear Schrödinger equation, the training process and experimental results with and without boundary conditions are compared. The network structure and training data continue the previous setup. Four sets of experiments were also set up.

Figure 6 shows the loss curve of the training process. Subplots a–d show the loss curves without boundary conditions, and subplots e–h show the loss curves with boundary conditions. It is clear that the total loss convergence levels are close for the top and bottom. Although e–h have more iterations than a–d under the influence of the boundary conditions, all ratio are very low, less than 0.01. Therefore, the boundary conditions do not positively influence the training process.

Table 7 demonstrates the relative L_2 errors with and without boundary conditions corresponding to figure 6. After adding the boundary loss to the total loss, the error is kept at the original level. The influence of the boundary conditions on the errors remains limited. Since the boundary conditions do not positively affect either the training process or the results, they are not necessary for the PIMNs.

3.2.3. The effect of network structure for solving the nonlinear Schrödinger equation. In this part, we set different numbers of network layers and neurons to study that how the network structure affects the errors. The data are the same as those used before. Table 8 shows results of numerical experiments with different network structures. Note that the structures in the table 8 do not include the final LSTM layer.

From table 8, the error usually decreases as the number of neuron nodes and LSTM layers increases. Due to some chance factors, not all errors satisfy this law. In summary, complex structure of the PIMNs is more advantageous for solving differential equations.

3.2.4. Effect of mesh size on solving the nonlinear Schrödinger equation. In this part, we set different spatial and temporal

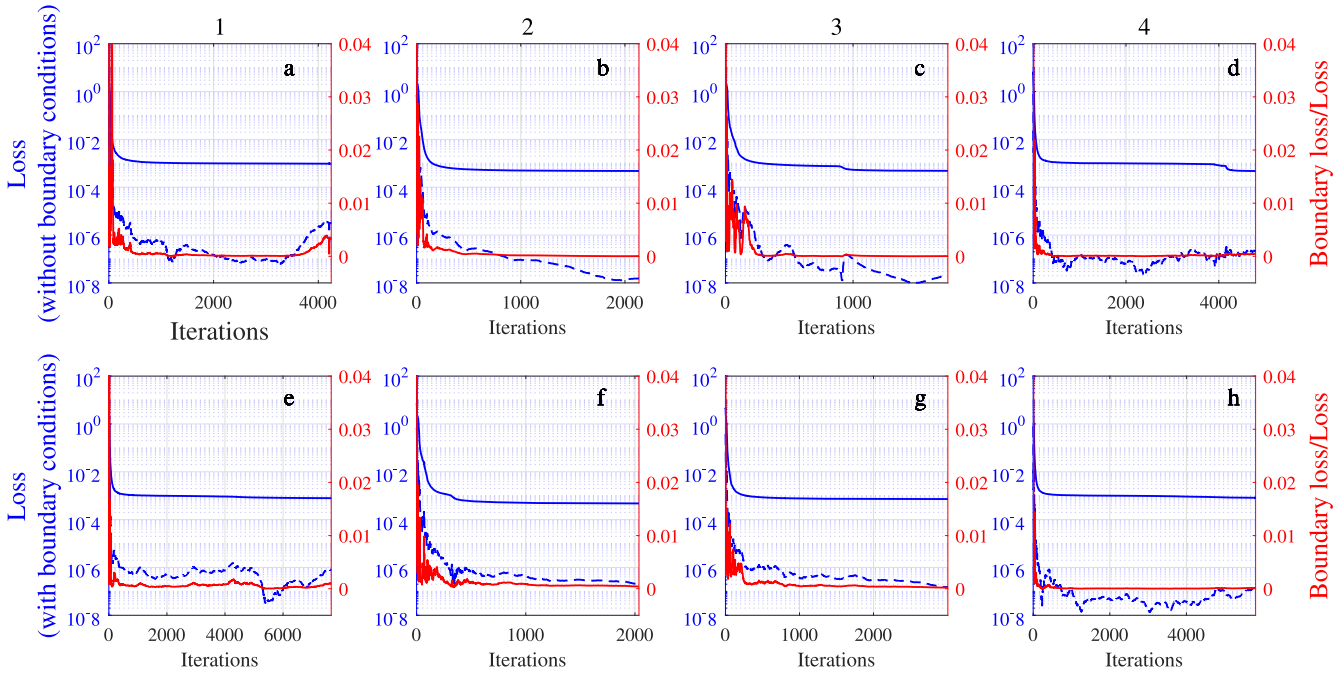

Figure 6. The loss curve: a–d are the loss curves without boundary conditions and e–h are the loss curves with boundary conditions. The blue solid line and the blue dashed line are MSE and MSE_B (the left y-axis), respectively, and the red line is the ratio of MSE_B to MSE (the right y-axis).

Table 7. The nonlinear Schrödinger equation: relative L_2 errors with and without boundary conditions.

	Relative L_2 errors (without boundary conditions)	Relative L_2 errors (with boundary conditions)
1	$1.352\,908 \times 10^{-2}$	$1.012\,920 \times 10^{-1}$
2	$2.350\,028 \times 10^{-3}$	$2.357\,040 \times 10^{-3}$
3	$2.757\,807 \times 10^{-3}$	$2.894\,889 \times 10^{-3}$
4	$2.364\,573 \times 10^{-3}$	$8.701\,724 \times 10^{-2}$

points to study that how the mesh size affects the errors. The region remains $[-10,\ 10] \times [0,\ 1]$. The number of spatial points is set to 500, 1000, 1500 and 2000, and the number of time points is set to 50, 100, 150 and 200. The network structure is 3 LSTM layers, and the nodes of the first two layers are 50. The relative L_2 errors for different mesh sizes are given in table 9.

As can be seen from table 9, the relative error reduces as the grid size decreases. However, the error is not minimal at a grid number of 2000×200. This indicates that the grid determines the relative L_2 error to some extent, and moderate adjustment of the mesh size can improve the accuracy of the solution.

Figure 7 shows the dynamic behavior of learned solution and the error density diagram when the time points are 100 and the spatial points are 2000. The iterations are 2013 and the training time is 225 s. From the error density, although the overall level of error is low, it also demonstrates an increasing trend over time. In general, the PIMNs can solve the nonlinear Schrödinger equation with high speed and quality.

3.2.5. Comparison of the PINNs and the PIMNs for nonlinear Schrödinger equation. In this part, the nonliner Schrödinger equation is solved using the PINNs and the PIMNs with and without boundary conditions, respectively. And the differences between the two models are discussed by comparing the relative errors. Similarly to 3.1.5, the parameters of the two models are set specifically. The PINNs has 5, 7, 9 layers with 50 nodes in each layer. The initial boundary points and the collocations points are 100 and 10 000, respectively. The PIMNs have two LSTM layers and the first layer has 50, 60, 70 nodes. Table 10 gives all the relative errors and the number of parameters.

From the data of table 10, both the PINNs and PIMNs can solve the nonlinear Schrödinger equation with very low error. All errors are very close except for individual experiments. The PINNs are more advantageous at the number of parameters around 10 000 and 20 000, and the PIMNs without boundary conditions have lower errors at the number of parameters around 15 000. That is, the PIMNs without boundary conditions can obtain similar results to the PIMNs with boundary conditions. This demonstrates the powerful generalization ability of the PIMNs when there is no boundary condition.

4. Conclusion

In this paper, the PIMNs are proposed to solve PDEs by physical laws and temporal structures. Differently from the PINNs, the framework of the PIMNs is based on LSTM, which can establish the long-term dependence of the PDEs' dynamic behavior. Moreover, the physical residuals are constructed using difference schemes, which are similar to finite difference method and bring better physical interpretation. To accelerate the network training, the difference schemes are implemented using the convolutional filter. The

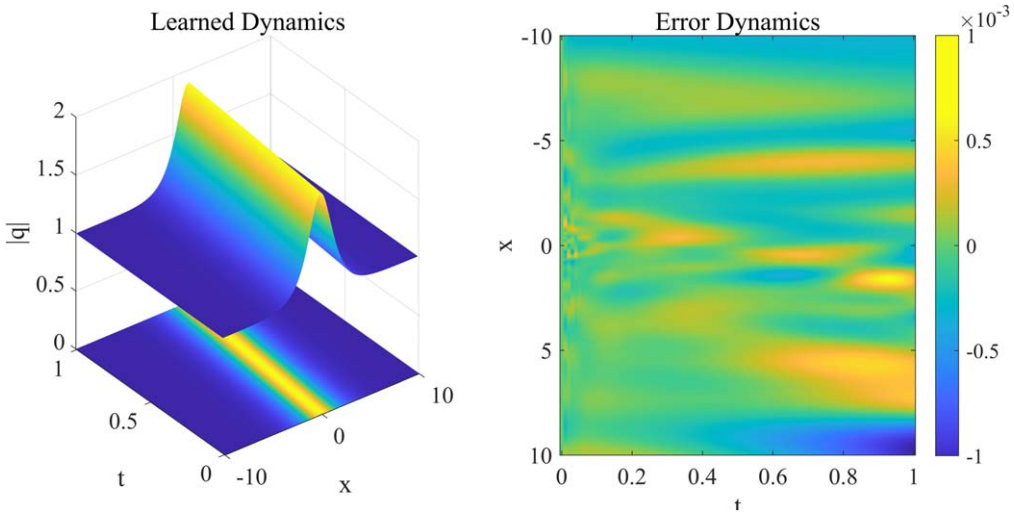

Figure 7. The traveling wave solution of the nonlinear Schrödinger equation: the dynamic behavior of learned solution and the error density diagram.

Table 8. The nonlinear Schrödinger equation: relative L_2 errors for different network structures.

Neurons/Layers	2	3	4	5
10	$2.548\,948 \times 10^{-3}$	$9.138\,814 \times 10^{-4}$	$1.605\,489 \times 10^{-3}$	$1.609\,296 \times 10^{-3}$
30	$2.434\,387 \times 10^{-3}$	$2.285\,506 \times 10^{-3}$	$7.020\,965 \times 10^{-4}$	$2.375\,666 \times 10^{-3}$
50	$2.686\,936 \times 10^{-3}$	$5.575\,241 \times 10^{-4}$	$5.776\,256 \times 10^{-4}$	$2.827\,150 \times 10^{-3}$
70	$2.246\,318 \times 10^{-3}$	$6.590\,603 \times 10^{-3}$	$2.362\,953 \times 10^{-3}$	$2.365\,898 \times 10^{-3}$

Table 9. The nonlinear Schrödinger equation: relative L_2 errors for different mesh sizes.

Spatial points\Time points	50	100	150	200
500	$1.082\,217 \times 10^{-3}$	$1.035\,790 \times 10^{-3}$	$1.109\,075 \times 10^{-3}$	$1.281\,616 \times 10^{-3}$
1000	$8.084\,908 \times 10^{-3}$	$8.685\,522 \times 10^{-4}$	$1.072\,378 \times 10^{-3}$	$8.839\,646 \times 10^{-4}$
1500	$6.154\,895 \times 10^{-4}$	$8.865\,785 \times 10^{-4}$	$9.732\,668 \times 10^{-4}$	$6.674\,587 \times 10^{-4}$
2000	$6.780\,904 \times 10^{-4}$	$\mathbf{6.010\,067 \times 10^{-4}}$	$7.655\,602 \times 10^{-3}$	$9.272\,919 \times 10^{-3}$

Table 10. The nonlinear Schrödinger equation: relative L_2 errors for the PINNs and the PIMNs.

	PINNs		PIMNs without boundary conditions		PIMNs with boundary conditions	
	Relative L_2 error	Parameters	Relative L_2 error	Parameters	Relative L_2 error	Parameters
1	$1.037\,740 \times 10^{-3}$	10 452	$2.203\,071 \times 10^{-3}$	11 024	$2.994\,355 \times 10^{-3}$	11 024
2	$1.794\,310 \times 10^{-3}$	15 552	$5.195\,681 \times 10^{-4}$	15 624	$3.743\,909 \times 10^{-1}$	15 624
3	$1.440\,035 \times 10^{-3}$	20 652	$2.324\,897 \times 10^{-3}$	21 024	$2.287\,124 \times 10^{-3}$	21 024

convolution filter is not involved in the model training and is only used to calculate the physical residuals. The performance and effectiveness of the PIMNs are demonstrated by two sets of numerical experiments. Numerical experiments show that the PIMNs have excellent prediction accuracy even when only the initial conditions are available.

However, the PIMNs use only second-order central differences and do not use higher-order difference schemes. And, solving higher-order PDEs is worth investigating. In addition, most physical information deep learning methods construct numerical solutions of PDEs. In the [46], a neural network model based on generalized bilinear differential operators is proposed to solve PDEs [46]. The method obtains a new exact network model solutions of PDEs by setting the network neurons as different functions. This proves that it is feasible to construct new exact analytical solutions of PDEs using neural networks. And how to construct new analytical solutions based on the PIMNs is a very worthwhile research problem.

Compared with the fully connected structure, it is difficult to set the LSTM units in the same layer as different functions. These are the main research directions for the future.

References

[1] Helal M A 2002 Soliton solution of some nonlinear partial differential equations and its applications in fluid mechanics *Chaos Solitons Fractals* **13** 1917

[2] Biswas A *et al* 2014 Singular solitons in optical metamaterials by ansatz method and simplest equation approach *J. Mod. Opt.* **61** 1550–5

[3] Parkins A S and Walls D F 1998 The physics of trapped dilute-gas Bose–Einstein condensates *Phys. Rep.* **303** 1

[4] Wang D, Guo B and Wang X 2019 Long-time asymptotics of the focusing Kundu–Eckhaus equation with nonzero boundary conditions *J. Differ. Equ.* **266** 5209

[5] Wang D, Xu L and Xuan Z 2022 The complete classification of solutions to the Riemann problem of the defocusing complex modified KdV equation *J. Nonlinear Sci.* **32** 3

[6] Xu L *et al* 2020 Exotic localized vector waves in a two-component nonlinear wave system *J. Nonlinear Sci.* **30** 537

[7] Bilman D, Buckingham R and Wang D 2021 Far-field asymptotics for multiple-pole solitons in the large-order limit *J. Differ. Equ.* **297** 320

[8] Zhang R and Bilige S 2019 Bilinear neural network method to obtain the exact analytical solutions of nonlinear partial differential equations and its application to p-gBKP equation *Nonlinear Dyn.* **95** 3041

[9] Li C and Chen A 2018 Numerical methods for fractional partial differential equations *Int. J. Comput. Math.* **95** 1048–99

[10] Dziuk G and Elliot C M 2012 Finite element methods for surface PDEs *Acta Numer.* **22** 289–396

[11] Young T *et al* 2018 Recent trends in deep learning based natural language processing *IEEE Comput. Intell. Mag.* **13** 55

[12] Karniadakis G E *et al* 2021 Physics-informed machine learning *Nat. Rev. Phys.* **3** 422

[13] Heinlein A *et al* 2021 Combining machine learning and domain decomposition methods for the solution of partial differential equations—a review *GAMM-Mitteilungen* **44** e202100001

[14] Winkler D A and Le T C 2017 Performance of deep and shallow neural networks, the universal approximation theorem, activity cliffs, and QSAR *Mol. Inf.* **36** 1600118

[15] Hornik K, Stinchcombe M and White H 1990 Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks *Neural Netw.* **3** 551

[16] Ranade R, Hill C and Pathak J 2021 Discretizationnet: a machine-learning based solver for Navier–Stokes equations using finite volume discretization *Comput. Meth. Appl. Mech. Eng.* **378** 113722

[17] Lu L *et al* 2021 Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators *Nat. Mach. Intell.* **3** 218–29

[18] Li Z *et al* 2020 Fourier neural operator for parametric partial differential equations arXiv:2010.08895

[19] Raissi M, Perdikaris P and Karniadakis G E 2019 Physics informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations *J. Comput. Phys.* **378** 686

[20] Yang L, Meng X and Karniadakis G E 2021 B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data *J. Comput. Phys.* **425** 109913

[21] Luo X and Kareem A 2020 Bayesian deep learning with hierarchical prior: predictions from limited and noisy data *Struct. Saf.* **84** 101918

[22] Jagtap A D, Kawaguchi K and Karniadakis G E 2020 Adaptive activation functions accelerate convergence in deep and physics-informed neural networks *J. Comput. Phys.* **404** 109136

[23] Lin S and Chen Y 2022 A two-stage physics-informed neural network method based on conserved quantities and applications in localized wave solutions *J. Comput. Phys.* **457** 111053

[24] Pu J and Chen Y 2022 Data-driven vector localized waves and parameters discovery for Manakov system using deep learning approach *Chaos Solitons Fractals* **160** 112182

[25] Lin S and Chen Y 2023 Physics-informed neural network methods based on Miura transformations and discovery of new localized wave solutions *Physica* D **445** 133629

[26] Pu J and Chen Y 2023 Complex dynamics on the one-dimensional quantum droplets via time piecewise PINNs *Physica* D **454** 133851

[27] Yu B 2018 The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems *Commun. Math. Stat.* **6** 1

[28] Samaniego E *et al* 2020 An energy approach to the solution of partial differential equations in computational mechanics via machine learning: concepts, implementation and applications *Comput. Meth. Appl. Mech. Eng.* **362** 112790

[29] Sirignano J and Spiliopoulos K 2018 DGM: a deep learning algorithm for solving partial differential equations *J. Comput. Phys.* **375** 1339

[30] Yazdani A *et al* 2020 Systems biology informed deep learning for inferring parameters and hidden dynamics *PLoS Comput. Biol.* **16** e1007575

[31] Henkes A, Wessels H and Mahnken R 2022 Physics informed neural networks for continuum micromechanics *Comput. Meth. Appl. Mech. Eng.* **393** 114790

[32] Ji W *et al* 2021 Stiff-pinn: Physics-informed neural network for stiff chemical kinetics *J. Phys. Chem.* A **125** 8098

[33] Zhu Y *et al* 2019 Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data *J. Comput. Phys.* **394** 56

[34] Ren P *et al* 2022 PhyCRNet: physics-informed convolutional-recurrent network for solving spatiotemporal PDEs *Comput. Meth. Appl. Mech. Eng.* **389** 114399

[35] Mohan A *et al* 2019 Compressed convolutional LSTM: an efficient deep learning framework to model high fidelity 3D turbulence arXiv:1903.00033

[36] Stevens B and Colonius T 2020 FiniteNet: a fully convolutional LSTM network architecture for time-dependent partial differential equations arXiv:2002.03014

[37] Fang Z 2021 A high-efficient hybrid physics-informed neural networks based on convolutional neural network *IEEE Trans. Neural Netw. Learn. Syst.* **33** 5514

[38] Hochreiter S and Schmidhuber J 1997 Long short-term memory *Neural Comput.* **9** 1735

[39] Gardner C S *et al* 1967 Method for solving the Korteweg–deVries equation *Phys. Rev. Lett.* **19** 1095

[40] Miura R M 1976 The Korteweg–deVries equation: a survey of results *SIAM Rev.* **18** 412

[41] Dutykh D, Chhay M and Fedele F 2013 Geometric numerical schemes for the KdV equation *Comput. Math. Math. Phys.* **53** 221

[42] Holmer J 2006 The initial-boundary value problem for the Korteweg–de Vries equation *Commun. Partial Differ. Equ.* **31** 1151

[43] Feit M D Jr, Fleck J A and Steiger A 1982 Solution of the Schrödinger equation by a spectral method *J. Comput. Phys.* **47** 412

[44] Solli D R *et al* 2007 Optical rogue waves *Nature* **450** 1054

[45] Ramos J I and Villatoro F R 1994 The nonlinear Schrödinger equation in the finite line *Math. Comput. Model.* **20** 31

[46] Gai L, Ma W and Sudao B 2021 Abundant multilayer network model solutions and bright-dark solitons for a (3 + 1)-dimensional p-gBLMP equation *Nonlinear Dyn.* **106** 867