International Journal of Applied Science; Vol. 8, No. 4; 2025
ISSN 2576-7240 E-ISSN 2576-7259
https://doi.org/10.30560/ijas.v8n4p82

A Hybrid CNN - TKAN Architecture for Solving Partial Differential Equations

Xuyang Zhang[1] & Jin Su[1]

[1] School of Science, Xi'an Polytechnic University, China

Correspondence: Xuyang Zhang, School of Science, Xi'an Polytechnic University, Xi'an, 710048, China. E-mail: 15029381916@163.com

Received: November 22, 2025; Accepted: December 1, 2025; Published: December 2, 2025

Abstract

This paper presents a Physics-Informed Convolutional Temporal Kolmogorov-Arnold Network (PhyCTKAN) to address the challenge of solving spatiotemporal partial differential equations (PDEs). The proposed method enhances the Temporal Kolmogorov - Arnold Network (TKAN) through convolution operations and integrates an encoder - decoder architecture for dimensionality reduction and spatial domain mapping. Solutions are iteratively computed at each time step via an autoregressive process based on Euler discretization. Experimental results on nonlinear spatiotemporal systems demonstrate that PhyCTKAN achieves superior long-term prediction accuracy compared to the PhyCRNet model, while maintaining high computational precision even with increased time step sizes.

Keywords: Spatiotemporal PDEs, Autoregressive Process, Physics-Informed learning, Encoder-Decoder

1. Introduction

Solving spatiotemporal PDEs, which are central to scientific modeling, remains challenging in practice. While conventional numerical methods offer wide applicability, their computational cost is often prohibitive [1,2]. Consequently, neural network-based solvers have been developed as a promising alternative [3]. The evolution of hardware like GPUs has further accelerated this paradigm shift, enabling deep neural networks to achieve groundbreaking progress in PDEs solutions.

While emerging techniques facilitate the development of surrogate models and eliminate repetitive computations in conventional PDE solvers, they exhibit two major limitations in physical system modeling: (i) high computational cost due to grid-dependent discretization, and (ii) reduced model reliability in data-scarce or multi-physics coupling scenarios. To address these issues, Raissi et al. proposed Physics-Informed Neural Networks (PINNs) [7], which systematically incorporate physical laws—such as conservation of momentum and energy—into the loss function or network architecture. By integrating observational data with physical constraints, PINNs enable efficient and physics-consistent modeling. Since their introduction, PINNs have demonstrated significant value in simulating systems governed by partial differential equations, including solid mechanics [8, 9] and fluid dynamics [10, 11], as well as in interdisciplinary applications such as blood flow modeling [12] and non-invasive parameter estimation [13]. Concurrently, the Autoregressive DenseED (AR-DenseED) approach proposed to effectively alleviate key limitations of PINNs—namely, high computational costs and instability in long-term integration [14]. Ren et al. developed PhyCRNet, a model combining CNNs and RNNs (e.g., ConvLSTMs), which significantly improved performance on high-dimensional spatiotemporal PDEs [15].

Inspired by PhyCRNet, this paper proposes the PhyCTKAN framework to solve spatiotemporal PDEs with unlabeled data. The framework employs an encoder-decoder architecture for autoregressive prediction and incorporates the TKAN proposed by Genet et al. [16] to enhance the robustness of multi-step forecasting and mitigate gradient issues.

This paper is structured as follows. Section 2 introduces the spatiotemporal PDEs to be solved and residual connection formula. Section 3 presents the design of the temporal integration module – ConvTKAN、the network architecture PhyCTKAN, and the corresponding numerical experiments. Finally, Section 4 provides a concluding summary.

2. Problem Statement

This paper considers a class of parameterized two-dimensional nonlinear spatiotemporal PDEs, with the following general form:

$$\boldsymbol{u}_t + \mathcal{F}(\boldsymbol{u}, \boldsymbol{u}_t, \nabla_x \boldsymbol{u}, \nabla_x^2 \boldsymbol{u}, \nabla_x \boldsymbol{u} \cdot \boldsymbol{u}, \cdots; \boldsymbol{\lambda}) = 0. \tag{1}$$

Where $\boldsymbol{u}(\boldsymbol{x}, t) \in R^2$ denotes the solution variable in the temporal domain $t \in [0, T]$ and the physical domain Ω; $\boldsymbol{u}_t$ is the first-order time derivative term; ∇ represents the gradient operator with respect to $\boldsymbol{x}$; ∇^2 represents the Laplacian operator; $\mathcal{F}(\cdot)$ is the nonlinear functional parameterized by $\boldsymbol{\lambda}$; I/BCs have the form as $\mathcal{I}(\boldsymbol{u}, \boldsymbol{u}_t, \cdots; t = 0, \boldsymbol{u} \in \Omega) = 0$ and $\mathcal{B}(\boldsymbol{u}, \nabla_x \boldsymbol{u}, \cdots; \boldsymbol{x} \in \partial\Omega) = 0$, where $\partial\Omega$ denotes the boundary of the spatial domain.

To facilitate the temporal evolution in solving the spatiotemporal PDEs, we employ a residual connection with the following iterative formula:

$$\boldsymbol{u}_{i+1} = \boldsymbol{u}_i + \Delta t \cdot \mathcal{NN}(\boldsymbol{u}_i; \boldsymbol{\theta}). \tag{2}$$

Notably, the network $\mathcal{NN}(\boldsymbol{u}_i; \boldsymbol{\theta})$ does not learn a direct mapping from $\boldsymbol{u}_i$ to $\boldsymbol{u}_{i+1}$, but rather learns the residual mapping $\Delta t \cdot \mathcal{NN}(\boldsymbol{u}_i; \boldsymbol{\theta})$.

3. Methodology

This section begins by detailing the architecture of the ConvTKAN unit. Subsequently, we introduce a coupled framework named the Physics-informed Convolutional Temporal Kolmogorov-Arnold Network (PhyCTKAN). Finally, the effectiveness of the proposed model is validated through numerical experiments.

3.1 ConvTKAN

The TKAN model, proposed by Remi Genet and Hugo Inzirillo, introduces two key enhancements to the static KAN. First, it extends the architecture into a recurrent one by replacing the linear weights in recurrent neural networks with learnable univariate functions $\varphi(x)$.

$$\varphi(x) = \omega\left(\frac{x}{1 + e^{-x}} + \sum\nolimits_i c_i B_i(x)\right).$$

Here, $B_i(x)$ is a B-spline curve; ω and c_i are parameters to be learned by the network. Second, inspired by the LSTM's gating mechanism, it incorporates forget, input, and output gates, thereby significantly improving its capability to model long-term dependencies. To address the lack of spatial modeling capability in TKAN, we replace its fully connected operations with convolutional operations to construct the ConvTKAN unit. This unit employs convolution to process the input and state vectors, enabling it to effectively capture spatial features while retaining TKAN's advantage in modeling long-term dependencies.

Let $\mathbf{X}_t$ denote the input tensor; $\mathbf{h}_t$、$\mathbf{C}_t$ and $\tilde{\mathbf{h}}_t$ represent the hidden state, cell state, and hidden state of sub-layer respectively. Then, the mathematical formulation of the ConvTKAN unit is presented as follows:

$$\mathbf{i}_t = \sigma(\mathbf{W}_i * [\mathbf{X}_t, \mathbf{h}_{t-1}] + \mathbf{b}_i), \quad \mathbf{f}_t = \sigma(\mathbf{W}_f * [\mathbf{X}_t, \mathbf{h}_{t-1}] + \mathbf{b}_f),$$

$$\tilde{\mathbf{C}}_{t-1} = \tanh(\mathbf{W}_c * [\mathbf{X}_t, \mathbf{h}_{t-1}] + \mathbf{b}_c), \quad \mathbf{C}_t = \mathbf{f}_t \odot \mathbf{C}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{C}}_{t-1},$$

$$\mathbf{s}_{l,t} = \mathbf{W}_{l,\tilde{x}} * \mathbf{X}_t + \mathbf{W}_{l,\tilde{h}} * \tilde{\mathbf{h}}_{l,t-1}, \quad \tilde{\mathbf{o}}_{l,t} = \Phi_l(\mathbf{s}_{l,t}),$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o * \mathbf{r}_t + \mathbf{b}_o), \quad \mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{C}_t).$$

Where $\mathbf{r}_t = Concat[\tilde{\mathbf{o}}_{1,t}, \tilde{\mathbf{o}}_{2,t}, \cdots, \tilde{\mathbf{o}}_{l,t}]$; $\mathbf{s}_{l,t}$ is the combination of the current input $\mathbf{X}_t$ and the previous hidden sub-state $\tilde{\mathbf{h}}_{l,t-1}$; the sub-state $\tilde{\mathbf{h}}_{l,t-1}$ is determined by the state $\tilde{\mathbf{h}}_{l,t-2}$ at the previous moment and the output $\tilde{\mathbf{o}}_{l,t-1}$ of KANs.

$$\tilde{\mathbf{h}}_{l,t-1} = \mathbf{W}_{hh} * \tilde{\mathbf{h}}_{l,t-2} + \mathbf{W}_{hz} * \tilde{\mathbf{o}}_{l,t-1}.$$

$\tilde{\mathbf{o}}_{l,t-1}$ denotes the output of the l-th KAN layer;" * "is a convolution operation; $\mathbf{W}_{l,\tilde{x}}, \mathbf{W}_{l,\tilde{h}}, \mathbf{W}_{hh}, \mathbf{W}_{hz}$ 、 are weight matrices—all parameters to be learned by the network.

3.2 Neural Network Framework

This part outlines the PhyCTKAN architecture, which enhances long-term forecasting performance and efficiency. The model consists of four key components: an encoder-decoder module, an autoregressive process, a time propagator - ConvTKAN, and a residual connection mechanism, as illustrated in Fig. 1.

The encoder employs convolutional layers, each followed by a ReLU activation, to project high-dimensional inputs into a lower-dimensional space, preserving essential features while reducing computational complexity. The resulting latent representation is then passed to a ConvTKAN temporal propagator for temporal evolution, which operates under zero-initialized conditions. This design enables accurate feature extraction and efficient multi-step prediction. Finally, the decoder utilizes sub-pixel convolution modules to up sample the processed data back to its original dimensions, and the prediction for the next time step is obtained via a residual connection formulation (Formula 2).

$$\boldsymbol{u}_{i+1} = \boldsymbol{u}_i + \Delta t \cdot \mathcal{NN}(\boldsymbol{u}_i; \boldsymbol{\theta}).$$

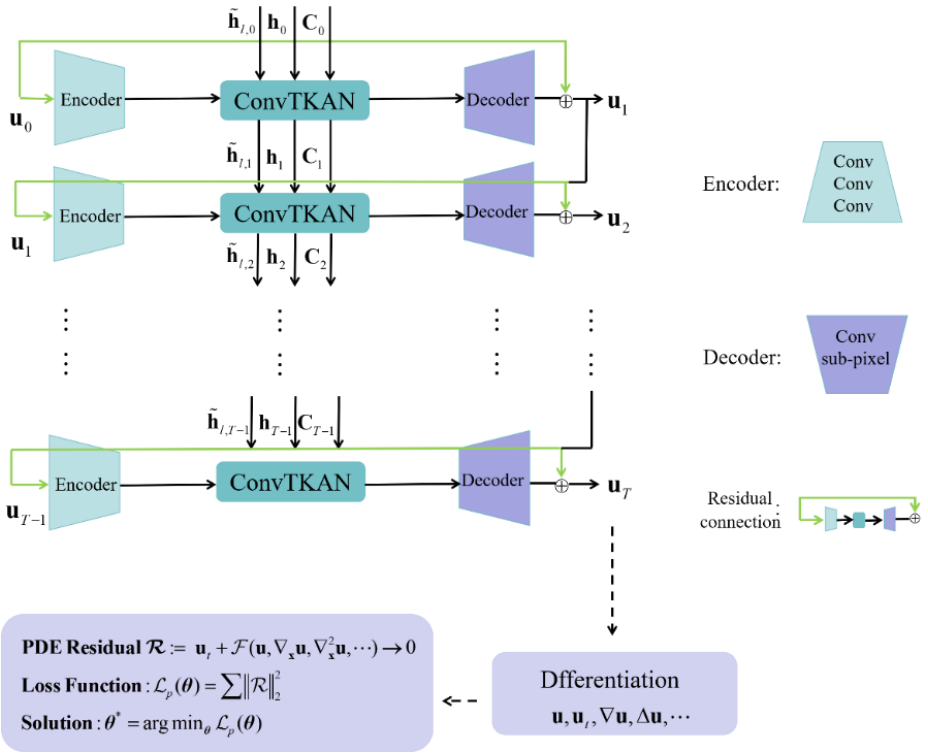

Figure 1. The network architecture of PhyCTKAN. The variables $\mathbf{C}$ 、 $\mathbf{h}$ and $\tilde{\mathbf{h}}$ are cell state、hidden state and sub-layer state of ConvTKAN cell respectively. $\mathbf{u}_0$ is the known state variable (i.e. Initial condition)

Where $\mathcal{NN}(\boldsymbol{u}_i; \boldsymbol{\theta})$ denotes the trained network operator and Δt is the time step. The output state $\boldsymbol{u}_{i+1}$ serves as the input for the subsequent time step, thereby forming an autoregressive process.

3.3 Numerical Experiments

The initial conditions are encoded as input to the neural network, and the solution at subsequent time steps is inferred through an autoregressive process. For periodic boundary conditions, a specialized circular padding method based on convolution is employed. The underlying idea of circular padding is to "wrap" the boundary values of the input data to the opposite side, thereby simulating periodic boundaries. The principle of this padding mechanism is illustrated in Fig. 2.

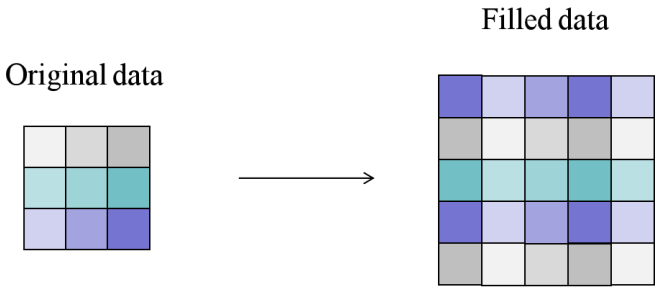

Figure 2. Periodic boundary filling principle

So, we only need to construct a loss function based on the governing PDEs, which is formally defined as:

$$\mathcal{R}(\boldsymbol{x}, t; \boldsymbol{\theta}) := \boldsymbol{u}_t^{\theta} + \mathcal{F}(\boldsymbol{u}^{\theta}, \nabla_x \boldsymbol{u}^{\theta}, \nabla_x^2 \boldsymbol{u}^{\theta}, \nabla_x \boldsymbol{u}^{\theta} \cdot \boldsymbol{u}^{\theta}, \cdots; \boldsymbol{\lambda}).$$

The residual, represented by the left-hand side of Equation (1), includes all physical variables learned by the network. The network parameters are further optimized by minimizing this loss function. For clarity, a two-dimensional PDE system is used as an illustrative example:

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

n and m denote the number of discrete points along the y-axis and x-axis, respectively, while T corresponds to the total number of time steps, $\| \cdot \|$ is the Euclidean norm.
Based on the above architecture design, the implementation process is detailed in Algorithm 1.

Algorithm 1: PhyCTKAN

1. Input $\boldsymbol{u}_0$:initial values of the equation; state variables $\{\boldsymbol{h}_0, \boldsymbol{C}_0, \tilde{\boldsymbol{h}}_{l,0}\}$
2. Output: $\{\boldsymbol{u}_0, \boldsymbol{u}_1, \cdots, \boldsymbol{u}_t\}$

   For i=0 to T:

   (1). $\boldsymbol{u}_i = \boldsymbol{u}_0$

   (2). the Encoder module extracts feature from $\boldsymbol{u}_i$

   (3). time propagator ConvTKAN updates memory state $\{\boldsymbol{h}_i, \boldsymbol{C}_i, \tilde{\boldsymbol{h}}_{l,i}\}$

   (4). the Decoder is used for up sampling $\rightarrow \mathcal{NN}(\boldsymbol{u}_i; \boldsymbol{\theta})$

   (5). $\boldsymbol{u}_{i+1} = \boldsymbol{u}_i + \delta t \cdot \mathcal{NN}(\boldsymbol{u}_i; \boldsymbol{\theta})$

   (6). $\boldsymbol{u}_0 = \boldsymbol{u}_{i+1}$

   Save solutions at all times: $[\boldsymbol{u}_0, \boldsymbol{u}_1, \cdots, \boldsymbol{u}_T]$
3. Calculating Derivatives: $\{\boldsymbol{u}_t, \nabla_x \boldsymbol{u}, \nabla_x^2 \boldsymbol{u}, \cdots\}$
4. PDEs Residual $\mathcal{R} := \boldsymbol{u}_t + \mathcal{F}(\boldsymbol{u}, \nabla_x \boldsymbol{u}, \nabla_x^2 \boldsymbol{u}, \nabla_x \boldsymbol{u} \cdot \boldsymbol{u}, \cdots; \boldsymbol{\lambda})$
5. Adam optimizer to minimize loss: $\mathcal{L}_p(\boldsymbol{\theta})$

We conducted numerical simulations on the Fitzhugh-Nagumo Equation (FN) and λ-ω RD Equation. In all our PDE case studies, the PhyCTKAN model follows a uniform architecture. The encoder consists of three convolutional layers with periodic padding, a 4×4 kernel, and a stride of 2. The encoded features are passed to a ConvTKAN layer, which uses a memory unit and a hidden state of 128 channels, along with sub-layer hidden states of 20 channels. This layer employs a 3×3 kernel with stride 1. The output is upsampled by a factor of 8 via a pixel-shuffling layer, while the final layer uses a larger 5×5 kernel (stride 1) to capture high-resolution spatial features.

3.3.1 Fitzhugh-Nagumo Equation

The FitzHugh-Nagumo (FN) equations are classic mathematical models in biology, widely used in nonlinear dynamics research. They effectively simulate the generation and propagation of neuronal action potentials and are given by:

$$\frac{\partial u}{\partial t} = \gamma_u \Delta u + u - u^3 - v + \alpha,$$
$$\frac{\partial v}{\partial t} = \gamma_v \Delta v + \beta(u - v).$$

The spatial diffusion coefficient $\gamma_u = 10, \gamma_v = 25$ represents the degree of diffusion of membrane potential; the external stimulus $\alpha = 0.01$ refers to the influence of current or chemical signals; and the coupling coefficient $\beta = 0.25$ describes the interaction between membrane potential and recovery variables. We are using the discrete learning framework PhyCTKAN to solve this equation. And the time step $\Delta t = 0.04$ and the spatial domain size as $x, y \in [0,1] \times [0,1]$. The learning rate is initialized to 1×10^{-4} and decayed by a factor of 0.97 every 100 steps using a scheduler. The predicted solutions generated by PhyCTKAN and the comparative model (PhyCRNet), along with their corresponding error maps, are presented in Figure 3 and Figure 4.

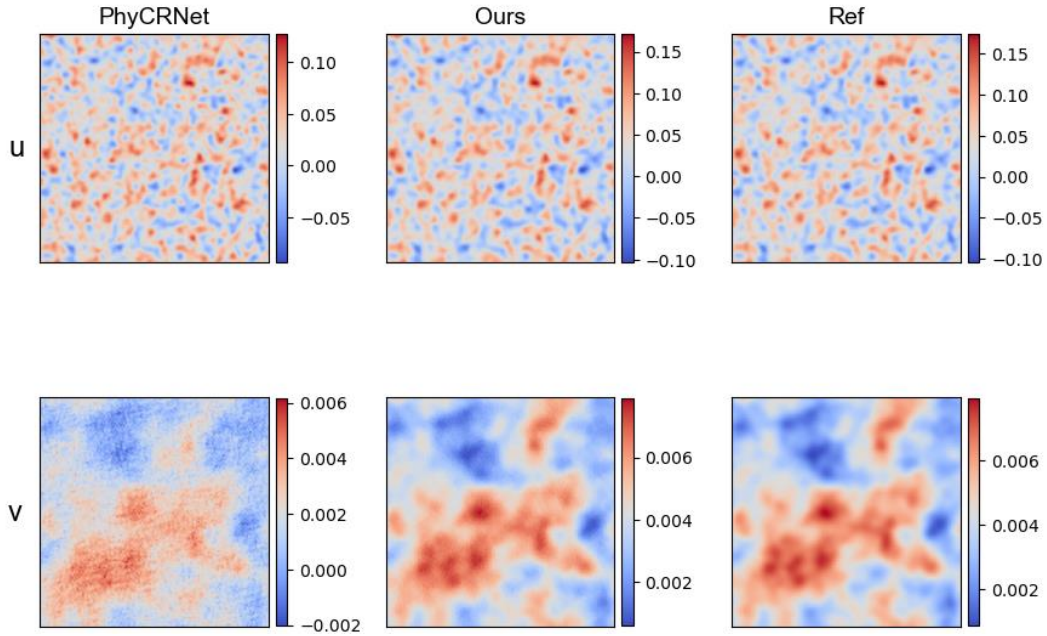

Figure 3. Comparison of solution accuracy between PhyCRNet and Our model (PhyCTKAN) for the FN Equation. Left: Predicted solution of PhyCRNet model; Middle: Predicted solution of Our; Right: Reference solution.

Our model (PhyCTKAN) shows close with the reference solution in Figure 3, unlike PhyCRNet. This superior accuracy is quantified by lower and less fluctuating errors in Figure 4 and Table 1, demonstrating ConvTKAN's advantages over ConvLSTM in precision, physical consistency, and long-term stability for complex PDEs.

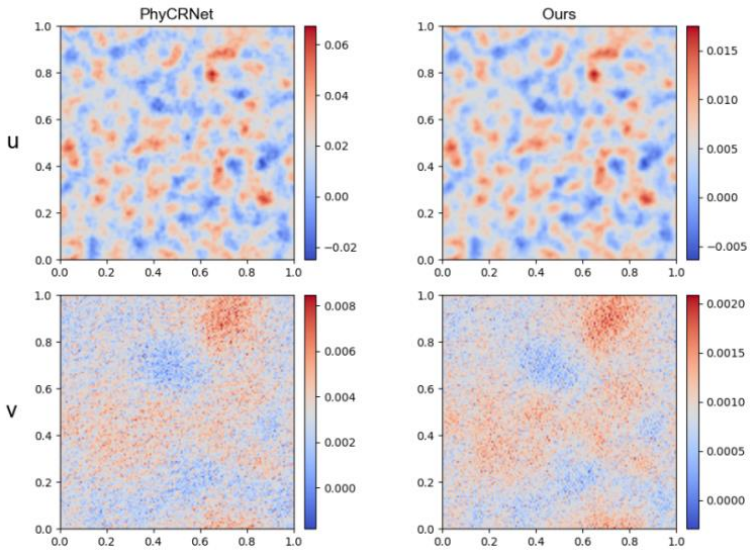

Figure 4. Comparison of error plot between PhyCRNet and Our model in the FN Equation. The first column represents the global error distribution plot between the baseline model predicted values and the actual values, while the second column shows the error distribution plot of Our model.

Table 1. Comparison table of quantitative indicators RMSE for two models at different time points for the FN equation.

u_RMSE	PhyCRNet	Ours	v_RMSE	PhyCRNet	Ours
T=0.08	0.0178	0.0019	-	0.0028	0.0004
T=0.24	0.0218	0.0058	-	0.0035	0.0010
T=0.36	0.0249	0.0090	-	0.0040	0.0015

As shown in Fig. 5, PhyCTKAN achieves faster convergence and a lower loss value, with a smooth, oscillation-free curve. This demonstrates its superior capability in feature capture and a more stable training process.

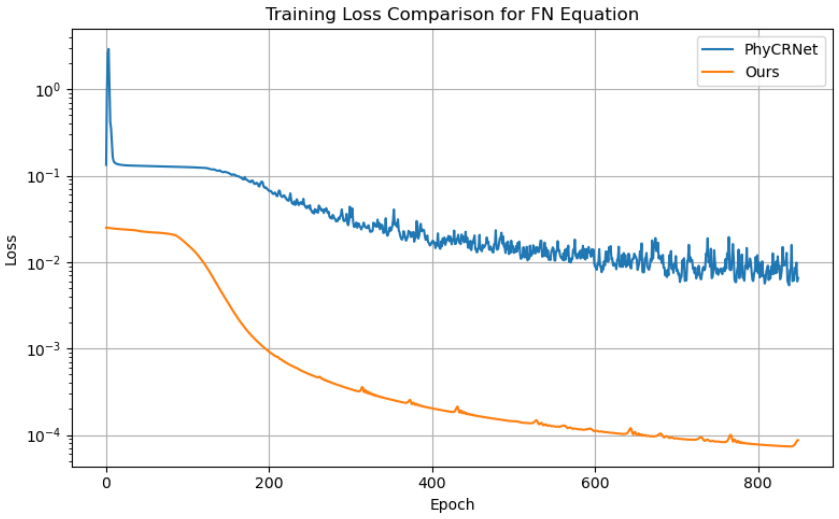

Figure 5. Comparison chart of training loss of the model.

3.3.2 Terzaghi's Consolidation Equation

The two-dimensional Terzaghi's Consolidation Equation is a key mathematical model that describes how saturated soil deforms and dissipates pore water pressure under external loads. It accounts for horizontal and vertical

deformation, as well as changes in pore water pressure over time and space. Using numerical methods to solve the equation allows simulation of pore water pressure variation and prediction of consolidation settlement. As a result, it provides an essential basis for foundation design and helps prevent damage from uneven settlement. The equation with periodic boundary conditions takes the following form:

$$\frac{\partial u}{\partial t} = C_v\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right),$$

$u(x, y, t)$ represents the excess pore-water pressure and is the key indicator of the consolidation process. The coefficient $C_v = 0.02$ reflects the speed of soil consolidation and is closely related to the permeability coefficient k, compression coefficient m_v, and unit weight of water γ_ω in the soil. The distribution of initial pore water pressure $u_0(x, y)$ in this equation is

$$u_0(x, y) = \sin\left(\frac{\pi x}{32}\right)\sin\left(\frac{\pi y}{32}\right).$$

The spatial domain as $x, y \in [0,1]\times[0,1]$. Now we use our model (PhyCTKAN) to solve the Terzaghi's CE with the time step fixed $\Delta t = 0.025$. From the prediction graph (Fig.6), it is clear that our model can accurately solve the Terzaghi's CE, verifying its effectiveness and applicability in solving such problem.

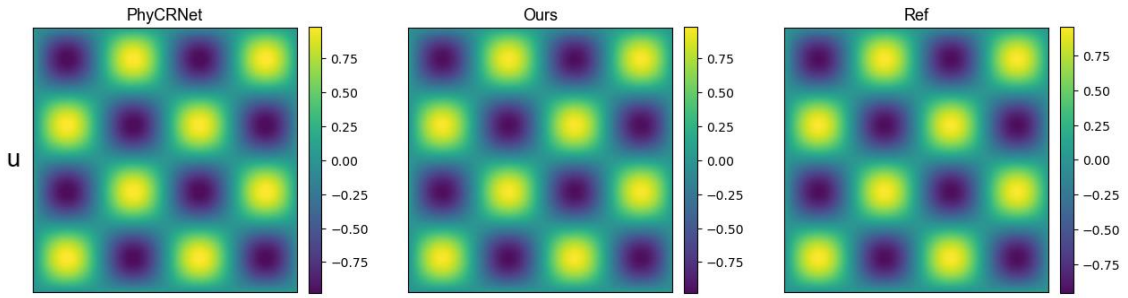

Figure 6. Comparison of solution accuracy between PhyCRNet and PhyCTKAN for the CE. Left: Predicted solution of the model PhyCRNet; Middle: Predicted solution of Our; Right: Reference solution

In addition, as can be seen from the error plot (Fig.7) and RMSE comparison table (Table 2), compared to the model PhyCRNet, our proposed method has smaller errors and is closer to the reference solution, exhibiting certain advantages in prediction accuracy.

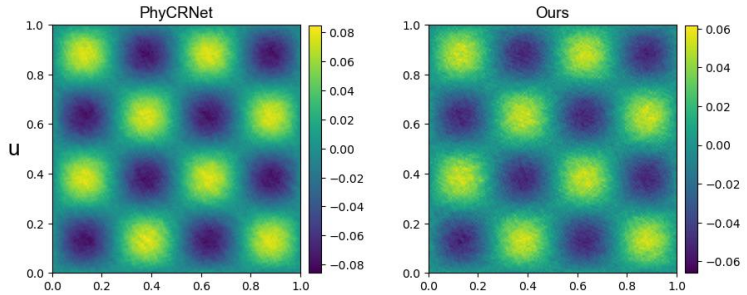

Figure 7. Comparison of error plot between PhyCRNet and Our in the CE. The first column represents the error distribution plot of PhyCRNet, while the second column represents the error distribution plot of our model.

Table 2. Comparison table of RMSE for two models at different time points for the CE.

u_RMSE	T=0.075	T=0.25	T=0.45
PhyCRNet	0.0182	0.0380	0.0450
Ours	0.0070	0.0189	0.0330

Furthermore, as shown in Fig 8, due to the introduction of nonlinear basis functions in the ConvTKAN module of our model, which enables the model to quickly extract information at different scales. Therefore, our proposed model exhibits markedly faster convergence than PhyCRNet. While PhyCRNet's loss curve declines slowly and exhibits noticeable fluctuations between epochs 400 and 800, the loss curve of ours remains considerably smoother.

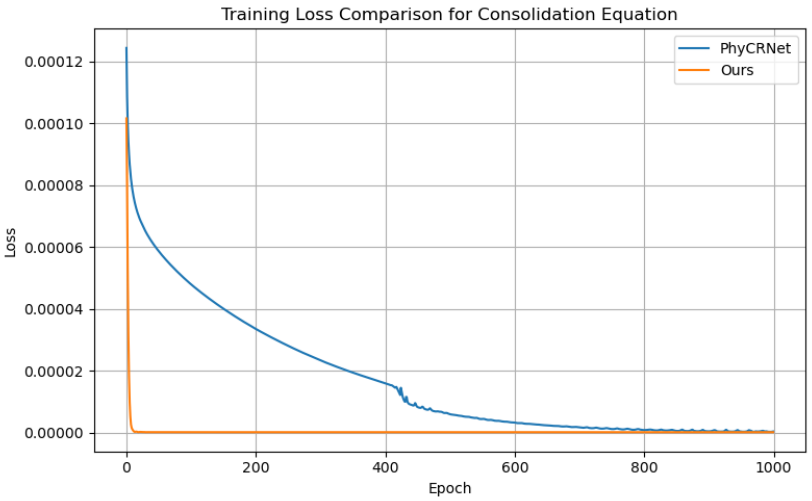

Figure 8. Comparison chart of training loss of the model

4. Conclusion

We present PhyCTKAN for solving spatiotemporal PDEs, a method that combines the local feature extraction of CNNs with the long-term modeling capacity of ConvTKAN. By explicitly encoding I/BCs and implementing the forward Euler method, the framework establishes robust input-output relationships and alleviates gradient vanishing. Experiments show PhyCTKAN attains superior accuracy and lower RMSE across various systems.

References

[1] Hughes, T. J. (2003). The finite element method: Linear static and dynamic finite element analysis. Courier Corporation.

[2] Moukalled, F., Mangani, L., & Darwish, M. (2015). The finite volume method. In The finite volume method in computational fluid dynamics: An advanced introduction with Open FOAM and MATLAB (pp. 103–135). Springer International Publishing. https://doi.org/10.1007/978-3-319-16874-6_5

[3] Dissanayake, M. G., & Phan-Thien, N. (1994). Neural-network-based approximations for solving partial differential equations. *Communications in Numerical Methods in Engineering, 10*(3), 195–201. https://doi.org/10.1002/cnm.1640100303

[4] Graves, A. (2012). Long short-term memory. In Supervised sequence labelling with recurrent neural networks (pp. 37–45). Springer. https://doi.org/10.1007/978-3-642-24797-2_4

[5] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 770–778). IEEE. https://doi.org/10.1109/CVPR.2016.90

[6] Scarselli, F., Gori, M., Tsoi, A. C., Buchner, M. H., & Monfardini, G. (2008). The graph neural network model. *IEEE Transactions on Neural Networks, 20*(1), 61–80. https://doi.org/10.1109/TNN.2008.2005605

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

[8] Rao, C., Sun, H., & Liu, Y. (2021). Physics-informed deep learning for computational elastodynamics without labeled data. *Journal of Engineering Mechanics, 147*(8), Article 04021043. https://doi.org/10.1061/(ASCE)EM.1943-7889.0001947

[9] Haghighat, E., Raissi, M., Moure, A., Gomez, H., & Juanes, R. (2021). A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics. *Computer Methods in Applied*

Mechanics and Engineering, 379, Article 113741. https://doi.org/10.1016/j.cma.2021.113741

[10] Sun, L., Gao, H., Pan, S., & Wang, J. X. (2020). Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Computer Methods in Applied Mechanics and Engineering, 361*, Article 112732. https://doi.org/10.1016/j.cma.2019.112732

[11] Wessels, H., Weißenfels, C., & Wriggers, P. (2020). The neural particle method—An updated Lagrangian physics informed neural network for computational fluid dynamics. *Computer Methods in Applied Mechanics and Engineering, 368*, Article 113127. https://doi.org/10.1016/j.cma.2020.113127

[12] Cai, S., Li, H., Zheng, F., Kong, F., Dao, M., Karniadakis, G. E., & Suresh, S. (2021). Artificial intelligence velocimetry and microaneurysm-on-a-chip for three-dimensional analysis of blood flow in physiology and disease. *Proceedings of the National Academy of Sciences, 118*(13), Article e2100697118. https://doi.org/10.1073/pnas.2100697118

[13] Yin, M., Zheng, X., Humphrey, J. D., & Karniadakis, G. E. (2021). Non-invasive inference of thrombus material properties with physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering, 375*, Article 113603. https://doi.org/10.1016/j.cma.2020.113603

[14] Geneva, N., & Zabaras, N. (2020). Modeling the dynamics of PDE systems with physics-constrained deep auto-regressive networks. *Journal of Computational Physics, 403*, Article 109056. https://doi.org/10.1016/j.jcp.2019.109056

[15] Ren, P., Rao, C., Liu, Y., Wang, J. X., & Sun, H. (2022). PhyCRNet: Physics-informed convolutional-recurrent network for solving spatiotemporal PDEs. *Computer Methods in Applied Mechanics and Engineering, 389*, Article 114399. https://doi.org/10.1016/j.cma.2021.114399

[16] Genet, R., & Inzirillo, H. (2024). TKAN: Temporal Kolmogorov-Arnold Networks. arXiv preprint arXiv:2405.07344v3. https://doi.org/10.2139/ssrn.4825654

[17] Wang, D., Yang, Y., & Ning, S. (2018). DeepSTCL: A deep spatio-temporal ConvLSTM for travel demand prediction. In 2018 International Joint Conference on Neural Networks (IJCNN) (pp. 1–8). IEEE. https://doi.org/10.1109/IJCNN.2018.8489530

[18] Liu, Z., Wang, Y., Vaidya, S., Liu, C., & Zhang, R. (2024). KAN: Kolmogorov-Arnold Networks. arXiv preprint arXiv:2404.19756. https://doi.org/10.1515/kant-2024-2032

[19] Somvanshi, S., Javed, S. A., Islam, M. M., Pandit, D., & Das, S. (2024). A survey on Kolmogorov-Arnold Network. ACM Computing Surveys. https://doi.org/10.1145/3743128

[20] Vaca-Rubio, C. J., Blanco, L., Pereira, R., & Caus, M. (2024). Kolmogorov-Arnold Networks (KANs) for time series analysis. arXiv preprint arXiv:2405.08790. https://doi.org/10.1109/GCWkshp64532.2024.11100692

[21] Shukla, K., Toscano, J. D., Wang, Z., Zou, Z., & Karniadakis, G. E. (2024). A comprehensive and FAIR comparison between MLP and KAN representations for differential equations and operator networks. Computer *Methods in Applied Mechanics and Engineering, 431*, Article 117290. https://doi.org/10.1016/j.cma.2024.117290

[22] Salimans, T., & Kingma, D. P. (2016). Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in Neural Information Processing Systems, 29*, 903–911.

[23] Aghaei, A. A. (2024). RKAN: Rational Kolmogorov-Arnold Networks. arXiv preprint arXiv:2406.14495.