Electronic Research Archive

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

ERA, 32(12): 6641–6659.
DOI: 10.3934/era.2024310
Received: 03 September 2024
Revised: 31 October 2024
Accepted: 03 December 2024
Published: 09 December 2024

Research article

PhyICNet: Physics-informed interactive learning convolutional recurrent network for spatiotemporal dynamics

Ruohan Cao[1], Jin Su[1,2,*], Jinqian Feng[1] and Qin Guo[1]

[1] School of Science, Xi'an Polytechnic University, Xi'an 710048, China
[2] Xi'an International Science and Technology Cooperation Base for Big Data Analysis and Algorithms, Xi'an 710048, China

* **Correspondence:** Email: sujin@xpu.edu.cn.

Abstract: The numerical solution of spatiotemporal partial differential equations (PDEs) using the deep learning method has attracted considerable attention in quantum mechanics, fluid mechanics, and many other natural sciences. In this paper, we propose an interactive temporal physics-informed neural network architecture based on ConvLSTM for solving spatiotemporal PDEs, in which the information feedback mechanism in learning is introduced between the current input and the previous state of network. Numerical experiments on four kinds of classical spatiotemporal PDEs tasks show that the extended models have superiority in accuracy, long-range learning ability, and robustness. Our key takeaway is that the proposed network architecture is capable of learning information correlation of the PDEs model with spatiotemporal data through the input state interaction process. Furthermore, our method also has a natural advantage in carrying out physical information and boundary conditions, which could improve interpretability and reduce the bias of numerical solutions.

Keywords: partial differential equations; physics-informed deep learning; data-driven modeling; interactive learning; neural network

1. Introduction

Spatiotemporal dynamics modeled by partial differential equations (PDEs) are ubiquitous in nature, and obtaining their numerical solution has proved difficult due to their inherent complexity and nonlinearity. Traditional numerical methods (such as finite element/difference/volume method [1–3] and geometric analysis [4]) are limited in their efficiency due to the complexity of the algorithm, which is inadaptive to tackling large-scale computations and complicated boundary conditions.

With the advancement of science and technology, deep learning provides a new perspective for the numerical study of nonlinear system models, which can accelerate the discovery of potential PDEs solutions. Recently, a series of methods based on neural networks have been developed to solve differential equations by learning input and output mapping functions. These methods have been applied to solving PDEs, addressing forward and inverse problems in uncertainty quantification, and other tasks closely related to differential equations [5, 6]. Nevertheless, most deep learning methods do not meet physical constraints. The physical information neural network (PINN) [7–9] has led to great breakthroughs in the field of scientific machine learning, such as in the reduced order model [10, 11], solving general PDEs [12], data-driven knowledge discovery [13], which uses prior knowledge of physical information to learn within a small data range [14–16]. However, most of these PINNs were constructed in a fully connected neural network (FCNN) to continuously approximate the solution of the physical system. This results in physical processes and locality being ignored, leading to training models lacking temporal and spatial extrapolation capabilities.

Recently, some new methods for solving PDEs were obtained [17, 18] with the introduction of recurrent neural networks (RNN) [19] and convolutional neural networks (CNN) [20]; from these, discrete learning models show better scalability and faster convergence [21]. The hard encoding mode of physics can embed the initial and boundary conditions (I/BCs) and incomplete PDE structures into the learning process, which can avoid the ill-posedness of optimization, even without any labeled data [22–24]. For time-independent systems, Han et al. [25] proposed a new method named PhyGeoNet, which uses a convolution kernel with fixed weights as a differential operator to construct CNN. Recently, more and more studies have focused on time-dependent systems [26–36]. Hu et al. [33] developed a sequence to sequence learning (Seq2Seq) framework called Neural PDEs, which allows automatic learning of the control rules of any time-dependent PDEs system from existing data by using a bidirectional long short-term memory (LSTM) encoder. Ren et al. [34] developed a physical information convolutional recurrent network (PhyCRNet), which is an unsupervised learning method for solving multi-dimensional spatiotemporal PDEs through physical prior knowledge and CNNs architecture. Jiang et al. [35] proposed a physical information graph neural network (PhyGNNet) for solving spatiotemporal PDEs based on the graph neural network composed of encoder, processor and decoder modules. The computational domain of PDEs is divided into a regular grid, and then the spatial differential and the time derivative with backward difference are constructed on the grid. Then, the grid is regarded as a graph, and the PDEs loss is organized to train the graph neural network. The network has the reasoning ability to satisfy the solution of the PDEs equation. Meng et al. [36] developed a real-time parallel physical information neural network (PPINN), which decomposes a long-term problem into many independent short-term problems and is supervised by a coarse-grained solver.Compared with the original PINN method, the proposed PPINN method can accelerate the long-term integration of PDEs. Although existing methods have successfully dealt with spatial and temporal dependencies by using specific network structures, they find it challenging to deal with PDEs with complex dynamic behaviors and long-distance dependencies. To improve the application scope of PINN in practice, this article tries to develop a novel physics-informed interactive learning convolutional recurrent network (PhyICNet) method to solve spatiotemporal PDEs.

Relevant literature shows that the application of ConvLSTM to systems such as 2D coupled viscous Burgers, heat diffusion systems,reaction-diffusion equations, wave propagation systems has achieved good results [37–39]. Inspired by the works of multiplicative integration LSTM [40] and Mogrifier LSTM [41], we included the existing recursive unit and an interactive learning mechanism into the physical information convolution recursive model based on the current input x and the previous time step output h. After several rounds of interactive iteration, the updated x and h enter the recursive unit together. From a more comprehensive perspective, our model can be regarded as enriching the main additive dynamics of cyclic conversion. The main contributions of this paper are as follows:

- Our model combines the advantages of CNN and RNN through interactive learning strategies. It not only improves the spatial characteristics expression and time modeling ability of the model but also strengthens the information association ability of the model in dealing with spatiotemporal data through the input state interaction process.
- Compared with the continuous approximation PINN method, our physical model is discrete (that is, the solution is based on the spatial grid and defined on the discrete time step). In particular, the initial/boundary conditions (I/BCs) are hard-coded into the network, so that the solution accuracy on the boundary is improved through hard-applied physical constraints, and the time evolution dynamics of PDEs can be strictly mapped in combination with global residual connections.
- We tested PhyICNet on 4 representative $2D$ PDEs problems and compared it with the benchmark model in various aspects. We found that PhyICNet outperforms the benchmark model in all comparisons. Therefore, we propose a new PINN model based on interactive learning, which has ideal theoretical properties and excellent empirical performance, and has the potential to be widely used to learn PDEs.

The rest of the article is organized as follows: Section 2 proposes the use of deep neural networks to solve spatiotemporal dynamic system problems; Section 3 mainly introduces the PhyICNet method and its network architecture. Section 4 provides the experimental part, in which we compare PhyICNet the method with the benchmark network through several classical numerical experiments to prove its effectiveness. Section 5 provides a summary of this article and discusses future prospects.

2. Theoretical background

Consider a set of spatiotemporal dynamical systems described by nonlinear coupled PDEs:

$$u_t = F(x, t, u, u^2, \nabla_x u, u \cdot \nabla_x u, \nabla^2 u, \cdots). \tag{2.1}$$

where $u(x, t) \in R^n$ denotes the state variable with n components defined on the spatiotemporal domain $\{(x, t)\} \in \Omega \times T$, Ω and T denote the spatial and temporal domains, respectively, ∇_x is the Nabla operator with respect to the spatial coordinate x, and $F(\cdot)$ is a nonlinear function describing the right-hand side (RHS) of PDEs. The solution of this problem is subject to initial conditions (ICs): $I(u; t = 0, x \in \Omega) = 0$ and boundary conditions (BCs): $B(u, \nabla_x u, \cdots; x \in \partial\Omega) = 0$. This paper focuses on the regular physical domain, so the state variables are defined on the discrete Cartesian grid.

Referring to the concept of numerical discretization, we aim to develop a spatiotemporal learning model for physical information based on a forward Eulerian format. That is, the state variable u will be updated by a recurrent network: $u^{(k+1)} = u^{(k)} + F^*(u^{(k)}; \theta)$, where $u^{(k)}$ is the prediction at moment t_k and $F^{(*)}$ is an F parametrized by θ, which integrates a set of operations to compute the RHS of the equation. Similar concepts of applying numerical discretization to design deep learning architectures can be found in some recent literature [42–44]. Thus, we consider network training as an optimization process by minimizing a loss function that consists of discrete PDEs residuals subject to I/BCs.

3. PhyICNet

In this section, we focus on the architecture of the PhyICNet model, which is designed to efficiently capture and integrate complex dynamic behaviors in spatiotemporal sequences. The network structure of PhyICNet is shown in Figure 1, which includes ConvLSTM time propagator, interactive learning module, BCs hard constraints, and physics-informed loss function.

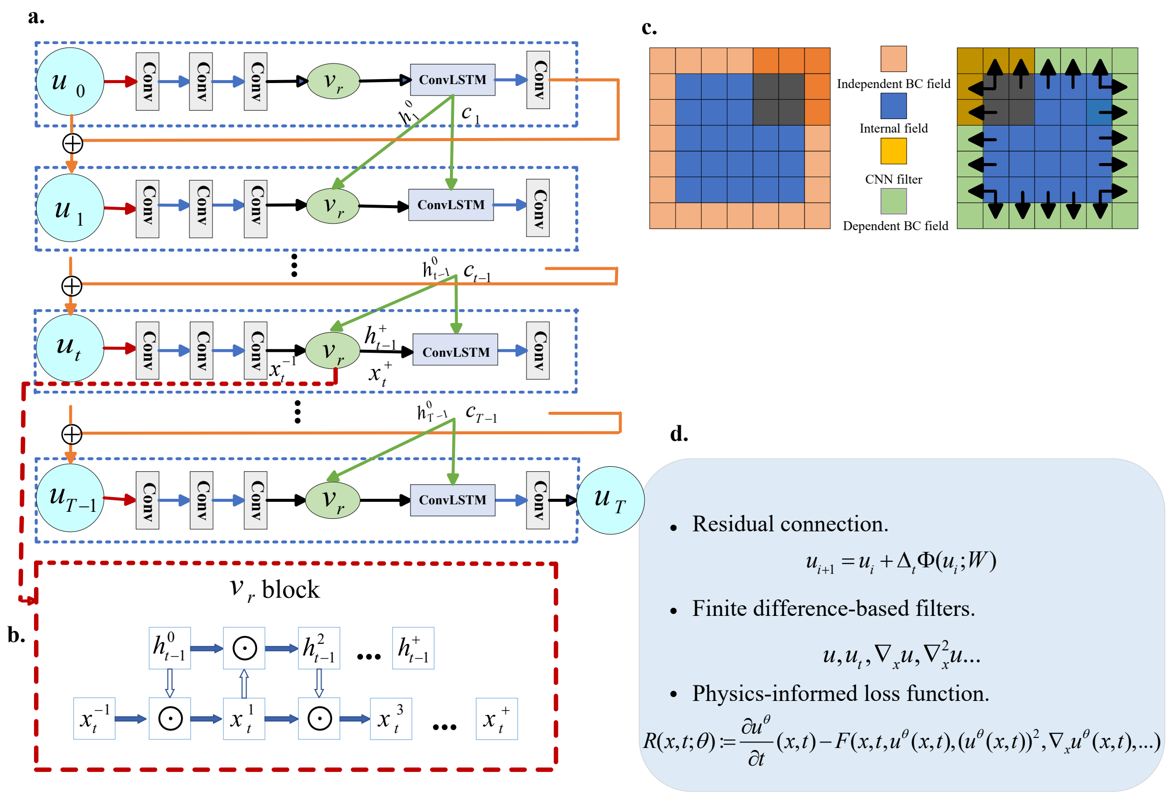


Figure 1. PhyICNet architecture. a. By learning the known state variable u_0 through T time steps in multiple convolutional layers, interactive learning modules, and ConvLSTM time propagator, the bounded output u_T is obtained. b. PhyICNet is updated through r rounds of interaction. c. Padding implements BCs hard constraints. Dirichlet BCs (left) and Neumann BCs (right). d. Key features of the network.

The following is a detailed description of the PhyICNet architecture:

Encoder. Considering that the changes of spatial variables in spatiotemporal PDEs have local correlation, PhyICNet incorporates spatial local convolution operation to take advantage of this structural feature. In the process of updating the hidden state, PhyICNet not only depends on the input of the current node itself but also integrates the input information of the nodes in its neighborhood through convolution operation. Convolution kernel parameter sharing and local receptive field characteristics ensure that the model can extract and utilize local features in the spatial domain, thereby improving the sensitivity and adaptability to spatial changes. The spatial local modeling process can be expressed as:

$$y_{i,j} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} w_{m,n} \cdot x_{i+m,j+n} + b. \tag{3.1}$$

where $y_{i,j}$ is the value of the output feature map at position (i, j), $x_{i+m,j+n}$ is the value of the input feature map at the position (m, n) deviation relative to the position (i, j), $w_{m,n}$ is the weight of the convolution kernel at position (m, n), and b is the bias term of the convolution layer (optional).

PhyICNet can effectively extract the local features of spatial variables in spatiotemporal PDEs by introducing convolution operations, and have the ability to use parameter sharing and local receptive field characteristics to improve the spatial modeling performance of the model.

In the initial stage of the model, we deployed an encoder module consisting of three consecutive convolutional layers. This series of convolution operations aim to extract deep features of the input state, where each convolutional layer is equipped with appropriate filter size, step size, and filling to ensure that more and more advanced feature representations can be abstracted layer by layer while maintaining spatial information. At this stage, the input data is decomposed into multiple time steps (T) along the time dimension, and the feature map of each time step is encoded to form a low-dimensional and compact potential feature sequence.

Interactive learning module. When dealing with spatiotemporal PDEs, the interaction between data points goes beyond simple single direction or local dependence, showing complex nonlocal and nonlinear characteristics. In order to capture dynamic interactivity, PhyICNet introduces an interactive hidden state update mechanism. Its innovation is to use the multiplicative interaction to update the hidden state, which can be described by the intermediate state u_t:

$$\begin{aligned} u_t &= (W_{ux}x_t) \odot (W_{uh}h_{t-1}), \\ \widehat{h_t} &= W_{hu}u_t + W_{hx}x_t. \end{aligned} \tag{3.2}$$

where W_{ux}, W_{uh}, W_{hu}, W_{hx} are the weight matrix of different transformations; u_t reflects the deep interaction between the current input x_t and the hidden state h_{t-1} at the previous moment; and the nonlinear effect between them is strengthened by the element-by-element product operation. This innovation gives PhyICNet a stronger ability to express spatiotemporal patterns, making it superior to traditional models in mining complex interactive features in sequence data.

Specifically, before entering the recursive unit, interactive learning is performed on the input data x (such as the I/BCs of PDEs) and the hidden state h (the internal state of the recursive unit), where the two inputs x and h are alternately adjusted before the recursive unit calculation occurs. That is:

$$v_r = g_r(x_t, h_{t-1}, r). \tag{3.3}$$

where g_r is the spatiotemporal feature interaction function. The interactive learning stage is to perform r-round interaction on the spatial position and the time step, and the adjustment input is defined as the highest index x^i and h^i from the interaction sequence, respectively:

$$x^i = 2\sigma(weightQ^i h^{i-1}) \odot x^{i-2}, for\ odd\ i \in [1, \ldots, r],$$
$$h^i = 2\sigma(weightR^i x^{i-1}) \odot h^{i-2}, for\ even\ i \in [1, \ldots, r]. \tag{3.4}$$

where $x^{-1} = x$, $h^0 = h$, and the number of rounds r is a hyperparameter. The multiplication with constant 2 ensures that the transformation generated by the randomly initialized Q, R weight matrix is close to the identity transformation. In order to reduce the number of parameters, we usually set Q, R as a low rank matrix.

The framework of interaction is summarized in Algorithm 1. Through the understanding of the characteristics of spatiotemporal PDEs and the reference of deep learning technology, we aim to build a deep learning model that can accurately capture spatiotemporal interactions and effectively use spatial local information.

Algorithm 1 Interactive learning

```
Require: Current input $x_t$, previous hidden state $h_{t-1}$, previous cell state $c_{t-1}$, number of interactions $r$
Ensure: Updated input $x_t^+$, updated hidden state $h_{t-1}^+$, updated cell state $c_{t-1}$
1: Initialize $x^{-1} \leftarrow x_t$, $h^0 \leftarrow h_{t-1}$
2: for $i = 1, \ldots, r$ do
3:     if $i$ is odd then
4:         $x^i = 2\sigma(weightQ^i h^{i-1}) \odot x^{i-2}$
5:     else
6:         $h^i = 2\sigma(weightR^i x^{i-1}) \odot h^{i-2}$
7: Apply $x_t^+$, $h_{t-1}^+$, $c_{t-1}$ as ConvLSTM input for the next round of updates
```

Following the encoder is the interactive learning module, whose function is to promote the information exchange and deep integration between the input feature x and the previous hidden state h. Through this module, the following goals are achieved: (1) Guide the model to focus on the most relevant and influential feature subsets in the current input and historical state, and improve the recognition accuracy of spatiotemporal dependencies. (2) The historical information is used to supplement the deficiency of the current input, and the historical state is dynamically adjusted according to the new input, so as to generate the fusion features that contain the global historical context and reflect the current local changes. (3) The interactive learning process generates new features that go beyond the individual input or hidden state, which may capture deeper patterns and long-term dependencies in the spatiotemporal sequence.

ConvLSTM time propagator. As we know, LSTM is an improved model to overcome the gradient problem of RNN, which effectively manages information storage and discarding by introducing memory cells and a multilayer gating mechanism. In the recursive unit of our method, we extract spatial features through local convolution operation and capture time dependence through the memory unit of LSTM. At the same time, combined with the idea of multiplicative interaction, spatiotemporal interactive learning will be performed before the recursive unit. The mathematical expression of the recursive unit is as follows:

$$\begin{aligned} i_t &= \sigma(W_{ix} * x_t + W_{ih} * h_{t-1} + b_i); \\ f_t &= \sigma(W_{fx} * x_t + W_{fh} * h_{t-1} + b_f); \\ c_t &= f_t \odot c_{t-1} + i_t \odot \tanh(W_{cx} * x_t + W_{ch} * h_{t-1} + b_c); \\ o_t &= \sigma(W_{ox} * x_t + W_{oh} * h_{t-1} + b_o); h_t = o_t \odot \tanh(c_t). \end{aligned} \tag{3.5}$$

where x_t denotes the input tensor; and $\{h_t, c_t\}$ denotes the hidden state and the cell state to be updated at time t, respectively; f_t,i_t,c_t,o_t is the forget gate, the input gate, the internal cell, and the output gate, respectively, which can achieve the transformation of ConvLSTM from input to state and state to state. "$*$" is a convolution operation, $\odot$ denotes the Hadamard product, and W_* and b_* are the weight matrix and bias, respectively.

Then, the obtained intermediate state is used to enter the ConvLSTM unit for state update after completing multiple rounds of interactive learning of spatiotemporal features:

$$\begin{aligned} ConvLSTM(x_t^+, c_{t-1}, h_{t-1}^+) &= v_r; \\ i_t &= \sigma(W_{ix} * x_t^+ + W_{ih} * h_{t-1}^+ + b_i); \\ f_t &= \sigma(W_{fx} * x_t^+ + W_{fh} * h_{t-1}^+ + b_f); \\ c_t &= f_t \odot c_{t-1} + i_t \odot \tanh(W_{cx} * x_t^+ + W_{ch} * h_{t-1}^+ + b_c); \\ o_t &= \sigma(W_{ox} * x_t^+ + W_{oh} * h_{t-1}^+ + b_o); h_t = o_t \odot \tanh(c_t). \end{aligned} \tag{3.6}$$

Specifically, we first use CNN to extract the input spatial features, and then combine the extracted features with the traditional LSTM to perform time series modeling. On this basis, we introduce a new interactive learning mechanism, so that x and h can effectively exchange and integrate information before ConvLSTM, thereby improving the model's solving ability.

The rich features obtained by interactive learning are then sent to the ConvLSTM module. In this module, each ConvLSTM unit is not only responsible for disseminating information on the time axis, but also performs parallel information processing on the spatial dimension through convolution kernels, laying the foundation for subsequent prediction tasks.

Decoder and recovery layer. After the spatiotemporal modeling is completed, the model uses an additional convolutional layer as the recovery layer, which aims to map the boundedness processed by ConvLSTM back to the same spatial resolution as the original input. This step involves upsampling operations to ensure that the final output and input data are comparable in the spatial dimension, thereby facilitating pixel-level prediction and reconstruction.

Residual connection. Inspired by the forward Euler scheme, a residual connection based on time increment is constructed between the current state u_i and the next state u_{i+1}. This process is understood as the following transformation from time t_i to time t_{i+1}:

$$u_{i+1} = u_i + \Delta_t \Phi(u_i; W). \tag{3.7}$$

where Φ represents the function module obtained by training optimization, W is the parameter set of the module, and Δ_t indicates a small step at two time points. Once the state u_{i+1} is obtained in t_{i+1}, it becomes the input state of t_{i+1}. Therefore, this continuous transmission of input and output can be regarded as a time series autoregressive (AR) model.

Coding physics prior knowledge. In the network design, we integrate an innovative coding strategy so that the system can strictly apply physical knowledge. We consider physical information encoding: requirements for I / BCs.

PhyICNet is calculated from the initial set of u_0, and ICs are directly integrated into the algorithm. Considering BCs, we use the finite difference method to guide the filling strategy through physical principles, and adjust the model prediction at each time step. For Dirichlet BCs, we directly use the preset value to fill the boundary part of the prediction result; for Neumann BCs, the filling value is calculated based on the boundary value and its gradient information. Here, we mainly consider the application of periodic Dirichlet BCs. This design has been proved to be very applicable in [45, 46]. We use a periodic filling scheme, as shown in Figure 1, which uses the boundary information on the other side of the grid for prediction and feature mapping. This is significantly better than simple zero filling, because it effectively prevents the loss of information on the physical boundary.

In this study, we use u_0 to represent the initial state, and $u_1, u_2, \cdots, u_T$ are a series of discrete solution variables to be solved. Next, the key problem to be solved is the calculation of the derivative term. The gradient-free convolution method proposed in [21, 25] is used to realize the discrete numerical approximation of the relevant derivative terms. Specifically, the convolution kernel based on finite difference includes: the second-order scheme for time derivative Eq (3.8) and the fourth-order central difference method for spatial derivative calculation Eq (3.9).

$$K_t = [-1, 0, 1] \times \frac{1}{2dt}. \tag{3.8}$$

$$K_x = \begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 16 & 0 & 0 \\ -1 & 16 & -60 & 16 & -1 \\ 0 & 0 & 16 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix} \times \frac{1}{12(dx)^2}. \tag{3.9}$$

where dt and dx represent the temporal and spatial distances, respectively.

In the field of scientific modeling, this method of encoding prior physical knowledge into network architecture is very useful for model prediction. We can predict some physical laws from previous studies and encode them into physical models so that the models can strictly follow physical laws.

Physics-informed loss function. In our method, we follow the principle of physical constraints and focus on the strict satisfaction of I/BCs, so we only need to construct a loss function that only depends on PDEs themselves. Firstly, PDEs residual expression $R(x, t; \theta)$ is designed, which is based on the output of the network model and satisfies Eq (3.10), as shown below:

$$R(x,t;\theta) := \frac{\partial u^{\theta}}{\partial t}(x,t) - F\left(x, t, u^{\theta}(x,t), (u^{\theta}(x,t))^2, \nabla_x u^{\theta}(x,t), u^{\theta}(x,t) \cdot \nabla_x u^{\theta}(x,t), \cdots\right). \tag{3.10}$$

Then, the network sharing parameter θ is adjusted to minimize a loss function $L(\theta)$ that reflects the physical law. This optimization process is actually the sum of the square values of the PDEs residuals at each point on the spatiotemporal discretization grid. Taking a typical $2D$ PDE problem as an example, the loss function $L(\theta)$ can be expressed as:

$$L(\theta) = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{T} ||R(x_i, y_i, t_k; \theta)||_2^2. \tag{3.11}$$

where n and m represent the resolution of the spatial grid in the vertical and horizontal directions, respectively, T represents the total number of time steps, and $|| \cdot ||$ represents the L_2 norm.

Therefore, compared with the PINN model, PhyICNet has the following advantages: (1) Limitations of data-driven methods. Although traditional data-driven methods can capture certain dynamic features when processing spatiotemporal data, their generalization ability and accuracy are often limited in the absence of sufficient training data or in the face of complex physical phenomena. (2) The importance of physical information. PINN ensures that the model prediction results conform to the physical law by adding physical constraints in the training process, thus improving the reliability and robustness of the model. However, PINN may be insufficient in dealing with spatiotemporal data, especially in capturing dynamic interactions. (3) Advantages of interactive learning. Our proposed interactive learning mechanism aims to make the model better capture dynamic features when dealing with spatiotemporal data by adding dynamic interaction capture to the model. Specifically, the interactive learning mechanism allows the model to effectively exchange information and feedback between different time steps, thereby improving the predictive ability of the model. Our model not only combines physical information but also further enhances the model's ability to capture spatiotemporal dynamics through interactive learning mechanisms. This combination makes the model more advantageous in dealing with complex dynamic systems.

In summary, the network structure of PhyICNet not only shows the local advantages of convolution operation in dealing with spatial data but also takes advantage of the excellent performance of LSTM in dealing with time series data. The biggest advantage is the ability to capture the complex interaction between features, which makes it effective in solving sequence problems with spatial correlation and time dependence such as PDEs. In this way, our model is expected to show strong performance when dealing with sequence data with high-intensity dynamic characteristics and nonlinear dependencies.

4. Experimental verification

To verify the effectiveness of the PhyICNet method proposed in this paper, we choose four classical PDEs for experimental verification, namely the Burgers' equation, the FitzHugh-Nagumo reaction-diffusion (FN RD) equation, the nonlinear Schrodinger (NLS) equation, and the Navier-Stokes (NS) equation, which have important application value in the fields of science and engineering and have proven to be difficult to solve.

4.1. Network setup

We consider that the PhyICNet network encoder part consists of three convolutional layers, namely 8, 32, and 128 units, using 4×4 kernels and a stride of 2, with all convolution kernels being periodically filled. ConvLSTM [34] has 128 node units and uses a convolution kernel with a step size of 1 kernel and a stride of 3 × 3. For the scaling layer before output, we use a kernel size of 5 × 5. In the first three experiments in this section, we conduct interactive learning 4 times before the input x and hidden state h of PhyICNet entered ConvLSTM. The network is trained by the stochastic gradient descent Adam optimizer [47]. All our experiments are coded in Pytorch [48] and executed on A6000 GPU card (48G).

4.2. Error analysis of FN equation solution evolving with time

In order to explore the performance difference between PhyICNet and baseline network in solving the FN RD equation, we analyze the error growth between the true value and the predicted value as time T evolves. The FN RD equation is a nonlinear PDEs, which describes the periodic oscillation of neuronal action potential under constant current stimulation above the threshold. It has important theoretical and practical significance in neuroscience research. We mainly discuss a $2D$ FN RD equation, which is expressed as follows:

$$\begin{aligned} u_t &= Du\Delta u + u - u^3 - v + \alpha, \\ v_t &= Dv\Delta v + \beta(u - v). \end{aligned} \tag{4.1}$$

where Du and Dv are diffusion coefficients, and α and β are reaction coefficients; we set the parameters $\alpha = 0.01$, $\beta = 0.25$, $Du = 10$, and $Dv = 25$. The computational domains of $\Omega \in [0, 64/3]$ and $T \in [0, 12]$ are discretized by the interval of $dx = 1/6$ and $dt = 0.0002$. The ICs of the problem are generated by random sampling from the Gaussian distribution, and the BCs are periodically filled. The learning rate is set to 1×10^{-4}, and the decay rate per 50 epochs is 0.995.

We generate the real solutions of the FN RD equation at different times T and use ConvLSTM and PhyICNet to predict. For each time step T, we calculate the error between the two network models' true and predicted solutions.

In this case, we focus on the performance comparison of ConvLSTM and PhyICNet architecture in time series prediction tasks. The prediction error measurement data of the two network models in three randomly selected time periods of $T = 3, 7, 12$ are recorded. Based on these data, we draw Figure 2, and calculate that at $T = 3$, the mean square error (MSE) of ConvLSTM and PhyICNet relative to u is 1.8e-03 and 1.6e-03, respectively, and the MSE relative to v is 2.4e-04 and 2.3e-04, respectively; at $T = 7$, the MSE of ConvLSTM and PhyICNet relative to u is 5.5e-03 and 5.0e-3, respectively, and the MSE relative to v is 7.6e-04 and 7.4e-04, respectively; at $T = 12$, the MSE of ConvLSTM and PhyICNet relative to u is 1.1e-02 and 9.6e-03, respectively, and the MSEs relative to v is 1.48e-03 and 1.44e-03, respectively. According to the error map drawn in Figure 2, it is found that PhyICNet is always superior to ConvLSTM as time T increases. This shows that PhyICNet has better stability and accuracy in long-term simulation.

Through the error analysis of the solution of the $2D$ FN RD equation over time, we verify that compared with ConvLSTM, PhyICNet can more effectively reduce the error and improve the prediction accuracy. This result is of great significance for scenarios that require long-term simulation in practical applications, indicating that PhyICNet has better applicability in solving spatiotemporal nonlinear PDEs.

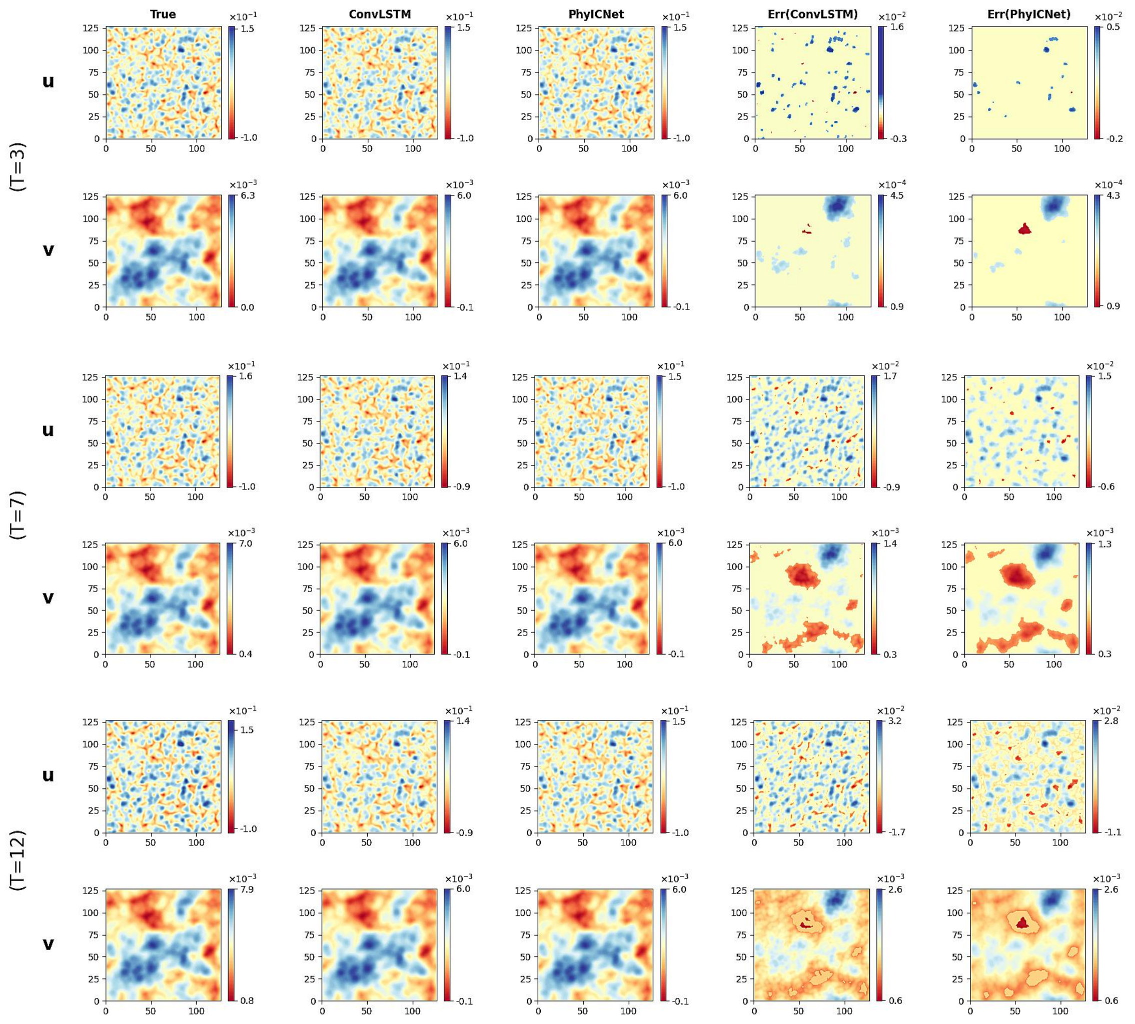


Figure 2. Comparison of long-term prediction performance. PhyICNet and baseline model predicted images and errors at randomly selected time T = 3, 7, 12 seconds.

4.3. Network sensitivity analysis in the solution of the NS equations

In this part, taking the case of the NS equation, we will study the influence of discrete-time steps(dt) on the solution accuracy of PhyICNet and compare it with the errors of PINN and ConvLSTM. The NS equation is the basic equation to describe the motion of viscous fluid, being widely used in fluid mechanics, meteorology, and other fields. We consider the $2D$ NS equation as follows:

$$\rho(\delta_t v + (v \cdot \nabla)v) - \lambda \bigtriangleup v + \nabla p = f. \tag{4.2}$$

where $v \in u, v$ is the velocity vector, p represents pressure, ρ stands for density, λ indicates viscosity, and the Reynolds number is 3900. We choose the computational domain Ω in the range of [0,64], the time interval T is set to [0,12], and the discretization is carried out with space spacing $dx = 1/10$. At the same time, we use different time step dt to obtain the error results of the model. In the process of model training, we uniformly set the learning rate to 1×10^{-4} and introduce the learning rate reduction strategy, that is, when the training process advances 100 epochs, the learning rate will decrease by a coefficient of 0.97.

To simulate different coarse and fine grids, we set up several different dt. Larger dt means a coarser network because the fluid state changes greatly per unit of time. Three networks are used to solve the NS equations under different dt. For each dt, the error between the true solution and the predicted solution of u and v at the last T time is calculated, and the MSE is used as the error measure. By comparing the errors of the network model under different dt, the sensitivity of the coarse and fine grids is analyzed.

Table 1 shows the prediction MSE error comparison of the three models under different dt. The experimental results show that with the increase of dt, the error of PhyICNet is always better than that of the other two networks, indicating that PhyICNet can still maintain good prediction accuracy under coarse grid, and the error growth is slow.

Table 1. The prediction error of each network for u and v under different dt.

dt	0.0002	0.0009	0.005	0.02
PINN(u)	$5.36e-03$	$5.62e-03$	$6.65e-02$	$1.06e-01$
ConvLSTM(u)	$5.32e-03$	$5.68e-03$	$4.24e-02$	$5.09e-02$
PhyICNet(u)	$5.31e-03$	$5.33e-03$	$2.99e-02$	$3.14e-02$
PINN(v)	$1.6966e-02$	$1.6972e-02$	$2.43e-02$	$9.91e-02$
ConvLSTM(v)	$1.6972e-02$	$1.6984e-02$	$1.77e-02$	$4.33e-02$
PhyICNet(v)	$1.6963e-02$	$1.6966e-02$	$1.71e-02$	$1.78e-02$

In summary, through a comprehensive analysis of the grid sensitivity in the solution of the NS equation, we verify the error changes of the PINN, ConvLSTM, and PhyICNet models for the solution of the NS equation under different dt. This result has important guiding significance for dealing with scenes with different grid scales in practical applications, and proves the superiority of PhyICNet in dealing with complex fluid dynamics problems.

4.4. The effect of noise level on the performance of two networks

In this case, to study the influence of noise level on the performance of the PhyICNet in solving spatiotemporal-related nonlinear PDEs, we will take NLS equation examples. The NLS equation has important applications in quantum mechanics, optics, and hydrodynamics, especially in describing the propagation and evolution of waves. It is expressed as follows:

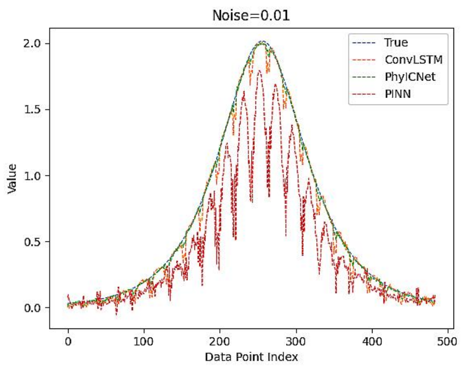
$$iu_t + u_{xx} + 2|u|^2u = 0. \tag{4.3}$$

where u represents the complex number wave, and the imaginary unit i describes the oscillation of the time evolution of the wave function. The computational domain of the spatial domain $\Omega \in [0, 1/2]$ and

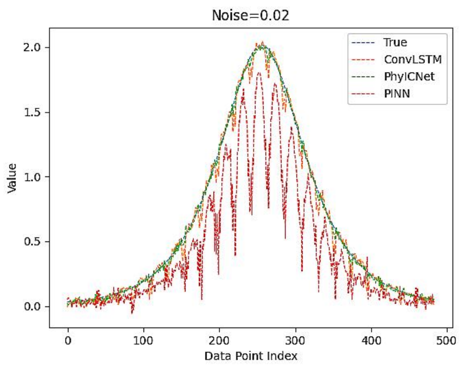
$T \in [0, 4]$ is discretized by the interval of $dx = 1/256$ and $dt = 0.005$. The learning rate is also set to 1×10^{-4}, and the decay rate per 100 epochs is 0.97.

In our experiment, in order to simulate the uncertainty and measurement error of the data in the real environment, we introduce different levels of Gaussian noise into the solution process of the NLS equation to evaluate the robustness and accuracy of the three models against noise interference.

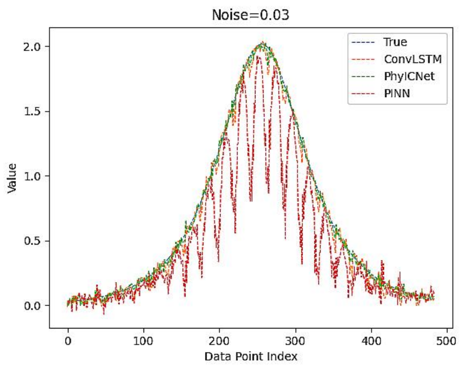
We use PINN, ConvLSTM, and PhyICNet to solve the NLS equation under different noise levels, and the curves of the true solution and the predicted solution of the three models at the last T moment under different noise levels are drawn. As shown in Figure 3, with the gradual increase of noise intensity, PhyICNet is always close to the real data, indicating that it has better performance stability in noise environment.



(a) Noise = 0.01



(b) Noise = 0.02



(c) Noise = 0.03

Figure 3. The influence of noise level on the three networks.

Through experimental design and data analysis, we confirm that PhyICNet shows higher robustness and accuracy than the other two networks in solving noisy NLS equations. This finding has important significance for applying deep learning models to solve PDEs in complex and uncertain environments.

4.5. The influence of interaction times on the learning effect of PhyICNet

In order to explore the influence of the number of interactions on the learning effect of PhyICNet and determine the optimal number of interactions to improve the long-distance learning ability of the model, we compared the performance of PINN, ConvLSTM, and PhyICNer under different interaction times (r). The MSE loss function is used as the main index to evaluate the effect of network learning, and the performance is evaluated after 2000 iterations.

As a typical nonlinear PDE, the Burgers'equation has important application value in simulating fluid dynamics and other physical phenomena. We consider a $2D$ hydrodynamic problem, namely the classical $2D$ Burgers'equation, which is expressed as follows:

$$\begin{aligned} u_t + uu_x + vu_y - 1/R\Delta u = 0, \\ v_t + uv_x + vv_y - 1/R\Delta v = 0. \end{aligned} \tag{4.4}$$

where $\{u, v\}$ represents the fluid velocity, $1/R$ denotes the viscous coefficient, and Δ is the Laplace operator. In our test, $R = 200$ is set and performed in the computational domain of the spatial domain $\Omega \in [0, 1]$ and $T \in [0, 12]$. The spatial interval is $d_x = 1/128$, the time interval is $d_t = 0.002$, the learning rate is 1×10^{-4}, and the decay rate per 100 epochs is 0.97.

We set the number of interactive learning as 3, 4, and 5 times, that is, $r = 3, 4, 5$. In each group of experiments, the model was trained by 2000 iterations and the loss value after each iteration was recorded. For each set of experiments, the same data set is used to train the model to ensure the consistency of training data. During the training process, we monitor the change of the loss function and record the loss value of each iteration step.

After 2000 iterations, we compare the loss curves obtained by different networks, and give the logarithmic loss value in Figure 4, and obtain some useful results:

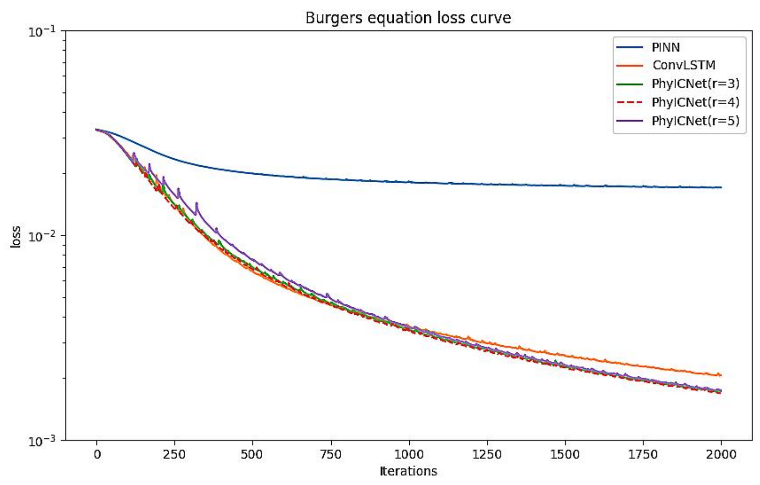


Figure 4. Comparing the difference of the error evolution of different networks over time.

For PhyICNet with $r = 3$, compared with ConvLSTM and PINN, its distance learning ability is improved. For PhyICNet with $r = 4$, compared with PhyICNet with $r = 3$, the loss value of the network is decreased, indicating that the network learning effect is better and its distance learning ability is further improved. For PhyICNet with $r = 5$, compared with $r = 4$, the effect is decreased. Compared with ConvLSTM, although the loss value is decreased, the decrease is relatively small. We also tested other equations and got the same results. In addition, too much interactive learning may lead to overfitting of the model and affect the generalization ability.

In summary, for PhyICNet, in the case of 2000 iterations, the network learning effect is better when $r = 4$. Therefore, in future experiments, we can set the number of interactive learning to 4 to further improve long-distance learning ability. This finding has important guiding significance for optimizing the network structure and improving the performance of the model.

5. Conclusions

In this paper, we proposed a new deep learning architecture, called PhyICNet, for the modeling and application of nonlinear spatiotemporal dynamics systems. The advantage of PhyICNet is that interactive learning can be performed to improve the accuracy of the solution and the ability of long-distance learning. Through several numerical experiments, we verified the superiority of PhyICNet, showing that it can achieve the best results with an interactive learning of 4 times. We emphasize that our method is superior to other networks in some aspects, so it also provides additional reference schemes for its wide application in science and engineering.

Use of AI tools declaration

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

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No.11601411), the Natural Science Basic Research Plan in Shaanxi Province of China (Program No.2023-JC-YB-063), the Natural Science Basic Research Plan in Shaanxi Province of China (Program No. 2024JC-YBQN-0039).

Conflict of interest

The authors declare there is no conflicts of interest.

References

1. T. J. R. Hughes, The finite element method. Linear static and dynamic finite element analysis, *Comput. Methods Appl. Mech. Eng.*, **65** (1987), 191–193. https://doi.org/10.1016/0045-7825(87)90013-2
2. M. W. M. G. Dissanayake, N. Phan-Thien, Neural-network-based approximations for solving partial differential equations, *Commun. Numer. Methods Eng.*, **10** (1994), 195–201. https://doi.org/10.1002/cnm.1640100303
3. I. E. Lagaris, A. Likas, D. I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, *IEEE Trans. Neural Networks*, **9** (1998), 987–1000. https://doi.org/10.1109/72.712178

4. T. J. R. Hughes, J. A. Cottrell, Y. Bazilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, *Comput. Methods Appl. Mech. Eng.*, **194** (2005), 4135–4195. https://doi.org/10.1016/j.cma.2004.10.008

5. J. Han, A. Jentzen, W. E, Solving high-dimensional partial differential equations using deep learning, *PNAS*, **115** (2018), 8505–8510. https://doi.org/10.1073/pnas.1718942115

6. M. Raissi, P. Perdikaris, G. E. Karniadakis, Numerical gaussian processes for time-dependent and nonlinear partial differential equations, *SIAM J. Sci. Comput.*, **40** (2018), A172–A198. https://doi.org/10.1137/17M1120762

7. M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, *J. Comput. Phys.*, **378** (2019), 686–707. https://doi.org/10.1016/j.jcp.2018.10.045

8. C. Song, T. Alkhalifah, U. B. Waheed, Solving the frequency-domain acoustic VTI wave equation using physics-informed neural networks, *Geophys. J. Int.*, **225** (2020), 846–859. https://doi.org/10.1093/gji/ggab010

9. F. Sahli Costabal, Y. Yang, P. Perdikaris, D. E. Hurtado, E. Kuhl, Physics-informed neural networks for cardiac activation mapping, *Front. Phys.*, **8** (2020), 42. https://doi.org/10.3389/fphy.2020.00042

10. J. D. Willard, X. Jia, S. Xu, M. S. Steinbach, V. Kumar, Integrating physics-based modeling with machine learning: A Survey, preprint, arXiv:2003.04919, 2020. https://doi.org/10.48550/arXiv.2003.04919

11. G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, *Nat. Rev. Phys.*, **3** (2021), 422–440. https://doi.org/10.1038/s42254-021-00314-5

12. M. Rashtbehesht, C. Huber, K. Shukla, G. Karniadakis, Physics-informed deep learning for wave propagation and full waveform inversions, *J. Geophys. Res.: Solid Earth*, **2021** (2021). https://doi.org/10.1029/2021JB023120

13. S. Liu, B. B. Kappes, B. Amin-ahmadi, O. Benafan, X. Zhang, A. P. Stebner, Physics-informed machine learning for composition - process - property design: Shape memory alloy demonstration, *Appl. Mater. Today*, **22** (2021), 100898. https://doi.org/10.1016/j.apmt.2020.100898

14. M. Raissi, G. E. Karniadakis, Hidden physics models: Machine learning of non-linear partial differential equations, *J. Comput. Phys.*, **357** (2018), 125–141. https://doi.org/10.1016/j.jcp.2017.11.039

15. E. Samaniego, C. Anitescu, S. Goswami, V. M. Nguyen-Thanh, H. Guo, K. Hamdia, et al., An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementation and applications, *Comput. Methods Appl. Mech. Eng.*, **362** (2020), 112790. https://doi.org/10.1016/j.cma.2019.112790

16. C. Rao, H. Sun, Y. Liu, Physics-informed deep learning for computational elastodynamics without labeled data, *J. Eng. Mech.*, **147** (2021), 04021043. https://doi.org/10.1061/(ASCE)EM.1943-7889.0001947

17. B. Wu, O. Hennigh, J. Kautz, S. Choudhry, W. Byeon, Physics informed RNN-DCT networks for time-dependent partial differential equations, preprint, arXiv:2202.12358, 2022. https://doi.org/10.48550/arXiv.2202.12358

18. A. Sherstinsky, Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network, *Physica D*, **404** (2020), 132306. https://doi.org/10.1016/j.physd.2019.132306

19. R. Schmidt, Recurrent neural networks (RNNs): A gentle introduction and overview, preprint, arXiv:1912.05911, 2019. https://doi.org/10.48550/arXiv.1912.05911

20. Y. Kim, Convolutional neural networks for sentence classification, in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 2014. https://doi.org/10.3115/v1/D14-1181

21. Y. Zhu, N. Zabaras, Bayesian deep convolutional encoder-decoder networks for surrogate modeling and uncertainty quantification, *J. Comput. Phys.*, **366** (2018), 243–266. https://doi.org/10.1016/j.jcp.2018.04.018

22. C. Rao, H. Sun, Y. Liu, Hard encoding of physics for learning spatiotemporal dynamics, preprint, arXiv:2105.00557, 2021. https://doi.org/10.48550/arXiv.2105.00557

23. Y. Zhu, N. Zabaras, P. S. Koutsourelakis, P. Perdikaris, Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data, *J. Comput. Phys.*, **394** (2019), 56–81. https://doi.org/10.1016/j.jcp.2019.05.024

24. L. Sun, H. Gao, S. Pan, J. Wang, Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data, *Comput. Methods Appl. Mech. Eng.*, **351** (2019), 112732. https://doi.org/10.1016/j.cma.2019.112732

25. H. Gao, L. Sun, J. X. Wang, PhyGeoNet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain, *J. Comput. Phys.*, **428** (2020), 110079. https://doi.org/10.1016/j.jcp.2020.110079

26. P. Ren, C. Rao, Y. Liu, Z. Ma, Q. Wang, J. X. Wang, et al., PhySR: Physics-informed deep super-resolution for spatiotemporal data, *J. Comput. Phys.*, **492** (2023), 112438. https://doi.org/10.1016/j.jcp.2023.112438

27. A. D. Jagtap, G. E. Karniadakis, Extended physics-informed neural networks (XPINNs): A generalized space-time domain decomposition based deep learning framework for non-linear partial differential equations, *Commun. Comput. Phys.*, **28** (2020), 2002–2041. https://doi.org/10.4208/cicp.OA-2020-0164

28. P. Ren, N. Erichson, S. Subramanian, O. San, Z. Lukic, M. Mahoney, SuperBench: A super-resolution benchmark dataset for scientific machine learning, preprint, arXiv:2306.14070, 2023. https://doi.org/10.48550/arXiv.2306.14070

29. L. Wang, Z. Zhou, Z. Yan, Data-driven vortex solitons and parameter discovery of 2D generalized nonlinear Schrödinger equations with a PT-symmetric optical lattice, *Comput. Math. Appl.*, **140** (2023), 17–23. https://doi.org/10.1016/j.camwa.2023.03.015

30. M. Sadr, T. Tohme, K. Youcef-Toumi, Data-driven discovery of PDEs via the adjoint method, preprint, arXiv:2401.17177, 2024. https://doi.org/10.48550/arXiv.2401.17177

31. F. J. Aguilar-Canto, C. Brito-Loeza, H. Calvo, Model discovery of compartmental models with graph-supported neural networks, *Appl. Math. Comput.*, **464** (2024), 128392. https://doi.org/10.1016/j.amc.2023.128392

32. C. Rao, P. Ren, Y. Liu, H. Sun, Discovering nonlinear PDEs from scarce data with physics-encoded learning, preprint, arXiv:2201.12354, 2022. https://doi.org/10.48550/arXiv.2201.12354

33. Y. Hu, T. Zhao, S. Xu, L. Lin, Z. Xu, Neural-PDE: a RNN based neural network for solving time dependent PDEs, *Commun. Inf. Syst.*, **22** (2020), 223–245. https://doi.org/10.4310/CIS.2022.v22.n2.a3

34. P. Ren, C. Rao, Y. Liu, J. X. Wang, H. Sun, PhyCRNet: Physics-informed convolutional-recurrent network for solving spatiotemporal PDEs, *Comput. Methods Appl. Mech. Eng.*, **389** (2022), 114399. https://doi.org/10.1016/j.cma.2021.114399

35. L. Jiang, L. Wang, X. Chu, Y. Xiao, H. Zhang, PhyGNNet: Solving spatiotemporal PDEs with physics-informed graph neural network, in *Proceedings of the 2023 2nd Asia Conference on Algorithms, Computing and Machine Learning*, 2022. https://doi.org/10.1145/3590003.3590029

36. X. Meng, Z. Li, D. Zhang, G. E. Karniadakis, PPINN: Parareal physics-informed neural network for time-dependent PDEs, *Comput. Methods Appl. Mech. Eng.*, **370** (2020), 113250. https://doi.org/10.1016/j.cma.2020.113250

37. A. Mavi, A. C. Bekar, E. Haghighat, E. Madenci, An unsupervised latent/output physics-informed convolutional-LSTM network for solving partial differential equations using peridynamic differential operator, *Comput. Methods Appl. Mech. Eng.*, **407** (2023), 115944. https://doi.org/10.1016/j.cma.2023.115944

38. P. Saha, S. Dash, S. Mukhopadhyay, Physics-incorporated convolutional recurrent neural networks for source identification and forecasting of dynamical systems, *Neural Networks*, **144** (2021), 359–371. https://doi.org/10.1016/j.neunet.2021.08.033

39. P. R. Kakka, Sequence to sequence AE-ConvLSTM network for modelling the dynamics of PDE systems, preprint, arXiv:2208.07315, 2022. https://doi.org/10.48550/arXiv.2208.07315

40. B. Krause, L. Lu, I. Murray, S. Renals, Multiplicative LSTM for sequence modelling, preprint, arXiv:1609.07959, 2017. https://doi.org/10.48550/arXiv.1609.07959

41. G. Melis, T. Kočiský, P. Blunsom, Mogrifier LSTM, preprint, arXiv:1909.01792, 2020. https://doi.org/10.48550/arXiv.1909.01792

42. G. Larsson, M. Maire, G. Shakhnarovich, FractalNet: Ultra-deep neural networks without residuals, preprint, arXiv:1605.07648, 2017. https://doi.org/10.48550/arXiv.1605.07648

43. Y. Lu, A. Zhong, Q. Li, B. Dong, Beyond finite layer neural networks: bridging deep architectures and numerical differential equations, preprint, arXiv:1710.10121, 2020. https://doi.org/10.48550/arXiv.1710.10121

44. L. Ruthotto, E. Haber, Deep neural networks motivated by partial differential equations, *J. Math. Imaging Vision*, **61** (2019), 787–805. https://doi.org/10.1007/s10851-019-00903-1

45. Z. Long, Y. Lu, B. Dong, PDE-Net 2.0: Learning PDEs from data with a numeric-symbolic hybrid deep network, *J. Comput. Phys.*, **399** (2019), 108925. https://doi.org/10.1016/j.jcp.2019.108925

46. M. Raissi, Deep hidden physics models: deep learning of nonlinear partial differential equations, *J. Mach. Learn. Res.*, **19** (2018), 932–955. https://doi.org/10.5555/3291125.3291150

47. D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, preprint, arXiv:1412.6980, 2017. https://doi.org/10.48550/arXiv.1412.6980

48. A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, et al., Automatic differentiation in PyTorch, in *NeurIPS Autodiff Workshop*, 2017.