Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 389 (2022) 114399
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

PhyCRNet: Physics-informed convolutional-recurrent network for solving spatiotemporal PDEs

Pu Ren[a], Chengping Rao[b], Yang Liu[b,*], Jian-Xun Wang[c], Hao Sun[d,e,a,f]

[a] *Department of Civil and Environmental Engineering, Northeastern University, Boston, MA 02115, USA*
[b] *Department of Mechanical and Industrial Engineering, Northeastern University, Boston, MA 02115, USA*
[c] *Department of Aerospace and Mechanical Engineering, University of Notre Dame, Notre Dame, IN 46556, USA*
[d] *Gaoling School of Artificial Intelligence, Renmin University of China, Beijing, 100872, China*
[e] *Beijing Key Laboratory of Big Data Management and Analysis Methods, Beijing, 100872, China*
[f] *Department of Civil and Environmental Engineering, MIT, Cambridge, MA 02139, USA*

Received 26 June 2021; received in revised form 2 October 2021; accepted 23 November 2021
Available online 18 December 2021

Dataset link: https://github.com/isds-neu/PhyCRNet

Abstract

Partial differential equations (PDEs) play a fundamental role in modeling and simulating problems across a wide range of disciplines. Recent advances in deep learning have shown the great potential of physics-informed neural networks (PINNs) to solve PDEs as a basis for data-driven modeling and inverse analysis. However, the majority of existing PINN methods, based on fully-connected NNs, pose intrinsic limitations to low-dimensional spatiotemporal parameterizations. Moreover, since the initial/boundary conditions (I/BCs) are softly imposed via penalty, the solution quality heavily relies on hyperparameter tuning. To this end, we propose the novel physics-informed convolutional-recurrent learning architectures (PhyCRNet and PhyCRNet-s) for solving PDEs without any labeled data. Specifically, an encoder–decoder convolutional long short-term memory network is proposed for low-dimensional spatial feature extraction and temporal evolution learning. The loss function is defined as the aggregated discretized PDE residuals, while the I/BCs are hard-encoded in the network to ensure forcible satisfaction (e.g., periodic boundary padding). The networks are further enhanced by autoregressive and residual connections that explicitly simulate time marching. The performance of our proposed methods has been assessed by solving three nonlinear PDEs (e.g., 2D Burgers' equations, the λ-ω and FitzHugh Nagumo reaction–diffusion equations), and compared against the start-of-the-art baseline algorithms. The numerical results demonstrate the superiority of our proposed methodology in the context of solution accuracy, extrapolability and generalizability.

Keywords: Convolutional-recurrent learning; Partial differential equations; Encoder–decoder; Physics-informed deep learning; Residual connection; Hard-encoding of I/BCs

* Corresponding author.
E-mail addresses: ren.pu@northeastern.edu (P. Ren), yang1.liu@northeastern.edu (Y. Liu), haosun@ruc.edu.cn (H. Sun).

https://doi.org/10.1016/j.cma.2021.114399
0045-7825/

1. Introduction

Complex spatiotemporal systems, modeled by PDEs, are ubiquitous in many disciplines such as applied mathematics, physics, biology, chemistry and engineering. Solving PDE systems has been a critical component in the community of scientific computing. Since the analytical solutions are inaccessible to most of the physical systems, numerical approaches have been extensively investigated and developed in recent decades, such as the finite difference/element/volume methods [1] and isogeometric analysis (IGA) [2]. Although the classical numerical methods that approximate the exact solutions with basis functions and unknown parameters can achieve great accuracy for forward analysis, the computational demand remains a critical issue in applications of data assimilation and inverse problems, e.g., due to the requirement of repeated forward simulations. In the meanwhile, inevitable modeling errors and uncertainties are hard to adjust/mitigate in these intractable problems.

Alternatively, recent developments in deep learning shed new lights on surrogate modeling of nonlinear systems for solving forward and inverse problems. The surrogate models leverage the powerful universal approximation capacity of deep neural networks (DNNs) [3] which avoid repeated forward analyses and provide a promising direction for data assimilation and inverse problems. Herein, the core and fundamental challenge for NN-based approaches lies in how to effectively solve PDEs. Actually, dated back to last century, Lagaris et al. [4,5] have already observed the similarities between spline-based numerical simulations and NN-based solutions for differential equations, and proposed the pioneering work on NN as the basic approximation element where the parameters are learnable by minimizing the physics-inspired error function. Over the past decade, thanks to the great advances in deep learning, many attempts have been devoted to this resurgent topic, which have led to a proliferation of studies in scientific machine learning [6–15] with growing attention on modeling and simulation of PDE systems. Latest studies utilizing DNNs for modeling physical systems fall into two streams: continuous and discretized networks. The representative work of continuous learning is physics-informed neural networks (PINNs) [9], introduced for forward and inverse analysis of PDEs based on fully-connected DNNs. The general principle of PINNs inherits and matures the previous NN-based scheme [4] with loss function consisting of PDE residuals, which facilitates DNNs training in "small data" stage (e.g., small [16–22] or zero labeled datasets [23,24]). PINNs not only foster the current success in simulating various PDE systems (e.g., fluid dynamics [25–28], solid mechanics [24,29] and stochastic PDEs [30,31]), but also show remarkable applications to a broad spectrum of other disciplines, including blood flows modeling [32,33], non-invasive inference [34–36], metamodeling of nonlinear structures [37,38], denoising [39], PDE discovery [40] and many others. Despite the great success and promise, there exists one central issue of scalability in the existing PINN framework: it is generally limited to low-dimensional parameterizations and less capable of handling PDE systems whose behavior has sharp gradients (e.g., propagating fronts of waves) or complex local morphology. The requirement of vast collocation points for PDE residuals may lead to huge computational cost and cause slow convergence in training [41]. In addition, data-driven neural operators [42–45] also seek for the nonlinear mapping from parametric DNNs to the numerical solution in a continuous setting. The infinite-dimensional operator learning methods prevail over the tyranny of meshes/grids, but require a moderate amount of high-quality training data and tend to be computational intensively.

A few very recent pilot studies show that physics-informed discrete learning schemes, e.g., convolutional neural networks (CNNs), possess better scalability and faster convergence [12,41,46–49] for modeling PDE systems, thanks to their light-weight architecture and strength of efficient filtering over the computational domain. For the time-independent systems (e.g., steady-state PDEs), Zhu et al. [12,50] applied CNNs for surrogate modeling and uncertainty quantification (UQ) of PDE systems in rectangular reference domain. Furthermore, PhyGeoNet [41] was proposed for geometry-adaptively solving steady-state PDEs via coordinate transformation between the physical and reference domains. On the other hand, for the time-dependent systems, the majority of the NN-based solutions still focus on data-driven approaches in the regular/rectangle grid [51–54] or the irregular mesh [55–59]. Very few research (e.g., the AR-DenseED method in [47]) explores the possibility of using discrete learning to solve PDEs without any labeled data. Although the existing effort escapes the demanding requirement of high-quality training data, it has not shown satisfactory performance in error propagation [47], due to the limitation of the basic autoregressive (AR) process. In general, relevant studies on scalable discretized learning architectures for solving spatiotemporal PDEs in "small data" regime remain limited in literature.

The specific objective of this paper is to propose a novel physics-informed convolutional-recurrent learning architecture (PhyCRNet) and its light-weight variant (PhyCRNet-s) for solving multi-dimensional spatiotemporal PDEs without any labeled data. We do not attempt to compare our proposed methods with classical numerical

solvers, but instead provide a spatiotemporal deep learning perspective for surrogate modeling of complex PDEs, which can further serve as a basis approach for tackling challenges in data-enabled scientific computation such as inverse problems and data assimilation especially in sparse and noisy data regimes. The contributions of our paper are summarized as follows. First of all, the innovative PhyCRNet architecture combines the strengths of (1) an encoder–decoder convolutional long short-term memory network (ConvLSTM) [60] that extracts low-dimensional spatial features and learns their temporal evolution, (2) a global residual-connection that stringently maps the time-marching dynamics of the PDE solution, and (3) high-order finite-difference-based spatiotemporal filtering that accurately determines the essential PDE derivatives for constructing the residual loss function. Based on the fundamental neural components of PhyCRNet, we also propose PhyCRNet-s which periodically skips the encoder part in order to improve the computational efficiency. Secondly, hard-encoding of initial/boundary conditions (I/BCs) into the networks is implemented. The hard-imposed physical constraints drastically promote the solution accuracy on the boundaries. Finally, the numerical experiments ranging from nonlinear fluid dynamics to reaction–diffusion (RD) systems are performed to validate the proposed approaches. The numerical results show the superiority of PhyCRNet/PhyCRNet-s in the context of solution accuracy, extrapolability and generalizability in comparison with two baseline models.

The rest of the paper is organized as follows, in addition to this Introduction section. Section 2 sets up the problem of solving PDE systems using DNNs. In Section 3, we elaborate the general principle and network architectures of PhyCRNet and PhyCRNet-s. In Section 4, we present the extensive numerical experiments and compare the performance between our networks and baseline methods. Section 5 discusses the observations as well as the outlook of our future study. Section 6 concludes the entire paper.

2. Problem statement

Herein, we consider the general form of a set of multi-dimensional (n), nonlinear, coupled PDE systems in parametric setting:

$$\mathbf{u}_t + \mathcal{F}[\mathbf{u}, \mathbf{u}^2, \ldots, \nabla_{\mathbf{x}}\mathbf{u}, \nabla^2_{\mathbf{x}}\mathbf{u}, \nabla_{\mathbf{x}}\mathbf{u}\cdot\mathbf{u}, \ldots; \boldsymbol{\lambda}] = \mathbf{0}, \tag{1}$$

where $\mathbf{u}(\mathbf{x}, t) \in \mathbb{R}^n$ denotes the solution variable in the temporal domain $t \in [0, T]$ and the physical domain Ω; $\mathbf{u}_t$ is the first-order time derivative term; ∇ represents the gradient operator with respect to $\mathbf{x}$; $\mathcal{F}[\cdot]$ is the nonlinear functional parameterized by $\boldsymbol{\lambda}$. Additionally, the I/BCs have the form as $\mathcal{I}[\mathbf{u}, \mathbf{u}_t; t = 0, \mathbf{x} \in \Omega] = 0$ and $\mathcal{B}[\mathbf{u}, \nabla_{\mathbf{x}}\mathbf{u}, \ldots; \mathbf{x} \in \partial\Omega] = 0$, where $\partial\Omega$ denotes the boundary of the spatial domain.

Our general goal is to develop DNN-based methods for forward analysis of spatiotemporal PDE systems given in Eq. (1), which could serve as a basis for inverse problems when data is available (e.g., uncertainty quantification and data assimilation). More precisely, such networks will act as a new class of numerical solvers for various time-dependent PDEs given specific I/BCs. The entire training stage is unsupervised, where we do not require any labeled data and merely utilize the physical laws (e.g., PDEs and I/BCs) as constraints. Besides, to portray better local details in the solution, we discretize the physical domain and solve PDEs by employing (1) convolutional operators due to its faster convergence and better accuracy compared with fully-connected neural networks according to previous studies [41,47,50] and (2) recurrent units for controlling error propagation. In this work, we mainly focus on regular (e.g., rectangular) physical domains, where both the spatial and temporal domains are discretized uniformly and convolutional filtering can be applied in nature. Our objective is to point-wisely approximate the discrete solution field $\mathbf{u}^\theta = \mathbf{u}(\mathbf{x}, t; \boldsymbol{\theta})$, that satisfies Eq. (1) for specifically given I/BCs, where $\boldsymbol{\theta}$ denotes the network trainable parameters. Furthermore, we view the network training as an optimization process by minimizing the loss function composed of the discrete PDE residuals subjected to I/BCs. The details are described in Section 3.

3. Methodology

In this section, two network architectures are proposed for solving spatiotemporal PDE systems. Based on the solid mathematical foundation in sparse representation for PDEs [61], we attempt to extract the low-dimensional spatial features from dynamics and learn the temporal evolution on the compressed information through an encoder–decoder convolutional-recurrent scheme. The light-weight networks are built with the input of previous state variable and the output of quantities of interest (next state variable). Previously, Geneva and Zabaras [47] claimed that recurrent neural networks (e.g., long short-term memory) are powerful tools for predicting time-series whereas trapped in the difficulty of training for solving PDEs. However, we show that, based on our network setting, such an issue can be mitigated.

3.1. ConvLSTM

ConvLSTM is a spatiotemporal sequence-to-sequence learning framework extended from long short-term memory (LSTM) [62] and its variant LSTM encoder–decoder forecasting architecture [63,64], which hold the strength of modeling the long-period dependencies that evolve in time. This great success comes from the distinctive and innovative memory cell and gated scheme of LSTM. Essentially, the memory cell is updated through the input and state information being accessed, accumulated and removed due to the delicate design of controlling gates. Based on such setup, the notorious gradient vanishing problem of vanilla recurrent neural networks (RNNs) has been relieved. It is also worthwhile to mention that a typical RNN model can be seen as a variant to a state–space model with nonlinear activation functions incorporated [65,66]. Herein, LSTM, a special class of RNNs, acts as an implicit numerical scheme for solving time-dependent differential equations.

The fundamental of ConvLSTM is to inherit the basic construction of LSTM (i.e., cells and gates) for controlling the information flow, and to modify the fully-connected NNs (FC-NNs) in gated operations to CNNs considering their better representational capability of spatial connections [60]. The graphic demonstration is presented in Fig. 1(a). Let $\mathbf{X}_t$ denote the input tensor, and $\{\mathbf{h}_t, \mathbf{C}_t\}$ the hidden state and cell state to be updated at time t respectively. Moreover, the ConvLSTM cell consists of four gate variables for input-to-state and state-to-state transitions, including a forget gate $\mathbf{f}_t$, an input gate $\mathbf{i}_t$, an internal cell $\widetilde{\mathbf{C}}_t$ and an output gate $\mathbf{o}_t$. In specific, due to the sigmoid activation function $\sigma(\cdot)$ mapping outputs to values between 0 and 1, the forget gate layer adaptively clears the memory information in the cell state $\mathbf{C}_{t-1}$. The memory stored in cell state originates from the cooperation between the input gate layer and the internal cell state, where the internal cell state is a new cell candidate created from hyperbolic tangent activation layer (i.e., $\tanh(\cdot)$) and the input gate layer decides the information propagating into the cell state. Lastly, the output gate layer filters and regulates the cell state for the final output variable/hidden state. The mathematical formulations of updating ConvLSTM cells are expressed as:

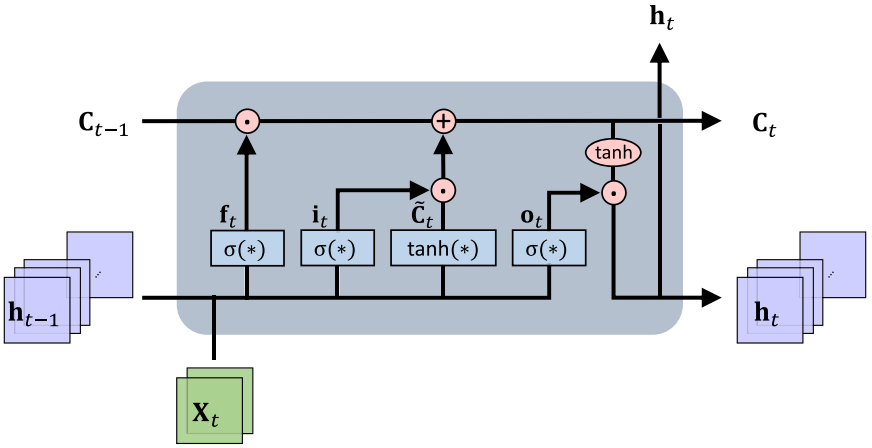
$$\begin{aligned} \mathbf{i}_t &= \sigma(\mathbf{W}_i * [\mathbf{X}_t, \mathbf{h}_{t-1}] + \mathbf{b}_i), & \mathbf{f}_t &= \sigma(\mathbf{W}_f * [\mathbf{X}_t, \mathbf{h}_{t-1}] + \mathbf{b}_f), \\ \widetilde{\mathbf{C}}_{t-1} &= \tanh(\mathbf{W}_c * [\mathbf{X}_t, \mathbf{h}_{t-1}] + \mathbf{b}_c), & \mathbf{C}_t &= \mathbf{f}_t \odot \mathbf{C}_{t-1} + \mathbf{i}_t \odot \widetilde{\mathbf{C}}_{t-1}, \\ \mathbf{o}_t &= \sigma(\mathbf{W}_o * [\mathbf{X}_t, \mathbf{h}_{t-1}] + \mathbf{b}_o), & \mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{C}_t), \end{aligned} \tag{2}$$

where $*$ is the convolutional operation and $\odot$ denotes the Hadamard product; $\{\mathbf{W}_i, \mathbf{W}_f, \mathbf{W}_c, \mathbf{W}_o\}$ are the weight parameters for the corresponding filters while $\{\mathbf{b}_i, \mathbf{b}_f, \mathbf{b}_c, \mathbf{b}_o\}$ represent bias vectors.

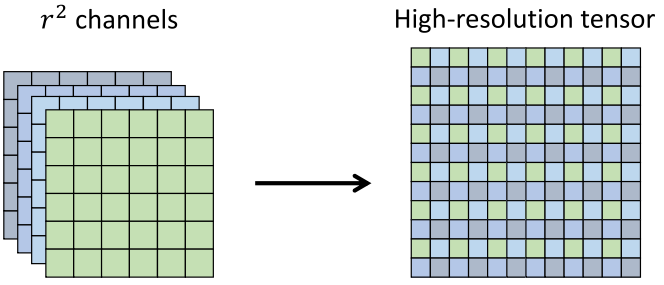
3.2. Pixel shuffle

Widely adopted in super-resolution tasks, pixel shuffle is an efficient operation to conduct the sub-pixel convolutions [67] in purpose of upscaling the low-resolution (LR) feature maps into the high-resolution (HR) outputs. The basic principle of pixel shuffle is described in Fig. 1(b). Let us consider a LR feature tensor of shape $(C \times r^2, H, W)$, where C denotes the number of channels, $\{H, W\}$ refer to the height and width respectively, and r is the upscaling factor. Straightforwardly, it realigns the elements in LR tensor to a HR tensor of shape $(C, H \times r, W \times r)$.

Through a simple and fast operation, pixel shuffle maintains the satisfactory reconstruction accuracy in image and video super-resolution tasks without high computational and memory cost. There are two tricks contributing to the efficiency. Firstly, we can implement the sub-pixel convolution in the last layer for the spatial upscaling. It has lower computational complexity compared with other classical upscaling methods (e.g., deconvolution [68]) which always need more layers to reach expected resolution. Secondly, before the upsampling layer, all the feature extraction layers are based on the LR tensors where smaller filter can be employed. Beyond that, another advantage of pixel shuffle is that it introduces fewer checkerboard artifacts compared with deconvolution [69]. Hence, we consider pixel shuffle operation as a preferable upsampling strategy in the network conception described in Section 3.3. Specifically, thanks to pixel shuffle for upscaling LR latent features to HR outputs, we can model the temporal evolution on LR dynamics by using ConvLSTM, which remarkably improves the computational efficiency and relieve the memory burden especially for large-scale tasks.



(a) The single ConvLSTM cell at time t.



(b) The pixel shuffle layer.

Fig. 1. The graphic illustration of network components.

3.3. Network architecture: PhyCRNet

In this part, we present the architecture of PhyCRNet, comprised of an encoder–decoder module, residual connection, autoregressive (AR) process and filtering-based differentiation. The framework is shown in Fig. 2. The encoder includes three convolutional layers for learning low-dimensional latent features from the input state variable $\mathbf{u}_i$ ($i = 0, 1, \ldots, T-1$), where T denotes the total number of time steps. We apply ReLU as the activation function for the convolutional layers. Then ConvLSTM layers act as the temporal propagator on the low-resolution latent features with the initial hidden/cell states starting at rest (e.g., $\mathbf{C}_0 = \mathbf{0}$ and $\mathbf{h}_0 = \mathbf{0}$). Modeling the essential dynamics on low-dimensional variables is capable of capturing the temporal dependency accurately and, meanwhile, helps alleviate the memory burden. Another strength of using LSTM comes from the hyperbolic tangent function for the output state, which holds smooth gradient curve and also pushes the values to be between -1 and 1. Thus, after establishing the convolutional-recurrent scheme in the center, we directly reconstruct the low-resolution latent space to the high-resolution quantities based on an upsampling operation. In particular, the sub-pixel convolution layer (i.e., pixel shuffle) is applied due to its preferable efficiency and reconstruction accuracy with fewer artifacts compared with deconvolution. In the end, we add another convolutional layer for scaling the bounded output back to the original magnitude of interest. There is no activation function behind this scaling layer. Besides, it is worthwhile to mention that we do not consider batch normalization [70] in PhyCRNet in view of the limited amount of input variables and its deficiency to super-resolution [71]. As a substitute, we train the network with weight normalization [72] for training acceleration and better convergence [71].

Inspired by the forward Euler scheme, we append a global residual connection between the input state variable $\mathbf{u}_i$ and the output variable $\mathbf{u}_{i+1}$. The learning process at time instant t_i is formulated as $\mathbf{u}_{i+1} = \mathbf{u}_i + \delta t \cdot \mathcal{NN}[\mathbf{u}_i; \boldsymbol{\theta}]$,

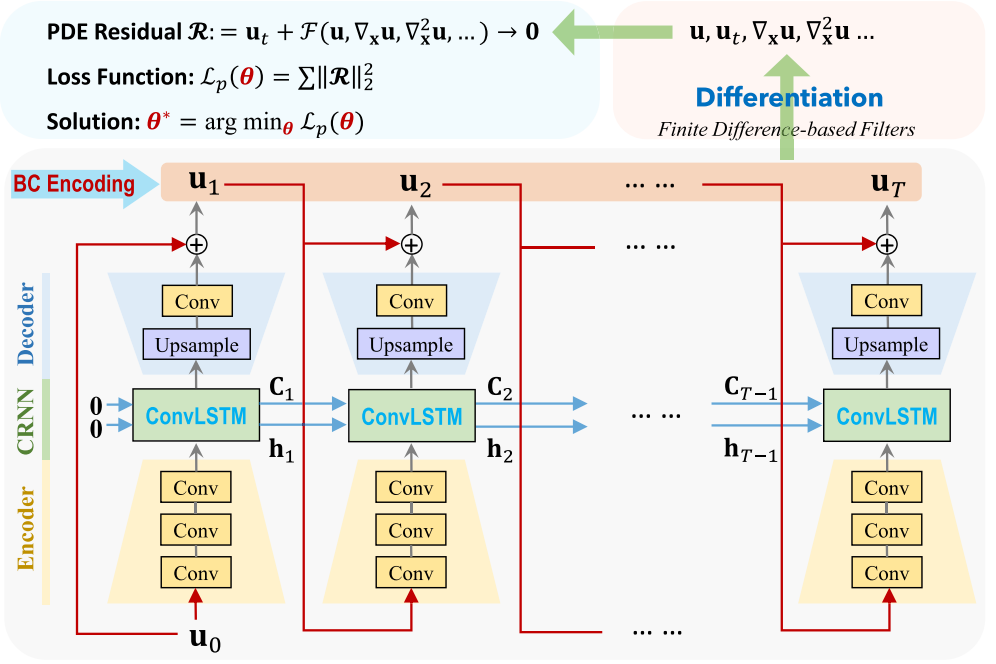


Fig. 2. The network architecture of PhyCRNet. The variables **C** and **h** are cell state and hidden state of ConvLSTM. $\mathbf{u}_0$ is the known state variable (i.e., IC). "BC Encoding" means hard enforcement of BCs on the learned output variables after each training epoch, which serves for differentiation. Besides, $\boldsymbol{\theta}$ denotes the unknown trainable parameters in PhyCRNet.

where $\mathcal{NN}[\cdot]$ denotes the trained network operator and δt is the time interval. The output state variable $\mathbf{u}_{i+1}$ at time instant t_i switches into the input variable at t_{i+1}. Actually, such input–output flow can be seen as a simple AR process (i.e., AR(1)).

Hence, not only do we have temporal evolution in the central latent representation, but also build the propagation on the input and output at each time instant. Moreover, the introduction of ConvLSTM can also help moderate the rigorous time-stepping issues, where larger time intervals may be adopted compared with traditional numerical methods (see Section 4). Here, $\mathbf{u}_0$ is the given IC, and $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_T$ are the discrete solution variables to be predicted. Next, the remaining challenge is how to calculate the derivative terms. We apply the gradient-free convolutional filters to represent the discrete numerical differentiation in order to approximate the derivative terms of interest [12,41]. For example, the finite-difference-based filters we considered in this paper are the second-order (see Eq. (3)) and fourth-order central difference schemes (see Eq. (4)) to compute temporal and spatial derivatives, respectively, given by

$$K_t = [-1, 0, 1] \times \frac{1}{2\delta t}, \tag{3}$$

$$K_s = \begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 16 & 0 & 0 \\ -1 & 16 & -60 & 16 & -1 \\ 0 & 0 & 16 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix} \times \frac{1}{12(\delta x)^2}. \tag{4}$$

where δt and δx denote the time spacing and spatial mesh size. Note that the derivatives on the boundaries cannot be computed directly due to the intrinsic deficiency of finite difference methods. The risk of losing differential information on the boundaries can be mitigated by designated padding mechanism (e.g., periodic padding) introduced in Section 3.5.

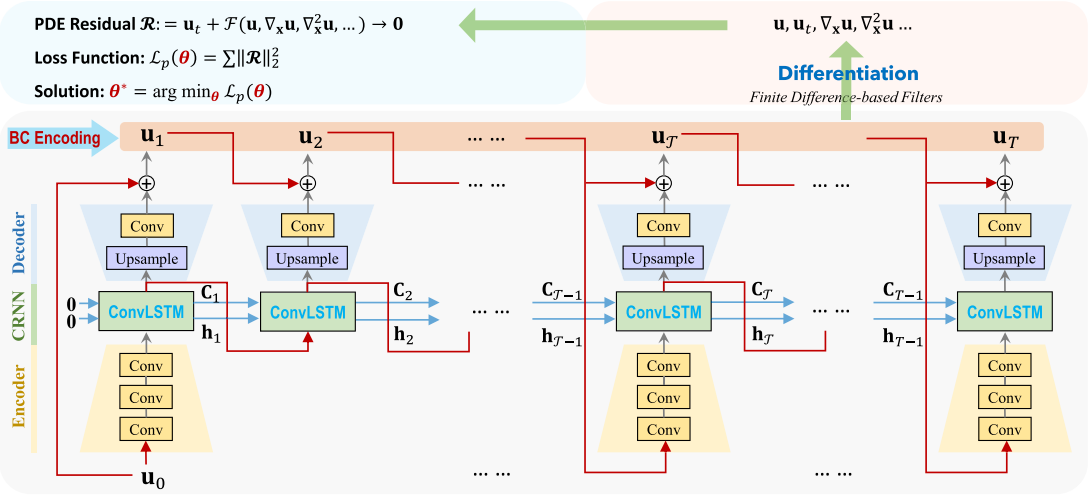


Fig. 3. The network architecture of PhyCRNet-s. $\mathcal{T}$ is the cycle index of skipping-encoder.

3.4. Efficient network architecture: PhyCRNet-s

To further improve the computational efficiency, we propose the second network architecture: PhyCRNet-s. We keep the majority of neural components in PhyCRNet, except skipping the encoder part periodically and setting a different input flow. Here we introduce a cycle index of skipping-encoder $\mathcal{T}$ ($\mathcal{T} \leq T$) for removing the redundant computation of the encoder. For instance, we have the HR input at time instant 0 and $\mathcal{T}-1$ (i.e., $\mathbf{u}_0$ and $\mathbf{u}_{\mathcal{T}-1}$) where the encoder part of PhyCRNet is implemented, whereas the LR output feature from ConvLSTM is directly transmitted to the next step as LR input during the time period $[1, \mathcal{T}-2]$. The graphic illustration of skipping-encoder scheme is shown in Fig. 3. Since the encoder–decoder component shares the same learnable parameters and is not involved in temporal evolution, it is derived that the sparse residual dynamics has been learned in ConvLSTM. Namely, at time instant t_i, ConvLSTM works as a sparse dynamics propagator between the LR input feature $\mathbf{X}_i$ and the LR output variable $\mathbf{X}_{i+1}$. Then $\mathbf{X}_{i+1}$ can be naturally regarded as the LR input at time instant t_{i+1}. Similarly, such a construction also implicates the forward Euler scheme on sparse dynamics: $\mathbf{X}_{i+1} \approx \mathbf{X}_i + \delta t \cdot \mathcal{NN}^c[\mathbf{X}_i; \boldsymbol{\theta}^c]$ where $\mathcal{NN}^c[\cdot]$, denotes the ConvLSTM network and $\boldsymbol{\theta}^c$ is its corresponding trainable parameters.

Although it brings a more light-weight architecture, this skipping-encoder scheme may cause approximation error in the temporal propagation. To seek for a tradeoff between accuracy and efficiency, we tend to set $\mathcal{T}$ as a relatively small value in case of error accumulation. The selection of $\mathcal{T}$ is empirically discussed in Section 4.7.

3.5. Hard imposition of I/BCs

The motivation of hard imposition of I/BCs is to prompt a well-posed optimization problem when training the network, which contributes to the improvement of solution accuracy and the facilitation of convergence [41,73,74]. Furthermore, the general philosophy behind it is to strictly integrate the known information of I/BCs into the network. Note that the IC can be easily imposed through the residual connection shown in Figs. 2 and 3. Since the solution field is uniformly discretized, it is suitable to apply padding for message passing pixel-wisely. For Dirichlet BCs, the known constants on the boundaries can be rigorously incorporated into the state variables via time-invariant padding operation. For Neumann BCs, ghost elements [1] are necessary for forcible satisfaction beyond the boundaries, whose values can be approximately inferred with finite difference. Herein, the relationship between ghost nodes and internal nodes is time-invariant, but the values for padding change during the training process. The idea of hard imposition of I/BCs is illustrated in Fig. 4.

Specifically, we consider solving PDEs with periodic Dirichlet BCs in this work. Hence, the periodic padding (also known as circular padding) is introduced here for handling the constant copy of boundary values. The boundary padding operation aims to ensure that the minimization of PDE residuals proceeds in the entire solution domain

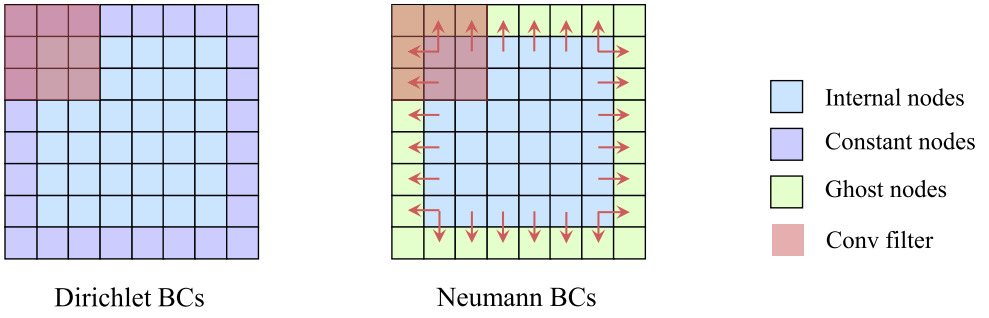


Fig. 4. A graphic demonstration of hard imposition of BCs into the network. For the Dirichlet BCs (left figure), the constant nodes are padded on the boundaries. For the Neumann BCs (right figure), the ghost nodes are padded with values derived from internal field. Moreover, we implement the hard enforcement of BCs before the convolution operations both in the learning process and the differentiation.

without sacrificing any boundary nodes. Here we extend the periodic padding to all the feature maps produced by convolutional operations including those in ConvLSTM. We find it helps boost the solution accuracy on the boundaries compared with zero-padding. In addition, the periodic padding will be implemented twice on the boundaries when applying the fourth-order finite-difference-based filtering for differentiation.

3.6. Physics-informed loss function

Thanks to the hard enforcement of I/BCs, we only need to construct the loss function based on the governing PDEs. Firstly, we define the PDE residual $\boldsymbol{\mathcal{R}}(\mathbf{x}, t; \boldsymbol{\theta})$ given by

$$\boldsymbol{\mathcal{R}}(\mathbf{x}, t; \boldsymbol{\theta}) := \mathbf{u}_t^{\theta} + \mathcal{F}[\mathbf{u}^{\theta}, \nabla_{\mathbf{x}}\mathbf{u}^{\theta}, \nabla_{\mathbf{x}}^2\mathbf{u}^{\theta}, \ldots; \boldsymbol{\lambda}], \tag{5}$$

which is exactly the left-hand-side of Eq. (1) with quantities of interest learned by the network. Furthermore, the shared network parameters $\boldsymbol{\theta}$ can be trained by minimizing the loss function $\mathcal{L}(\boldsymbol{\theta})$, which is equivalent to the mean of squares of the PDE residuals over the spatiotemporal discretization. Taking a two-dimensional PDE system as an example, $\mathcal{L}(\boldsymbol{\theta})$ can be expressed as

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{nmT}\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{T}\|\boldsymbol{\mathcal{R}}(x_i, y_j, t_k; \boldsymbol{\theta})\|_2^2, \tag{6}$$

where n and m denote the height and width in the spatial grid, T is the total number of time steps, and $\|\cdot\|_2$ denotes ℓ_2 norm.

4. Numerical experiments

In this section, we evaluate the performance of our proposed methods on three nonlinear PDE systems, and compare them with two baseline algorithms: the vanilla PINN approach [9] and the AR-DenseED model [47]. These numerical experiments cover the Burgers' equations and two RD systems (i.e., λ-ω and FitzHugh–Nagumo equations) in 2D domains with periodic BCs. The specific experiments include three groups: (1) comparing the solution accuracy and extrapolability of PhyCRNet with baseline algorithms; (2) testing the generalization ability of PhyCRNet to different ICs; (3) comparing the performance between PhyCRNet and PhyCRNet-s. For all of these experiments, the spatial resolutions of input and output state variables are set as 128×128. All the numerical implementations in this paper are coded in Pytorch [75] and performed on a NVIDIA Tesla V100 GPU card (32G) in a standard workstation.[1]

[1] Source codes/datasets are available on GitHub at https://github.com/isds-neu/PhyCRNet upon publication.

4.1. Network setup

Herein, we consider the same network setting, of PhyCRNet and PhyCRNet-s, for all PDE cases. The encoder consists of three convolutional layers with 8, 32, 128 units respectively, using 4×4 kernels and a stride of 2. The periodic padding is applied to all convolutional operations. Afterwards, one ConvLSTM layer is merged on the latent space with cell/hidden states of 128 nodes. The convolutional operations in ConvLSTM have 3×3 kernels and a stride of 1. For the scaling layer before the output, we use a larger kernel (5×5) and the same stride in consideration of filtering on high-resolution spatial features. The upsampling factor for pixel-shuffle is 8. The networks are trained by the stochastic gradient descent Adam optimizer [76]. We train PhyCRNet for 10,000 epochs based on a shorter period of evolution, e.g., $T/3 \sim 2T/3$.

4.1.1. Baseline model setup

The training implementations of baselines are based on the reasonable neural architectures and training efforts similar to our proposed methods for fair comparison. Besides, we apply the same network parameters of the baselines to solve all PDE systems. We mainly compare the solution snapshots between PhyCRNet and PINN as a typical case of discrete and continuous learning, and compare the error propagation among PhyCRNet, PINN and AR-DenseED.

The PINN model has 4 fully-connected layers, each with 80 nodes. A total of 1.62×10^5 collocation points are used to evaluate the total loss function (e.g., PDE residual loss + I/BC loss), trained by 1×10^4 epochs using Adam followed by up to 1×10^5 epochs with the L-BFGS optimizer [77]. More details on the selection of hyper-parameters of PINN is provided in Appendix A. For the AR-DenseED method which incorporates the AR process into the DenseNet architecture [78], we leverage the open source code and network parameter setting in [47]. The AR-DenseED model is constitutive of an encoding convolutional block, single dense block and a decoding block of different dense layers $\{4, 3, 4\}$, respectively. The growth rate is set as 4. We encode a 2D input variable consisting of five previous time-steps to latent features of half spatial dimensionality of input, and then latent features are decoded to the prediction of next time-step. In addition, we train the network for 1000 epochs using Adam with exponential decay rate of 0.995 after pre-training.

4.1.2. Evaluation metrics

To evaluate the solution accuracy produced by all the NN-based solvers, we assess the full-field error propagation in two phases: training and extrapolation. The definition of the full-field error ϵ_τ at time τ is defined as the accumulative root-mean-square error (a-RMSE) given by

$$\epsilon_\tau = \sqrt{\frac{1}{N_\tau} \sum_{k=1}^{N_\tau} \frac{\|\mathbf{u}^*(\mathbf{x}, t_k) - \mathbf{u}^\theta(\mathbf{x}, t_k)\|_2^2}{mn}}, \tag{7}$$

where N_τ is the total number of time steps within $[0, \tau]$, and $\mathbf{u}^*(\mathbf{x}, t_k)$ is the reference solution.

4.2. 2D Burgers' equations

We firstly consider a 2D fluid dynamics problem, the famous Burgers' equations, given in the following tensor form:

$$\mathbf{u}_t + \mathbf{u} \cdot \nabla \mathbf{u} - \nu \Delta \mathbf{u} = \mathbf{0}, \tag{8}$$

where $\mathbf{u} = \{u, v\}$ denotes the fluid velocities; ν is the viscosity coefficient; Δ is the Laplacian operator. The 2D Burgers' equations describe the complex interaction of nonlinear convection and diffusion processes with possible shock behaviors, which usually acts as a benchmark model for comparing different computational algorithms. Herein, we choose $\nu = 0.005$ and the spatial domain size as $\mathbf{x} \in [0, 1]$ [47] for performance comparison.

Moreover, the IC is generated from a Gaussian random field with periodic BCs using the open source code in [54]: $\mathbf{u}_0 \sim \mathcal{N}(0, 625(-\Delta + 25\mathbf{I})^{-2})$. The ground truth reference solution is calculated using a high-order finite difference method with the 4th-order Runge–kutta time integration ($\delta t = 1 \times 10^{-4}$). While for the PhyCRNet, we choose a relatively larger time interval $\delta t = 0.002$ over the discretized domain considering the implicit time marching in ConvLSTM. The empirical results show that the time step size of PhyCRNet can be approximately

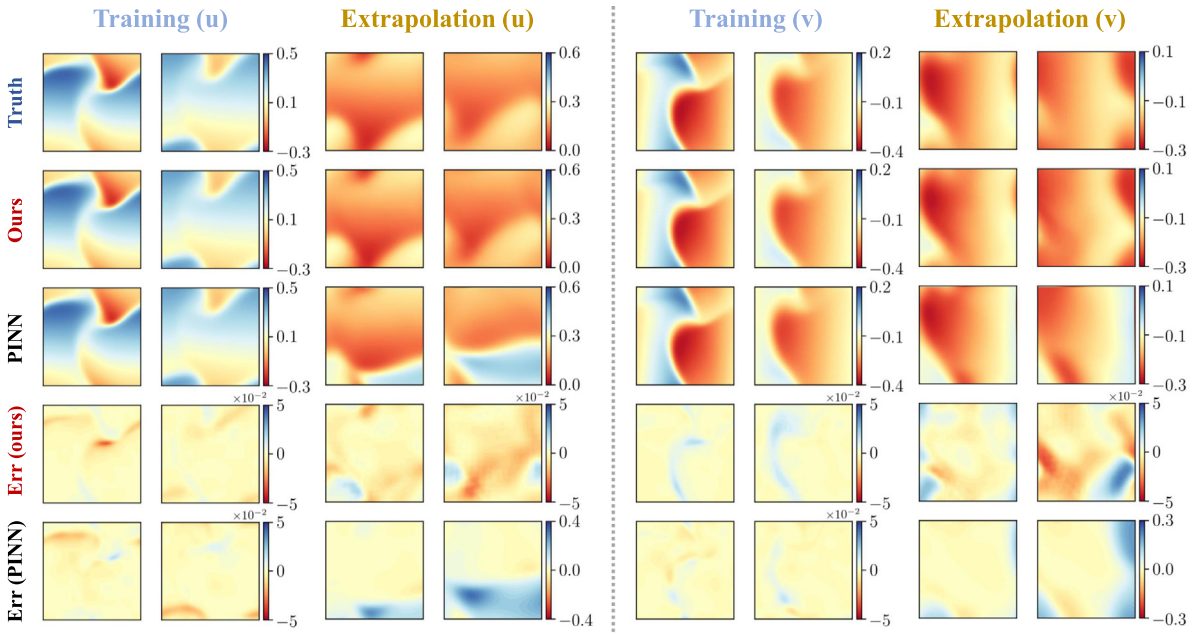


Fig. 5. Comparison of solution accuracy and extrapolability between PhyCRNet and PINN for the 2D Burgers' equations. Four representative time instants are chosen for training ($t = 1.0, 2.0$) and extrapolation ($t = 3.0, 4.0$) phases. Err (ours) and Err (PINN) refer to the difference in the entire domain between ground truth reference and prediction by these two approaches.

at least 2 times of maximal δt of traditional finite difference methods. Moreover, we train PhyCRNet to obtain the numerical solution of the Burgers' equations for 1000 time steps within the duration of $[0, 2]$. The training time is 24 h. Furthermore, based on the trained model, we predict the solution for another 1000 time steps (within time $[2, 4]$) to test the extrapolability of our proposed method. The learning rate starts at 6×10^{-4} and decays every 50 epochs by 1%.

The solution snapshots predicted by PhyCRNet and PINN are shown in Fig. 5 in comparison with the ground truth reference, and the loss history is included in Appendix B. In particular, we select four representative snapshots in the training and extrapolation phases at $t = 1.0, 2.0$ and $t = 3.0, 4.0$ respectively for illustration. The solution error propagation with time for both methods is presented in Fig. 8(a). Firstly, it can be seen in Fig. 5 that the trained and extrapolated results using PhyCRNet both possess excellent agreement with the reference solution, while the PINN results fail to match the ground truth. In particular, we observe that most of the error field from PhyCRNet is close to zero, whereas the outcome of PINN exhibits much larger error especially on the boundaries (due to "soft" imposition of I/BCs). Secondly, the error propagation in Fig. 8(a) further validates the superior solution accuracy of PhyCRNet with a-RMSE always below 0.01. PhyCRNet holds similar performance with PINN in training but leads in extrapolation, and prevails AR-DenseED both in training and extrapolation by two orders of magnitude. The result of AR-DenseED here turns out to be unsatisfactory on account of longer time evolution and larger spatial discretization compared to the 2D Burgers' case in [47]. The reason behind this phenomenon lies in the network designing of AR-DenseED only relying on fully-connected convolutional operations, which lacks sufficient expressiveness on explicit temporal propagation. Besides, the error propagation curve of PhyCRNet remains flattened along with time marching. The minor discrepancy between training and extrapolation shows the great potential of PhyCRNet for solution generalization.

In addition, PhyCRNet can be flexibly modified (e.g., the grid size δx and the time spacing δt) in order to handle the 2D Burgers' equations with different viscosity coefficients. For instance, for larger Reynolds numbers, we can correspondingly increase the grid size and reduce the time spacing size to guarantee the convergence. Note that the discretization requirement of PhyCRNet is less strict than that of traditional finite difference methods due to the implicit time marching scheme of our framework. Moreover, we also provide a convergence study on the grid sizes (i.e., 32, 64 and 128) to investigate their effects to the performance of PhyCRNet Appendix C.

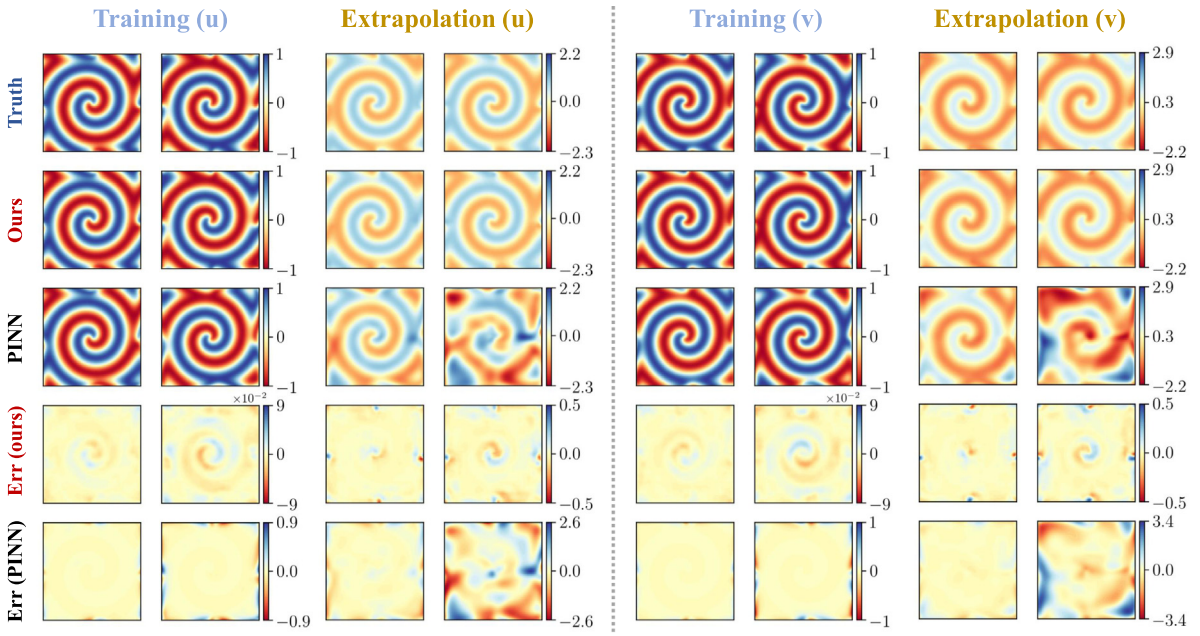


Fig. 6. Comparison of solution accuracy and extrapolability between PhyCRNet and PINN for the λ-ω RD equations. Four representative time instants are chosen for training ($t = 2.5, 5.0$) and extrapolation ($t = 7.5, 10.0$) phases. Err(ours) and Err(PINN) refer to the difference in the entire domain between ground truth reference and these predicted by two DNN-based approaches.

4.3. λ-ω RD equations

The second example considered here is a λ-ω RD system in a 2D domain, which is widely known for its representation of multi-scale phenomenon for chemical and biological processes, e.g., turbulent behavior and self-organized patterns. Specifically, the two coupled nonlinear PDEs with the formation of spiral wave patterns are expressed as:

$$\begin{aligned} u_t &= 0.1\Delta u + \lambda(r)u - \omega(r)v, \\ v_t &= 0.1\Delta v + \omega(r)u + \lambda(r)v, \end{aligned} \tag{9}$$

where u and v are two field variables; $r = u^2+v^2$; λ and ω are two real functions given by $\lambda = 1-r^2$ and $\omega = -r^2$, respectively. The reference solution was generated using a spectral method [79] in the domain of $[-10, 10]$ for 801 time steps ($\delta t = 0.0125$). We train the model for 200 time steps with time duration of [0, 5], and perform the inference for [5, 10], where $\delta t = 0.025$. The learning rate is set as 5×10^{-4} and decays by 2% every 100 epochs.

The solution snapshots predicted by PhyCRNet and PINN, compared with the ground truth reference, are shown in Fig. 6. Given the simplicity of periodic patterns, both PhyCRNet and PINN produce good result in the training phase. We consider this phenomena coming from the truth that the portraits of this specific λ-ω RD system are simple and smooth, where the continuous approximation of solution by PINN plays a positive role. However, PINN yields large error on the boundaries and does not extrapolate well even for such simple patterns. On the contrary, PhyCRNet performs robustly with much smaller distributed errors even in the extrapolation phase. Moreover, Fig. 8(b) depicts the error propagation for PINN, AR-DenseED and PhyCRNet, where we observe that PhyCRNet outperforms both PINN and AR-DenseED with up to one order of magnitude smaller error.

4.4. FitzHugh–Nagumo RD equations

The final example aims to further test the capability of PhyCRNet for solving PDEs. Herein, we consider an explicable and mathematical portrait of neural excitation, namely, FitzHugh–Nagumo (FN) RD equations, written

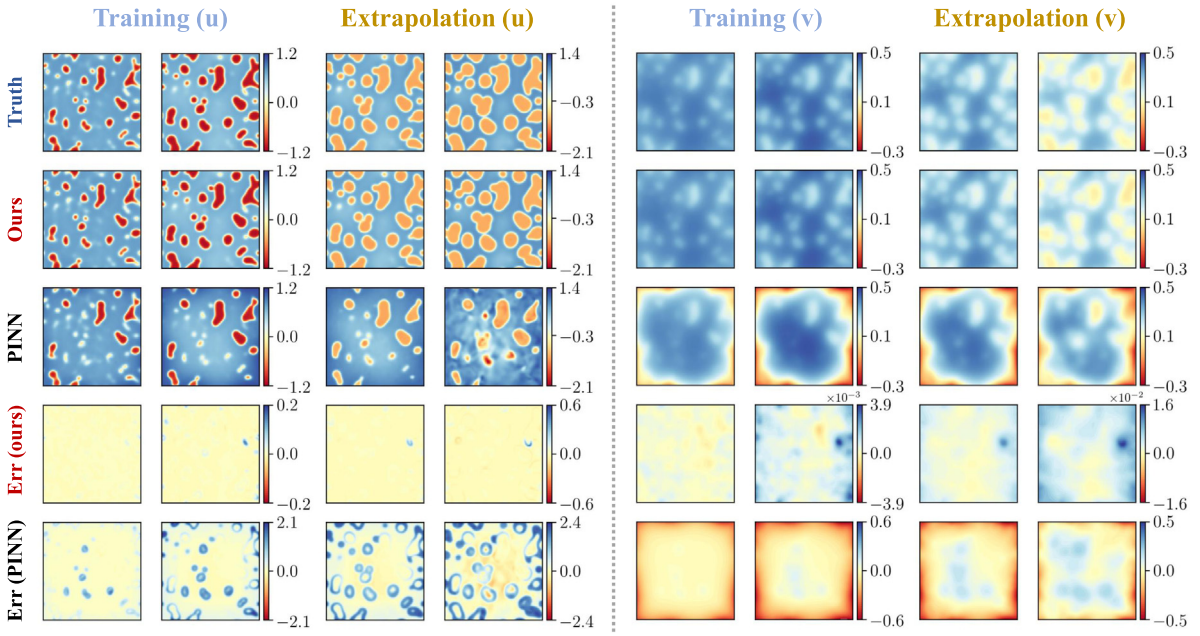


Fig. 7. Comparison of solution accuracy and extrapolability between PhyCRNet and PINN for the FitzHugh–Nagumo RD Equations. Four representative time instants are chosen for training ($t = 2.16, 4.32$) and extrapolation ($t = 6.48, 8.64$) phases. Err(ours) and Err(PINN) refer to the difference in the entire domain between ground truth reference and these predicted by two DNN-based approaches.

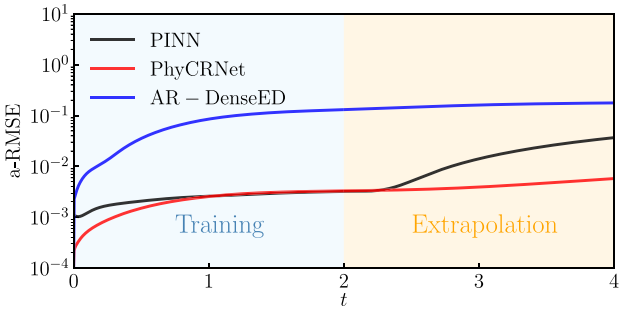
as:

$$\begin{aligned} u_t &= \gamma_u \Delta u + u - u^3 - v + \alpha, \\ v_t &= \gamma_v \Delta v + \beta(u - v), \end{aligned} \tag{10}$$

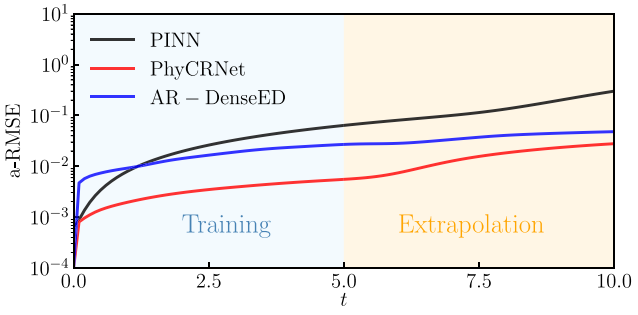
where u and v are two interactive components; $\gamma_u = 1$ and $\gamma_v = 100$ are diffusion coefficients, while $\alpha = 0.01$ denotes the external stimulus and $\beta = 0.25$ is the coefficient for reaction terms. With different diffusion and reaction coefficients, we can simulate varying neuron activities. The IC is generated by drawing random samples from a Gaussian distribution and the ground truth solution is produced by finite difference with the 4th-order Runge–Kutta time integration, in 2D domain $\Omega = [0, 128]$ with $\delta t = 2 \times 10^{-4}$. We attempt to train PhyCRNet to solve this PDE for 750 time steps with $\delta t = 0.006$ (i.e., time duration $[0, 4.5]$), and use the trained model to extrapolate the solution in $[4.5, 9]$. The learning rate is set as 5×10^{-5} and the decaying coefficient is 0.995 for every 50 epochs.

The solution snapshots predicted by PhyCRNet and PINN are shown in Fig. 7, along with the ground truth reference and error maps. It is notable that the FN system demonstrates more complex and interactive patterns compared with the previous two examples. Herein, PhyCRNet presents outstanding goodness of fit with the ground truth both in training and extrapolation, whereas PINN can barely simulate parts of the dynamical patterns. Moreover, the PhyCRNet error maps exhibit almost perfect results that are close to zeros, especially for the field variable v. The relatively large errors mainly cluster on propagating wave fronts (e.g., mid-right locations). Nevertheless, considering the complex patterns of this PDE system, this kind of error is negligible. Moreover, it is rather challenging to capture the contours of the evolutionary dynamics, especially for the long-term modeling of DNN-based approaches. Note that although PhyCRNet is inspired from the forward Euler scheme which usually does not present such overshooting solutions, it is quite different from the traditional forward simulation due to the nonlinearity of NNs.

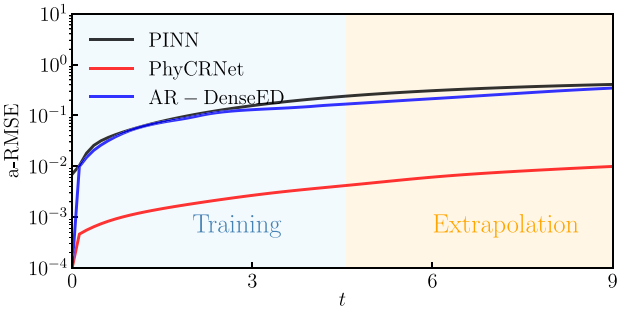
For the error propagation shown in Fig. 8(c), PhyCRNet surpasses PINN and AR-DenseED up to two orders of magnitude. The a-RMSE of PhyCRNet increases mildly along with time but always keeps at a low level (i.e., 10^{-2}), while the errors of PINN and AR-DenseED rise to the magnitude of 1. Noteworthy, the error propagation in PhyCRNet extrapolation is relatively larger compared with the training phase, primarily due to the complicated



(a) 2D Burgers' equations



(b) λ-ω RD equations



(c) FN RD equations

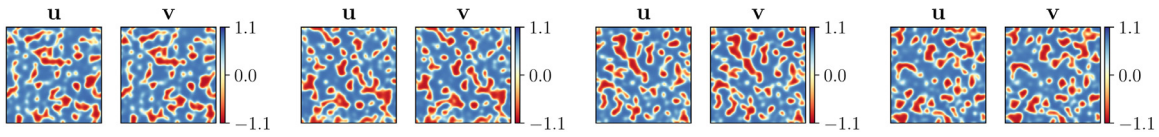
Fig. 8. Comparison of error propagation between PhyCRNet and PINN for three PDE systems. The curves in blue and orange areas present error propagation in the training and extrapolation phases, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

interaction between the field components (e.g., with sharp propagating wave fronts). This problem can be alleviated by involving a longer time duration in training such that richer dynamics is captured.

In general, PhyCRNet is highly capable of learning the spatiotemporal dependencies of PDE systems, ranging from fluid dynamics to RD systems. The salient outcomes of solution accuracy and extrapolation capability demonstrate PhyCRNet to be a novel and powerful method to solve complex multi-dimensional PDEs.

4.5. Generalization to different ICs

Theoretically, an ideal DNN-based solver should be capable of providing accurate solutions given any IC for a specific PDE. However, PINN fails to generalize to different ICs due to the intrinsic limitation of softly incorporating I/BCs into loss functions. Our proposed method (i.e., PhyCRNet/PhyCRNet-s) can generalize well outside the



(a) IC #1 (b) IC #2 (c) IC #3 (d) IC #4

Fig. 9. Four randomly generated ICs.

training phase, with respect to the system parameters and spatiotemporal discretization. This prominent strength comes from the architectural design of our networks, especially the AR input–output flow and hard-imposition of I/BCs.

We evaluate the generalization performance of our model using the FN RD equations under four different ICs, as shown in Fig. 9. These four ICs are randomly sampled from Gaussian distribution with mean and standard deviation as 0 and 0.1, respectively. Compared with the extrapolation in Section 4.4, we consider inferring a longer temporal evolution (i.e., 4500 time instants referring to the time duration [0, 27]) for all these IC scenarios. The testing results are presented in Fig. 10, where we select four typical snapshots (i.e., $t = 4.32, 8.64, 12.96, 17.28$) of both $\mathbf{u}$ and $\mathbf{v}$ for illustration in comparison with the ground truth. All of the predicted results reveal the excellent abilities of our model to capture the evolutionary patterns and portray the local details. The positive evidence is also observed in error propagation described in Fig. 11(a). The error variances share similar tendency which increase smoothly and slowly, and are bounded below 0.04. Overall, these evaluation experiments suggest our model learns the underlying physical laws well and generalize to different ICs robustly.

4.6. Ablation study

To justify the effectiveness of our designed frameworks, we implement an ablation study on PhyCRNet by solving the 2D Burgers' equations. Apart from the encoder–decoder, the rest of the network architecture consists of ConvLSTM, residual connection, input–output autoregressive (AR) scheme and filtering-based differentiation. Here we mainly investigate the contributions of AR scheme and global residual connection. The experimental setting includes three architectures: full PhyCRNet, PhyCRNet without AR scheme and PhyCRNet without global residual connection. We apply the unified training efforts for performance evaluation, which is exactly same in Section 4.2. The comparison result is described in Fig. 11(b). The structure of PhyCRNet without AR scheme works worst, obtaining larger errors by nearly two orders of magnitude compared with PhyCRNet. Additionally, PhyCRNet without the global residual connection also exhibits inferior performance both in the training and extrapolation stages, compared with the full model. Therefore, these two ablation experiments validate the significance and indispensability of AR scheme and global residual connection.

4.7. Comparison between PhyCRNet and PhyCRNet-s

Our idea of physics-informed convolutional-recurrent networks enjoys prominent flexibility in the architecture designing, where PhyCRNet and PhyCRNet-s are both proposed. More precisely, PhyCRNet is a special class of PhyCRNet-s with $\mathcal{T} = 0$. The skip-encoder strategy with respect to different $\mathcal{T}$ can effectively provide a balance/tradeoff between accuracy and efficiency.

In this part, we investigate the effects of hyper-parameter $\mathcal{T}$ to the performance of network architectures, considering the 2D Burgers' equations as a testing example. Herein, we select four different architectural structures with $\mathcal{T} = \{0, 10, 50, 100\}$ respectively, and keep other neural components fixed. All of the four experiments here undergo the same training procedures, which include 10,000 epochs using Adam after pre-training with the learning rate starting at 6×10^{-4} and decaying every 50 epochs by 1%.

The evaluation results are presented in Table 1 and Fig. 11(c). Herein, we define a relative full-field error with respect to the ground truth, in order to assess the model performance in training and extrapolating stages. As shown in Table 1, the general trend is that with the parameter $\mathcal{T}$ increasing, the training effort (i.e., computational time) decreases mildly and the solution errors both in training and extrapolation grow. However, there is also one

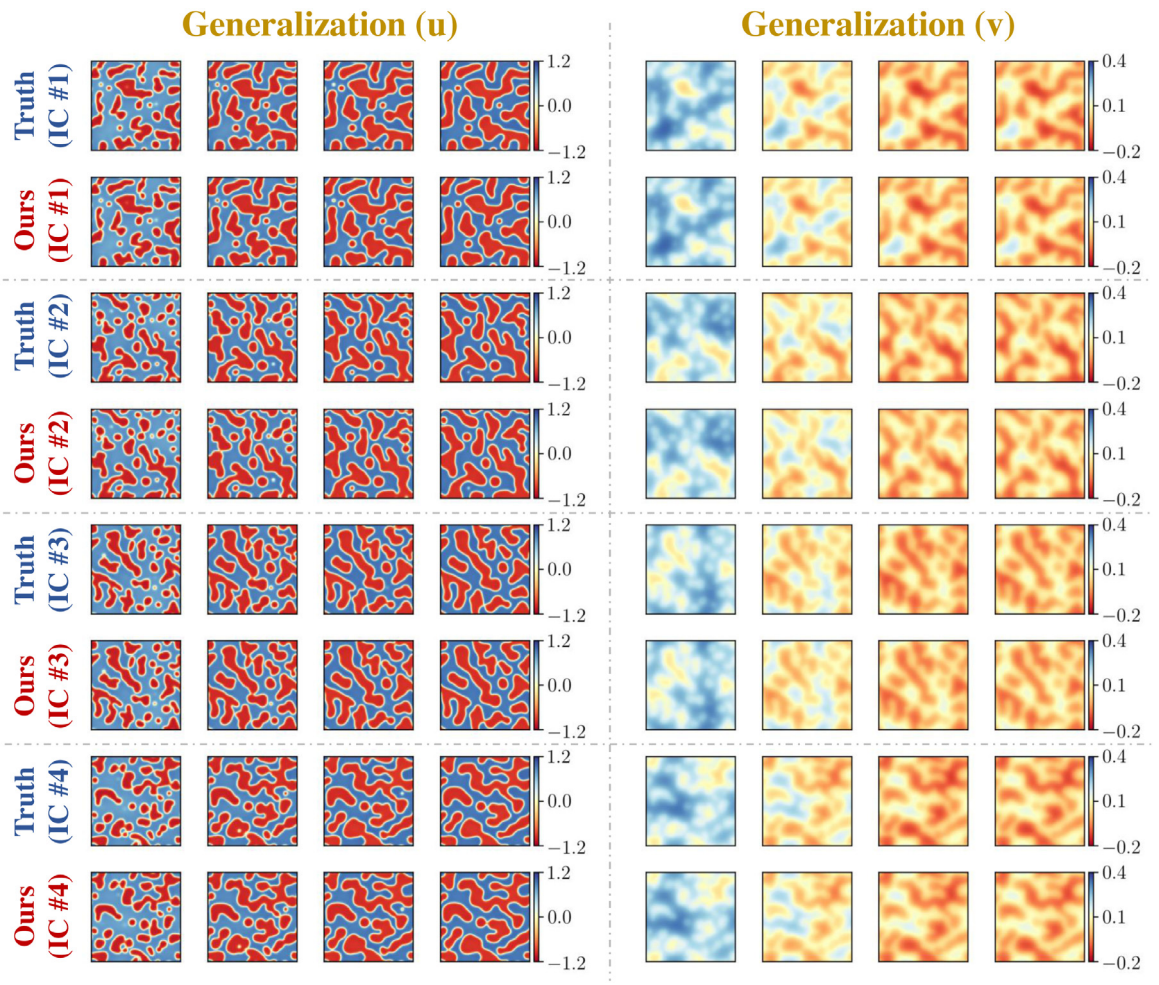
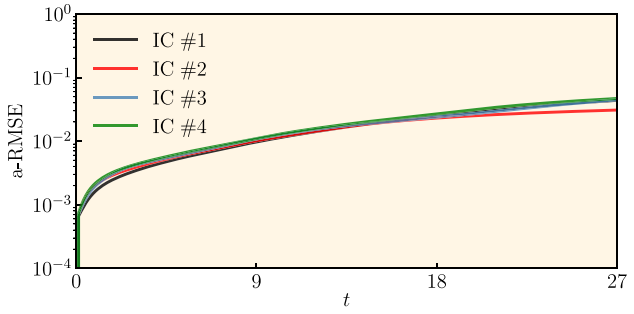


Fig. 10. Generalization for four testing ICs of FN RD equations. Four representative time instants are chosen (i.e., $t = 4.32, 8.64, 12.96, 17.28$).

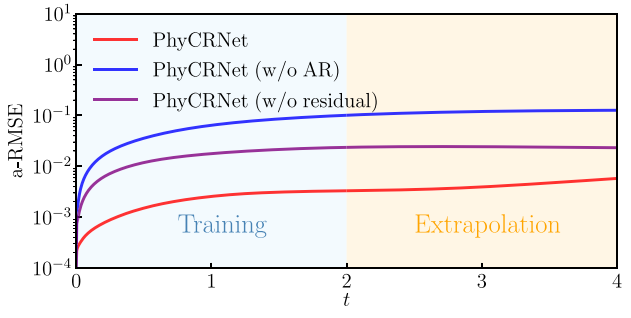
Table 1
Performance comparison between PhyCRNet and PhyCRNet-s.

Model	Time [s]/epoch	Training error [%]	Extrapolating error [%]
PhyCRNet	8.64	0.83	3.40
PhyCRNet-s ($\mathcal{T} = 10$)	8.18	1.28	2.97
PhyCRNet-s ($\mathcal{T} = 50$)	8.04	2.09	3.93
PhyCRNet-s ($\mathcal{T} = 100$)	7.94	2.94	7.52

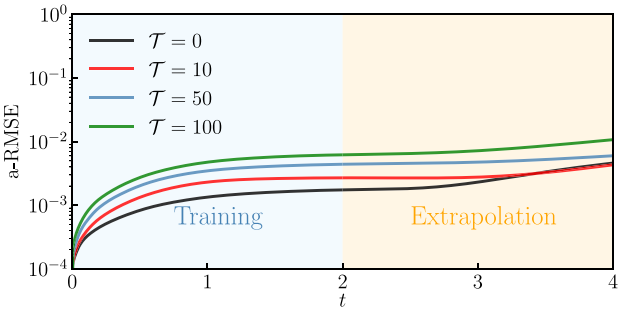
surprising observation that PhyCRNet works inferiorly than PhyCRNet-s ($\mathcal{T} = 10$) in the extrapolation, though leads in training solution accuracy. We think the reason behind this interesting phenomenon is that a small $\mathcal{T}$ helps relax the model training, which enhances the extrapolation ability. Moreover, the details of error propagation described in Fig. 11(c) further validates the significant result. In the training phase, the model with smaller $\mathcal{T}$ behaves more excellent. Nevertheless, when extrapolating relatively more time steps, PhyCRNet becomes weaker against ($\mathcal{T} = 10$). Therefore, we conclude that a reasonably small $\mathcal{T}$ contributes to the balance between accuracy and efficiency as well as the robustness for extrapolation.



(a) Generalization test



(b) Ablation study



(c) Selection of $\mathcal{T}$

Fig. 11. (a) Error propagation for generalization testing at different ICs. t is time duration ($[0, 27]$). (b) Ablation study on neural architectures of PhyCRNet. (c) Error propagation of PhyCRNet (i.e., $\mathcal{T} = 0$) and PhyCRNet-s (i.e., $\mathcal{T} = \{10, 50, 100\}$) frameworks.

5. Discussion

In this section, we aim to make a comprehensive and comparative discussion between our discrete methods and the standard continuous approaches (i.e., PINNs), as well as provide future perspectives based on the current investigation. Prior studies [9,18] have shown the great potential of PINNs for forward and inverse analysis of spatiotemporal PDEs, but the comparison against discrete methods are still lacking. Generally, there are two distinctive aspects, mainly covering mesh requirement and encoding of I/BCs.

First of all, as a continuous learning method, PINNs hold the apparent strength of meshfree, while our PhyCRNet and PhyCRNet-s frameworks rely on prescribed spatiotemporal mesh and are limited to regular grids. However, thanks to the emerging geometric learning methods (e.g., graph neural networks [80–84]), we can extend the current architectures to irregular meshes/grids, making them more versatile. Besides, due to the way of functional approximation with global basis, PINNs are dedicated for "global learning" which helps capture the general dynamical patterns of nonlinear PDE systems. Discrete learning methods focus on local features within the physical

domains because of the employment of convolutional kernels, which provides a more reliable way of capturing local morphology in the PDE solution.

Secondly, the enforcement of I/BCs significantly affects the solution accuracy of PDEs, especially on the boundaries. For PINNs, the soft imposition of I/BCs, which is presented as a penalty factor in loss function, requires massive hyper-parameter tuning for the optimal weight of I/BC loss component. Usually, such a soft strategy cannot guarantee the solution accuracy on the boundaries, and fail to generalize to different ICs as discussed in Section 4. On the other hand, for our discrete approaches, it is easy and suitable to rigorously incorporate the discrete boundary values into the network via an intrinsic padding scheme, which painlessly and remarkably promotes the solution accuracy on the boundaries. In addition, instead of integrating IC loss into the loss function, we define the IC as the first input state variable for the network, which makes generalization to different IC scenarios achievable. Apart from the I/BCs, the known conservation laws (e.g., mass conservation) can also be enforced simultaneously if applicable. For example, strict enforcement of mass conservation can be realized by applying a stream function as the solution variable in the network for fluid dynamics. Besides, it is more significant and realistic to design a BC encoding strategy for handling various BCs. The idea is to utilize parametric learning, where parametric BCs are encoded to serve as an additional input to the network.

6. Conclusion

Solving PDEs is fundamental to a wide range of scientific computational problems, where physics-informed DNNs show promise to tackle relevant challenges especially in inverse problems and data assimilation. The majority of existing methods may suffer from issues of scalability, error propagation and generalization. This motivates us to develop a novel PhyCRNet/PhyCRNet-s architecture as a universal model for solving spatiotemporal PDEs. The hard enforcement of I/BCs via designated padding prompts a well-posed optimization problem in network training, improves the solution accuracy and facilitates convergence. Finally, we validate the excellent performance of our proposed methods in solution accuracy, extrapolability, and generalizability. These proposed networks have promising potential to serve as a reliable surrogate model for data assimilation and inverse analysis of physical systems where data is scarce and noisy, which will be demonstrated in the next step of our study.

However, we also see several limitations of the current PhyCRNet and aim to discuss the future research perspectives here. Firstly, our current networks can be naturally extended to tackle irregular spatial domains by incorporating graph neural networks, as well as modified from forward Euler scheme to high-order difference strategy (e.g., high-order Runge–Kutta scheme) for more accurate temporal evolution modeling. Secondly, for second-order PDEs with $\mathbf{u}_{tt}$ in the PDEs, we can turn the equations to a state–space PDE system, e.g., by introducing an additional state variable $\mathbf{z} = \mathbf{u}_t$ and then $\mathbf{z}_t = \mathbf{u}_{tt}$. These second-order PDE tasks are thus simplified to the first-order time derivative problem with the solution variable of $[\mathbf{u}, \mathbf{z}]^T$ discussed in this paper. Thirdly, another limitation is that our discretization for the physical domain is fixed which is used mainly for Eulerian dynamics. We will investigate the learnable and adaptive mesh approach for Lagrangian systems in the future. Additionally, it is also worthwhile to explore the potential of PhyCRNet in a variational form for solving PDEs since many existing investigations have proven the effectiveness of variational deep learning approaches [20,85,86].

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The datasets and computer codes are available on https://github.com/isds-neu/PhyCRNet upon final publication of the paper.

Acknowledgments

Pu Ren would like to acknowledge the support by the Vilas Mujumdar Fellowship at Northeastern University, USA. Yang Liu acknowledges the support by the TIER 1 Seed Grant Program at Northeastern University, USA. Hao Sun acknowledges the support by the Engineering for Civil Infrastructure program at National Science Foundation, USA under grant CMMI-2013067 and the research award from MathWorks, USA.

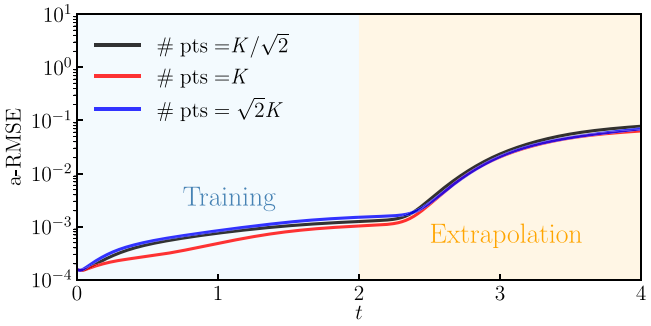


Fig. A.1. The convergence study on the number of collocation points for PINNs.

Appendix A. Clarification on the hyper-parameters for PINNs

We provide all of the training details and justify the selection of the hyper-parameters for PINNs in this paper, including the determination of collocation points and the weighting coefficients for I/BCs. Firstly, we generate the collocation points with Latin hypercube sampling (LHS) in the spatiotemporal domain. Moreover, the number of collocation points K is set as 1.62×10^5 in this paper after empirical trials. A convergence test is performed to ensure the number of collocation points for PINN is sufficient. The alternative numbers are $\sqrt{2}K$ and $K/\sqrt{2}$. All of the numerical experiments here are tested on 2D Burgers' equations. As shown in Fig. A.1, the network with K collocation points performs best, and the increase of collocation points (i.e., $\sqrt{2}K$) has no positive effects on the improvement of solution accuracy and extrapolation due to the optimization issue. Besides, PINNs with fewer collocation points (i.e., $K/\sqrt{2}$) also show the inferior performance than the network with K collocation points in error propagation, which is induced by the insufficient expressiveness of the PDE manifold. Hence, we set the number of collocation points as K for PINN experiments in this paper.

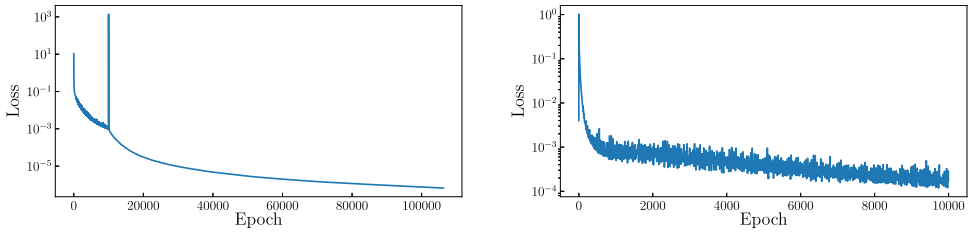
In addition, the training process consists of 1×10^4 iterations with Adam optimizer and a maximum 1×10^5 iterations with L-BFGS until convergence. The initial learning rate is set to be 0.001 and multiplied by 0.97 every 200 iterations, and we employ Xavier initialization for the initialization of the network parameters (i.e., weights and biases). Moreover, the number of collocation points are 16,384 (i.e., 128^2) for the IC and 4.8×10^4 for the BC. Regarding the weighting coefficients in $\mathcal{L} = \mathcal{L}_{\text{phy}} + \lambda_1 \cdot \mathcal{L}_{\text{IC}} + \lambda_2 \cdot \mathcal{L}_{\text{BC}}$ [9], we set $\lambda_1 = 10$ and $\lambda_2 = 10$ for the I/BC loss components, respectively. They are selected through the grid search with $\lambda_1 \in \{1, 10, 100\}$ and $\lambda_2 \in \{0.1, 1, 10\}$. The center of the grid $(\lambda_1, \lambda_2) = (10, 1)$ is chosen so that each loss component has the same scale before the training.

Appendix B. Loss history

The loss histories of training on three PDEs are similar. Therefore, we select 2D Burgers' equations as the representative case for exhibiting the convergence histories of PINNs and PhyCRNet (see Fig. B.2). Here the training loss of PINNs is smaller than that of PhyCRNet due to two reasons: (1) The number of collocation points used in PINNs is smaller than the grid size of PhyCRNet, which facilitates the convergence of PINNs; (2) The training epoch of PINNs is much more than that of PhyCRNet. Thus, we observe a longer history of loss decreasing in PINNs. Nevertheless, both methods achieve satisfactory solution accuracy in the training phase (see Fig. 8(a) for example).

Appendix C. Convergence study on grid sizes

For discrete learning methods, it is significant to conduct the convergence study on the sensitivity of grid sizes. Herein, by testing on 2D Burgers' equations, we investigate the performance of PhyCRNet with the grid sizes δx as 32, 64 and 128 respectively. As shown in Fig. C.3, we observe that the PhyCRNet frameworks with $\delta x = 128$ and $\delta x = 64$ have very close performance on error propagation though the network with $\delta x = 64$ shows the slightly weaker capability on solution accuracy. However, the PhyCRNet with $\delta x = 32$ presents unsatisfactory results due to the coarse discretization. Overall, from coarse grids to fine meshes, the numerical results exhibit a convergent tendency at $\delta x = 128$.



(a) Training loss of PINNs (b) Training loss of PhyCRNet

Fig. B.2. Training loss of PINNs and PhyCRNet for testing on 2D Burgers' equations.

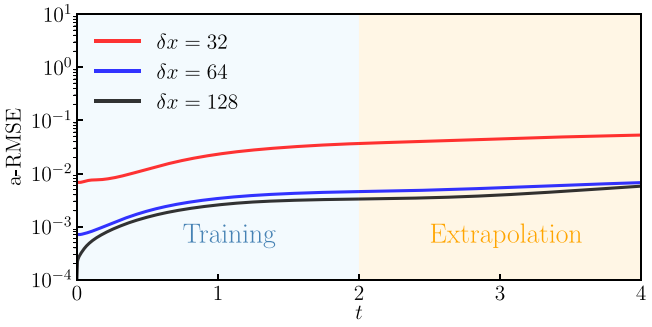


Fig. C.3. The convergence study on grid sizes of PhyCRNet.

References

[1] T.J. Hughes, The Finite Element Method: Linear Static and Dynamic Finite Element Analysis, Courier Corporation, 2012.

[2] T.J. Hughes, J.A. Cottrell, Y. Bazilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, Comput. Methods Appl. Mech. Engrg. 194 (39–41) (2005) 4135–4195.

[3] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Netw. 2 (5) (1989) 359–366.

[4] I.E. Lagaris, A. Likas, D.I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE Trans. Neural Netw. 9 (5) (1998) 987–1000.

[5] I.E. Lagaris, A.C. Likas, D.G. Papageorgiou, Neural-network methods for boundary value problems with irregular boundaries, IEEE Trans. Neural Netw. 11 (5) (2000) 1041–1049.

[6] X. Huan, Y.M. Marzouk, Simulation-based optimal Bayesian experimental design for nonlinear systems, J. Comput. Phys. 232 (1) (2013) 288–317.

[7] J. Han, A. Jentzen, E. Weinan, Solving high-dimensional partial differential equations using deep learning, Proc. Natl. Acad. Sci. 115 (34) (2018) 8505–8510.

[8] J. Zhang, M.D. Shields, The effect of prior probabilities on quantification and propagation of imprecise probabilities resulting from small datasets, Comput. Methods Appl. Mech. Engrg. 334 (2018) 483–506.

[9] 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.

[10] Y. Bar-Sinai, S. Hoyer, J. Hickey, M.P. Brenner, Learning data-driven discretizations for partial differential equations, Proc. Natl. Acad. Sci. 116 (31) (2019) 15344–15349.

[11] J. Zhang, M.D. Shields, Efficient Monte Carlo resampling for probability measure changes from Bayesian updating, Probab. Eng. Mech. 55 (2019) 54–66.

[12] Y. Zhu, N. Zabaras, BayesIan deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification, J. Comput. Phys. 366 (2018) 415–447.

[13] S.L. Brunton, B.R. Noack, P. Koumoutsakos, Machine learning for fluid mechanics, Annu. Rev. Fluid Mech. 52 (2020) 477–508.

[14] A.A. Gorodetsky, J.D. Jakeman, G. Geraci, M.S. Eldred, MFNets: MUlti-fidelity data-driven networks for bayesian learning and prediction, Int. J. Uncertain. Quantif. 10 (6) (2020).

[15] Z. Wang, X. Huan, K. Garikipati, Variational system identification of the partial differential equations governing microstructure evolution in materials: Inference over sparse and spatially unrelated data, Comput. Methods Appl. Mech. Engrg. 377 (2021) 113706.

[16] M. Raissi, Deep hidden physics models: Deep learning of nonlinear partial differential equations, J. Mach. Learn. Res. 19 (1) (2018) 932–955.

[17] M. Raissi, Z. Wang, M.S. Triantafyllou, G.E. Karniadakis, Deep learning of vortex-induced vibrations, J. Fluid Mech. 861 (2019) 119–137.
[18] M. Raissi, A. Yazdani, G.E. Karniadakis, Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations, Science 367 (6481) (2020) 1026–1030.
[19] T. Qin, K. Wu, D. Xiu, Data driven governing equations approximation using deep neural networks, J. Comput. Phys. 395 (2019) 620–635.
[20] E. Samaniego, C. Anitescu, S. Goswami, V.M. Nguyen-Thanh, H. Guo, K. Hamdia, X. Zhuang, T. Rabczuk, An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementation and applications, Comput. Methods Appl. Mech. Engrg. 362 (2020) 112790.
[21] L. Sun, J.-X. Wang, Physics-constrained bayesian neural network for fluid flow reconstruction with sparse and noisy data, Theor. Appl. Mech. Lett. 10 (3) (2020) 161–169.
[22] C. Rao, H. Sun, Y. Liu, Physics-informed deep learning for incompressible laminar flows, Theor. Appl. Mech. Lett. 10 (3) (2020) 207–212.
[23] L. Sun, H. Gao, S. Pan, J.-X. Wang, Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data, Comput. Methods Appl. Mech. Engrg. 361 (2020) 112732.
[24] C. Rao, H. Sun, Y. Liu, Physics-informed deep learning for computational elastodynamics without labeled data, J. Eng. Mech. 147 (8) (2021) 04021043.
[25] X. Yang, S. Zafar, J.-X. Wang, H. Xiao, Predictive large-eddy-simulation wall modeling via physics-informed neural networks, Phys. Rev. Fluids 4 (3) (2019) 034602.
[26] X. Jin, S. Cai, H. Li, G.E. Karniadakis, NSFnets (Navier-Stokes Flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations, J. Comput. Phys. 426 (2021) 109951.
[27] Z. Mao, A.D. Jagtap, G.E. Karniadakis, Physics-informed neural networks for high-speed flows, Comput. Methods Appl. Mech. Engrg. 360 (2020) 112789.
[28] H. Wessels, C. Weißenfels, P. Wriggers, The neural particle method–an updated Lagrangian physics informed neural network for computational fluid dynamics, Comput. Methods Appl. Mech. Engrg. 368 (2020) 113127.
[29] E. Haghighat, M. Raissi, A. Moure, H. Gomez, R. Juanes, A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics, Comput. Methods Appl. Mech. Engrg. 379 (2021) 113741.
[30] D. Zhang, L. Lu, L. Guo, G.E. Karniadakis, Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems, J. Comput. Phys. 397 (2019) 108850.
[31] D. Zhang, L. Guo, G.E. Karniadakis, Learning in modal space: Solving time-dependent stochastic PDEs using physics-informed neural networks, SIAM J. Sci. Comput. 42 (2) (2020) A639–A665.
[32] G. Kissas, Y. Yang, E. Hwuang, W.R. Witschey, J.A. Detre, P. Perdikaris, Machine learning in cardiovascular flows modeling: Predicting arterial blood pressure from non-invasive 4D flow MRI data using physics-informed neural networks, Comput. Methods Appl. Mech. Engrg. 358 (2020) 112623.
[33] S. Cai, H. Li, F. Zheng, F. Kong, M. Dao, G.E. Karniadakis, S. Suresh, Artificial intelligence velocimetry and microaneurysm-on-a-chip for three-dimensional analysis of blood flow in physiology and disease, Proc. Natl. Acad. Sci. 118 (13) (2021).
[34] K. Shukla, P.C. Di Leoni, J. Blackshire, D. Sparkman, G.E. Karniadakis, Physics-informed neural network for ultrasound nondestructive quantification of surface breaking cracks, J. Nondestruct. Eval. 39 (3) (2020) 1–20.
[35] M. Yin, X. Zheng, J.D. Humphrey, G.E. Karniadakis, Non-invasive inference of thrombus material properties with physics-informed neural networks, Comput. Methods Appl. Mech. Engrg. 375 (2021) 113603.
[36] L. Lu, M. Dao, P. Kumar, U. Ramamurty, G.E. Karniadakis, S. Suresh, Extraction of mechanical properties of materials through deep learning from instrumented indentation, Proc. Natl. Acad. Sci. 117 (13) (2020) 7052–7062.
[37] R. Zhang, Y. Liu, H. Sun, Physics-guided convolutional neural network (PhyCNN) for data-driven seismic response modeling, Eng. Struct. 215 (2020) 110704.
[38] R. Zhang, Y. Liu, H. Sun, Physics-informed multi-LSTM networks for metamodeling of nonlinear structures, Comput. Methods Appl. Mech. Engrg. 369 (2020) 113226.
[39] Q. He, J.-S. Chen, A physics-constrained data-driven approach based on locally convex reconstruction for noisy database, Comput. Methods Appl. Mech. Engrg. 363 (2020) 112791.
[40] Z. Chen, Y. Liu, H. Sun, Physics-informed learning of governing equations from scarce data, 2020, arXiv arXiv:2005.03448.
[41] 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. (2020) 110079.
[42] K. Bhattacharya, B. Hosseini, N.B. Kovachki, A.M. Stuart, Model reduction and neural networks for parametric pdes, 2020, arXiv preprint arXiv:2005.03180.
[43] L. Lu, X. Meng, Z. Mao, G.E. Karniadakis, DeepXDE: A Deep learning library for solving differential equations, SIAM Rev. 63 (1) (2021) 208–228.
[44] L. Lu, P. Jin, G. Pang, Z. Zhang, G.E. Karniadakis, Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators, Nat. Mach. Intell. 3 (3) (2021) 218–229.
[45] R.G. Patel, N.A. Trask, M.A. Wood, E.C. Cyr, A physics-informed operator regression framework for extracting data-driven continuum models, Comput. Methods Appl. Mech. Engrg. 373 (2021) 113500.
[46] N. Winovich, K. Ramani, G. Lin, ConvPDE-UQ: COnvolutional neural networks with quantified uncertainty for heterogeneous elliptic partial differential equations on varied domains, J. Comput. Phys. 394 (2019) 263–279.
[47] N. Geneva, N. Zabaras, Modeling the dynamics of PDE systems with physics-constrained deep auto-regressive networks, J. Comput. Phys. 403 (2020) 109056.

[48] R. Ranade, C. Hill, J. Pathak, DiscretizationNet: A Machine-learning based solver for Navier–Stokes equations using finite volume discretization, Comput. Methods Appl. Mech. Engrg. 378 (2021) 113722.
[49] S. Bhatnagar, Y. Afshar, S. Pan, K. Duraisamy, S. Kaushik, Prediction of aerodynamic flow fields using convolutional neural networks, Comput. Mech. 64 (2) (2019) 525–545.
[50] 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.
[51] W.E. Sorteberg, S. Garasto, A.S. Pouplin, C.D. Cantwell, A.A. Bharath, Approximating the solution to wave propagation using deep neural networks, 2018, arXiv preprint arXiv:1812.01609.
[52] Z. Long, Y. Lu, X. Ma, B. Dong, PDE-net: Learning PDEs from data, in: International Conference on Machine Learning, PMLR, 2018, pp. 3208–3216.
[53] N. Geneva, N. Zabaras, Transformers for modeling physical systems, 2020, arXiv preprint arXiv:2010.03957.
[54] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, A. Anandkumar, Fourier neural operator for parametric partial differential equations, 2020, arXiv preprint arXiv:2010.08895.
[55] S. Seo, C. Meng, Y. Liu, Physics-aware difference graph networks for sparsely-observed dynamics, in: International Conference on Learning Representations, 2019.
[56] N. Trask, R.G. Patel, B.J. Gross, P.J. Atzberger, GMLS-Nets: A framework for learning from unstructured data, 2019, arXiv preprint arXiv:1909.05371.
[57] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, A. Anandkumar, Neural operator: Graph kernel network for partial differential equations, 2020, arXiv preprint arXiv:2003.03485.
[58] F.d.A. Belbute-Peres, T. Economon, Z. Kolter, Combining differentiable PDE solvers and graph neural networks for fluid flow prediction, in: International Conference on Machine Learning, PMLR, 2020, pp. 2402–2411.
[59] T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, P.W. Battaglia, Learning mesh-based simulation with graph networks, 2020, arXiv preprint arXiv:2010.03409.
[60] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, W.-c. Woo, Convolutional LSTM network: A machine learning approach for precipitation nowcasting, Adv. Neural Inf. Process. Syst. 28 (2015) 802–810.
[61] H. Schaeffer, R. Caflisch, C.D. Hauck, S. Osher, Sparse dynamics for partial differential equations, Proc. Natl. Acad. Sci. 110 (17) (2013) 6634–6639.
[62] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[63] A. Graves, Generating sequences with recurrent neural networks, 2013, arXiv preprint arXiv:1308.0850.
[64] I. Sutskever, O. Vinyals, Q.V. Le, Sequence to sequence learning with neural networks, 2014, arXiv preprint arXiv:1409.3215.
[65] M.I. Jordan, Serial order: A parallel distributed processing approach, in: Advances in Psychology, Vol. 121, Elsevier, 1997, pp. 471–495.
[66] T.Q. Chen, Y. Rubanova, J. Bettencourt, D. Duvenaud, Neural ordinary differential equations, in: NeurIPS, 2018, pp. 6572–6583.
[67] W. Shi, J. Caballero, F. Huszár, J. Totz, A.P. Aitken, R. Bishop, D. Rueckert, Z. Wang, Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1874–1883.
[68] Q. Shan, Z. Li, J. Jia, C.-K. Tang, Fast image/video upsampling, ACM Trans. Graph. 27 (5) (2008) 1–7.
[69] A. Odena, V. Dumoulin, C. Olah, Deconvolution and checkerboard artifacts, Distill 1 (10) (2016) e3.
[70] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: International Conference on Machine Learning, PMLR, 2015, pp. 448–456.
[71] J. Yu, Y. Fan, J. Yang, N. Xu, Z. Wang, X. Wang, T. Huang, Wide activation for efficient and accurate image super-resolution, 2018, arXiv preprint arXiv:1808.08718.
[72] T. Salimans, D.P. Kingma, Weight normalization: A simple reparameterization to accelerate training of deep neural networks, 2016, arXiv preprint arXiv:1602.07868.
[73] C. Rao, H. Sun, Y. Liu, Embedding physics to learn spatiotemporal dynamics from sparse data, 2021, arXiv preprint arXiv:2106.04781.
[74] C. Rao, H. Sun, Y. Liu, Hard encoding of physics for learning spatiotemporal dynamics, 2021, arXiv preprint arXiv:2105.00557.
[75] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in pytorch, 2017.
[76] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.
[77] D.C. Liu, J. Nocedal, On the limited memory BFGS method for large scale optimization, Math. Program. 45 (1) (1989) 503–528.
[78] G. Huang, Z. Liu, L. Van Der Maaten, K.Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4700–4708.
[79] S.H. Rudy, S.L. Brunton, J.L. Proctor, J.N. Kutz, Data-driven discovery of partial differential equations, Sci. Adv. 3 (4) (2017) e1602614.
[80] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, 2016, arXiv preprint arXiv:1609.02907.
[81] X. Bresson, T. Laurent, Residual gated graph convnets, 2017, arXiv preprint arXiv:1711.07553.
[82] M.M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, P. Vandergheynst, Geometric deep learning: going beyond euclidean data, IEEE Signal Process. Mag. 34 (4) (2017) 18–42.
[83] P.W. Battaglia, J.B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, et al., Relational inductive biases, deep learning, and graph networks, 2018, arXiv preprint arXiv:1806.01261.
[84] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, 2016, arXiv preprint arXiv:1606.09375.
[85] E. Weinan, B. Yu, The deep ritz method: a deep learning-based numerical algorithm for solving variational problems, Commun. Math. Stat. 6 (1) (2018) 1–12.
[86] S. Goswami, M. Yin, Y. Yu, G. Karniadakis, A physics-informed variational DeepONet for predicting the crack path in brittle materials, 2021, arXiv preprint arXiv:2108.06905.