ARTICLE

Check for updates

Multi-resolution partial differential equations preserved learning framework for spatiotemporal dynamics

Xin-Yang Liu [1], Min Zhu[2], Lu Lu [2], Hao Sun [3,4] & Jian-Xun Wang [1,5,6 ✉]

Traditional data-driven deep learning models often struggle with high training costs, error accumulation, and poor generalizability in complex physical processes. Physics-informed deep learning (PiDL) addresses these challenges by incorporating physical principles into the model. Most PiDL approaches regularize training by embedding governing equations into the loss function, yet this depends heavily on extensive hyperparameter tuning to weigh each loss term. To this end, we propose to leverage physics prior knowledge by "baking" the discretized governing equations into the neural network architecture via the connection between the partial differential equations (PDE) operators and network structures, resulting in a PDE-preserved neural network (PPNN). This method, embedding discretized PDEs through convolutional residual networks in a multi-resolution setting, largely improves the generalizability and long-term prediction accuracy, outperforming conventional black-box models. The effectiveness and merit of the proposed methods have been demonstrated across various spatiotemporal dynamical systems governed by spatiotemporal PDEs, including reaction-diffusion, Burgers', and Navier-Stokes equations.

[1] Department of Aerospace and Mechanical Engineering, University of Notre Dame, Notre Dame, IN, USA. [2] Department of Statistics and Data Science, Yale University, New Haven, CT, USA. [3] Gaoling School of Artificial Intelligence, Renmin University of China, Beijing, China. [4] Beijing Key Laboratory of Big Data Management and Analysis Methods, Beijing, China. [5] Lucy Family Institute for Data & Society, University of Notre Dame, Notre Dame, IN, USA. [6] Center for Sustainable Energy (ND Energy), University of Notre Dame, Notre Dame, IN, USA. ✉email: jwang33@nd.edu

Computational modeling and simulation capabilities play an essential role in understanding, predicting, and controlling various physical processes (e.g., turbulence, heat-flow coupling, and fluid-structure interaction), which often exhibit complex spatiotemporal dynamics. These physical phenomena are usually governed by partial differential equations (PDEs) and can be simulated by solving these PDEs numerically based on, e.g., finite difference (FD), finite volume (FV), finite element (FE), or spectral methods. However, predictive modeling of complex spatiotemporal dynamics using traditional numerical methods can be significantly challenging in many practical scenarios: (1) governing equations for complex systems might not be fully known due to a lack of complete understanding of the underlying physics, for which a first-principled numerical solver cannot be built; (2) conventional numerical simulations are usually time-consuming, making it infeasible for many applications that require many repeated model queries, e.g., optimization design, inverse problems, and uncertainty quantification (UQ), attracting increasing attention in scientific discovery and engineering practice.

Recent advances in scientific machine learning (SciML) and ever-growing data availability open up new possibilities to tackle these challenges. In the past few years, various deep neural networks (DNNs) have been designed to learn the spatiotemporal dynamics in latent spaces enabled by proper orthogonal decomposition (POD)[1–4] or convolutional encoding-decoding operations[5–8]. In particular, fast neural simulators based on graph neural networks (GNN) have been proposed and demonstrated to predict spatiotemporal physics on irregular domains with unstructured meshes[9,10]. Although showing good promise, most of these works are purely data-driven and black-box in nature, which rely on "big data" and may have poor generalizability, particularly in out-of-sample regimes in the parameter space. As a more promising strategy, baking physics prior knowledge (e.g., conservation laws, governing equations, and constraints) into deep learning is believed to be very effective to improve its sample efficiency and generalizability[11], here referred to as physics-informed deep learning (PiDL). An impressive contribution in this direction is physics-informed neural networks (PINNs)[12], where well-posed PDE information is leveraged to enable deep learning in data-sparse regimes. The general idea of PINNs is to learn (or solve) the PDE solutions with DNNs, where the loss functions are formulated as a combination of the data mismatch and residuals of known PDEs, unifying forward and inverse problems within the same DNN optimization framework. The merits of PINNs have been demonstrated over various scientific applications, including fast surrogate/meta modeling[13–15], parameter/field inversion[16–19], and solving high-dimensional PDEs[20,21], to name a few. Due to the scalability challenges of the pointwise fully-connected PINN formulation to learn continuous functions[22–24] or operators[25–28], many remedies and improvements in terms of training and convergence have been proposed[29–31]. In particular, there is a growing trend in developing field-to-field discrete PINNs by leveraging convolution operations and numerical discretizations, which have been demonstrated to be more efficient in spatiotemporal learning[32,33]. For example, convolution neural networks (CNN) or graph convolution networks (GCN) were built to approximate the discrete PDE solutions, where the PDE residuals can be formulated in either strong or weak forms by finite-difference[34–36], finite volume[37], or finite element methods[38–42]. Moreover, recurrent network formulation informed by discretized PDEs have been developed for spatiotemporal dynamic control using model-based reinforcement learning[43].

In the realm of PINN framework, the term "physics-informed" generally denotes the incorporation of PDE residuals into the loss or likelihood functions to guide or constrain DNN training. Despite this development, the question of how to effectively use physics-inductive bias—i.e., (partially) known governing equations—to inform the learning architecture design remains an intriguing, relatively unexplored area. The primary focus of this paper is to address this issue. Recent studies have revealed the deep-rooted relationship between neural network structures and ordinal/partial differential equations (ODEs/PDEs)[44–49]. For example, Lu et al.[45] bridged deep convolutional network architectures and numerical differential equations. Chen et al.[50] showed that the residual networks (ResNets)[51] can be interpreted as the explicit Euler discretization of an ODE, and ODEs can be used to formulate the continuous residual connection with infinite depths, known as the NeuralODE[52]. Motivated by differential equations, novel deep learning architectures have been recently developed in the computer science community, e.g., new convolutional ResNets guided by parabolic and hyperbolic PDEs[47], GRAND as a graph network motivated by diffusion equations[48], and PDE-GCN motivated by hyperbolic PDEs to improve over-smooth issues in deep graph learning[49]. However, these studies mainly aimed to develop generic DNN architectures with some desired features by utilizing specific properties of certain PDEs (e.g., diffusion, dispersion, etc.), and the designed neural networks are not necessarily used to learn the physical processes governed by those PDEs. An attempt was made by Shi et al.[53] to learn PDE-governed dynamics by limiting trainable parameters of CNN using finite difference operators. Despite being a novel attempt, the approach is still purely data-driven without effectively utilizing governing PDEs.

Therefore, this work explores PiDL through learning architecture design, inspired by the broader concept of differentiable programming (∂P) - extending DNNs to more general computer programs that can be trained in a similar fashion to deep learning models[54]. In general, a ∂P model is formulated by marrying DNNs with a fully differentiable physics-based solver, and thus the gradients can be back-propagated through the entire hybrid neural solver based on automatic differentiation (AD) or discrete adjoint methods. Relevant works include universal differential equations (UDE)[55], NeuralPDE[56], and others, where DNNs are formulated within a differentiable PDE solver for physics-based modeling. In particular, this idea has been recently explored in predictive modeling of rigid body dynamics[57,58], epidemic dynamics[59], and fluid dynamics[60–62]. These studies imply great promise of incorporating physics-induced prior (i.e., PDE) into DNN architectures.

In this paper, we present a creative approach to designing distinctive learning architectures for predicting spatiotemporal dynamics, where the governing PDEs are preserved as convolution operations and residual connections within the network architecture. This is in sharp contrast to prior PiDL work where the physical laws were enforced as soft constraints within the loss functions, supported by an comprehensive comparision between the proposed method and physics-informed variants of multiple state-of-the-art neural operators. Specifically, we develop an auto-regressive neural solver based on a convolutional ResNet framework, where the residual connections are constructed by preserving the PDE operators in governing equations, which are (partially) known a priori, discretized on low-resolution grids. Meanwhile, encoding-decoding convolution operations with trainable filters enable high-resolution state predictions on fine grids. Compared to classic ResNets with black-box residual connections, the proposed PPNN is expected to be superior in terms of both training efficiency and out-of-sample generalizability for, e.g., unseen boundary conditions and parameters, and extrapolating in time. Conceptually, the proposed framework is similar to using neural networks for closure modeling of classic

numerical solvers, which has been explored previously. However, several distinct features make our methodology more general that extends substantially beyond prior studies on merging machine learning with numerical solvers[63–65]. Our work is not focused on simply coupling a neural network with a numerical solver or training it to learn specific closures. Instead, the proposed framework integrates (partially or wholly known) physical laws, expressed as PDE operators, directly into the neural networks. This leads to a creative neural architecture design, reflecting a unique design strategy that leverages the profound connection between neural network architecture components and ODEs/PDEs. The differentiability brought by representing numerical operators with neural network components makes an end-to-end time sequence training possible, which distincts the proposed method from closure model learning. This strategy offers a fresh perspective on incorporating physical knowledge into neural network design, underscoring that such integration can enhance the model's performance in predicting complex spatiotemporal dynamics. When compared with the other approach of leveraging physics priors into neural network training: the "physics-informed" methods, our proposed PPNN does show significant merit in terms of cost, generalizability and long-term prediction accuracy. The contributions of this work are summarized as follows: (i) a framework for physics-inspired learning architecture design is presented, where the PDE structures are preserved by the convolution filters and residual connection; (ii) multi-resolution information passing through network layers is proposed to improve long-term model rollout predictions over large time steps; (iii) the superiority of the proposed PPNN is demonstrated for PDE operator learning in terms of training complexity, extrapolability, and generalizability in comparison with the baseline black-box models, using a series of comprehensive numerical experiments on spatiotemporal dynamics governed by various parametric unsteady PDEs, including reaction-diffusion equations, Burgers' equations, and unsteady Navier-Stokes equations.

Results and discussion

Learning spatiotemporal dynamics governed by PDEs. We consider a multi-dimensional spatiotemporal system of $u(x, t; \lambda)$ governed by a set of nonlinear coupled PDEs parameterized by $\lambda \in \mathbb{R}^d$, which is a $d-$ dimensional parameter vector, while x and t are spatial and temporal coordinates, respectively. Our goal is to develop a data-driven neural solver for rapid predictions of spatiotemporal dynamics given different parameters λ. The neural solver is formulated as a next-step DNN model by learning the dynamic transitions from the current step t to the next time step $t + \Delta t$ (Δt is the time step).

This study focuses on the learning architecture design for improving the robustness, stability, and generalizability of data-driven next-step predicting models, which commonly suffer from considerable error accumulations due to the auto-regressive formulation and fails to operate in a long-span model rollout. In contrast to existing models which are black-box, we propose a PDE-preserved neural network (PPNN) architecture inspired by the relationship between network structures and PDEs, by hypothesizing that the predictive performance can be significantly improved if the network is constructed by preserving (partially) known governing PDEs of the spatiotemporal dynamics to be learned. Specifically, the known portion of the governing PDEs in discrete forms are preserved in residual connection blocks. As shown in Fig. 1a, the PPNN architecture features a residual connection which consists of two parts: a trainable network and a PDE preserving network, where the right hand side (RHS) of the governing PDE, discretized on finite difference grid, is

represented by a convolution neural network. The weights of the PDE preserved convolutional residual component are determined by the discretization scheme and remain constant during training.

However, in practice, neural solvers are expected to roll out much faster than numerical solvers, and the time step Δt would be orders of magnitude larger than that used in conventional numerical solvers, which may lead to catastrophic stability issues if naively embedding the discretized PDE into the neural network. To this end, we implement a multi-resolution PPNN based on the convolutional (conv) ResNet backbone (shown in Fig. 1b), where PDE-preserving blocks work on a coarse grid to enable stable model rollout with large evolving steps. This is achieved by using the bilinear down-sampling and bicubic up-sampling algorithms to auto-encode the PDE-preserved hidden feature in a low-resolution space, which is then fed into the main residual connection in the original high-resolution space.

Together with the trainable block, which consists of decoding-encoding convResNet blocks defined on the fine mesh, PPNN enables predictions at a high resolution. Moreover, the network is conditioned on physical parameters λ, enabling fast parametric inference and generalizing over the high-dimensional parameter space. (More details are discussed in the Methods section.)

In this section, we evaluate the proposed PDE structure-preserved neural network (PPNN) on three nonlinear systems with spatiotemporal dynamics, where the governing PDEs are known or partially-known a priori. Specifically, the spatiotemporal dynamics governed by FitzHugh-Nagumo reaction diffusion (RD) equations, Burgers' equations, and incompressible Navier-Stokes (NS) equations with varying parameters λ (e.g., IC, diffusion coefficients, Reynolds number, etc.) in 2D domains are studied. In particular, we will study the scenarios where either fully-known or incomplete/inaccurate governing PDEs are preserved. To demonstrate the merit of preserving the discrete PDE structure in ConvResNet, the proposed PPNN is compared with the corresponding black-box ConvResNet next-step model as a baseline, which is a CNN variant of the MeshGraphNet[9] (see section Next-step prediction models based on convolutional ResNets). For a fair comparison, the network architecture of the trainable portion of the PPNN is the same as the black-box baseline model. Moreover, all models are compared on the same sets of training data in each test case. The generalizability, robustness, training and testing efficiency of the PPNN are investigated in comparison with its corresponding blackbox baseline. It is noted that the novelty of this work lies not in exploring varied methods for learning closures for traditional PDE solvers but in the inventive integration of known physical laws into the architecture of convolutional residual neural networks. We, therefore, consider it critical to compare the PPNN with its black-box counterpart, which learn from data without explicit integration of the underlying physics. This comparison enables us to highlight the unique benefits of integrating known physics into deep learning models, an area that has, to date, received limited attention. Given the prevalence of black-box neural networks in data-driven surrogate modeling where the governing PDEs are often known or partially known, this comparison is both relevant and fair. We believe that this provides a valuable perspective and a substantial contribution to the field. Moreover, it is also worth noting that, PPNN is not constrained to any specific DNN architectures. Rather, we demonstrate that it serves as a versatile framework that can be synergistically combined with a variety of DNN architectures such as U-Net[66] – widely recognized for its multi-scale structure, and Vision Transformer (ViT)[67], which has become the backbone for most computer vision tasks. (see section PPNN as a general framework for embedding known physics). Moreover, the
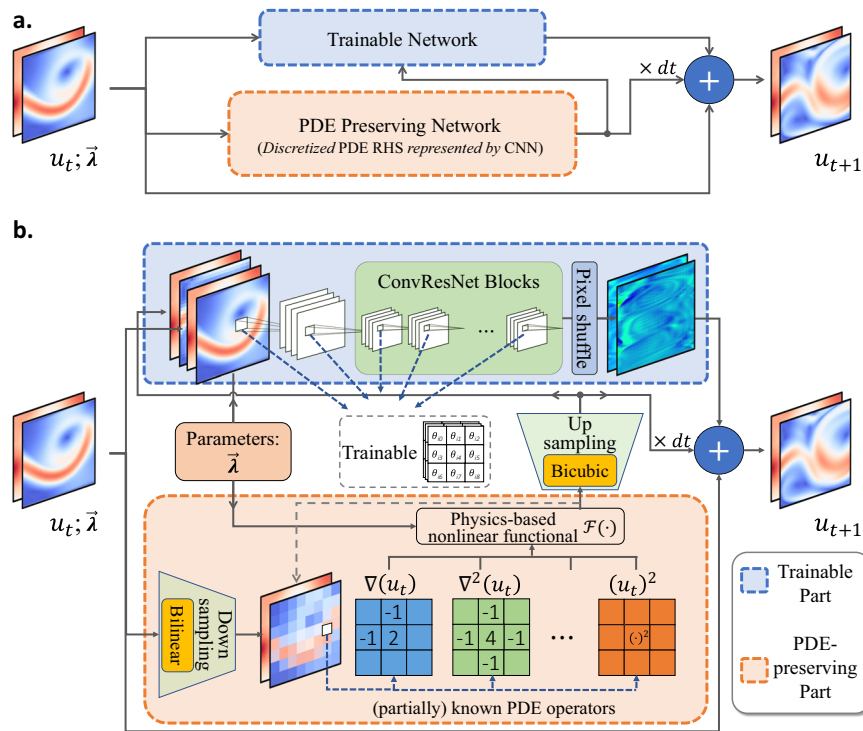
Fig. 1 Schematic diagram of the proposed partial differential equation (PDE)-preserved neural network (PPNN). a A schematic representation illustrating the concept of the PPNN framework. **b** A detailed schematic of the ConvResNet-based PPNN, which consists of the trainable part and the PDE-preserving part. The two portions of PPNN are combined together in a multi-resolution setting. The discretized form of the governing PDEs are embedded into the network structure via prescribed convolutions filters and the residual connection.

relationship between the PDE-preserving portion of PPNN and numerical solvers is discussed. Note that we use the same network setting, i.e., same network structure, hyperparameters and training epochs, for all the test cases (except for the NS system, which has slight modifications adapting to three state variables). More details about the neural network settings can be found in Section Supplementary Note 3 in supplementary information.

All the DNN predictions are evaluated against the high-resolution fully-converged numerical solutions as the reference using a full-field error metric ϵ_t defined at time step t as,

$$\epsilon_t = \frac{1}{N} \sum_{i=1}^{N} \frac{\left\| f_\theta(\hat{u}_{t-1}, \lambda_i | \widetilde{\theta}) + \hat{u}_{t-1} - u_t(\lambda_i) \right\|_2}{\left\| u_t(\lambda_i) \right\|_2}, \quad (1)$$

where N indicates the number of the testing physical parameters λ_i, $u_t(\lambda_i)$ is the reference solution at time step t corresponding to the physical parameter λ_i, f_θ represents the trained neural network function with optimized weights $\widetilde{\theta}$, and $\hat{u}_{t-1}$ represents the state predicted by the model at previous time step $t - 1$,

$$\hat{u}_t = f_\theta(\hat{u}_{t-1}, \lambda_i | \theta) + \hat{u}_{t-1}, t \in [2, n] \\ \hat{u}_1 = f_\theta(u_0(\lambda_i), \lambda_i | \theta) + u_0(\lambda_i) \quad (2)$$

where n is the number of testing steps, $u_0(\lambda_i)$ represents the initial condition given λ_i. For brevity, numerical details for each case are given in Section Supplementary Note 4 of the supplementary information.

When the governing PDEs are fully known. We herein consider three well-known spatiotemporal PDEs (e.g., the FitzHugh-Nagumo reaction diffusion equations, the Viscous Burgers' equation and the Naiver-Stokes equations) when the closed-form equations are fully known.

FitzHugh-Nagumo reaction diffusion equations. We first consider a spatiotemporal dynamic system governed by the FitzHugh-Nagumo equations with periodic BCs, which is a generic model for excitable media. The main part of the FitzHugh-Nagumo model is reaction-diffusion (RD) equations,

$$\frac{\partial u}{\partial t} = \gamma \nabla^2 u + R(u), \quad t \in [0, T], \quad (3)$$

where $u = \left[u(x, y, t), v(x, y, t) \right]^T \in \mathbb{R}^2$ are two interactive components, γ is the diffusion coefficient, $T = 0.2s$ is the time length we simulated, and $R(u) = \left[R_u(u, v), R_v(u, v) \right]^T$ are source terms for the reaction,

$$R_u(u, v) = u - u^3 - v + \alpha, \\ R_v(u, v) = \beta(u - v), \quad (4)$$

where $\alpha = 0.01$ represents the external stimulus and $\beta = 0.25$ is the reaction coefficient. The initial condition (IC) u_0 is a random field and generated by randomly sampling from a normal distribution,

$$u(x, y, 0), v(x, y, 0) \sim \mathcal{N}(0, 1), \quad (5)$$

which is then linearly scaled to [0.1, 1.1]. Given different ICs and diffusion coefficients γ, varying dynamic spatial patterns of neuron activities can be simulated. Here, the next-step neural solvers are trained to learn and used to predict the spatiotemporal dynamics of varying modeling parameters (i.e., ICs and diffusion coefficients). Namely, we attempt to build a surrogate model in a very high-dimensional parameter space $\lambda \in \mathbb{R}^d$, where $d = 65,537$, since the dimensions for IC and diffusion coefficient are 256^2 and 1, respectively. The reference solutions are obtained on the simulation domain $(x, y) \in [0, 6.4] \times [0, 6.4]$, discretized with a fine mesh of 256×256 grids, based on the finite difference method.
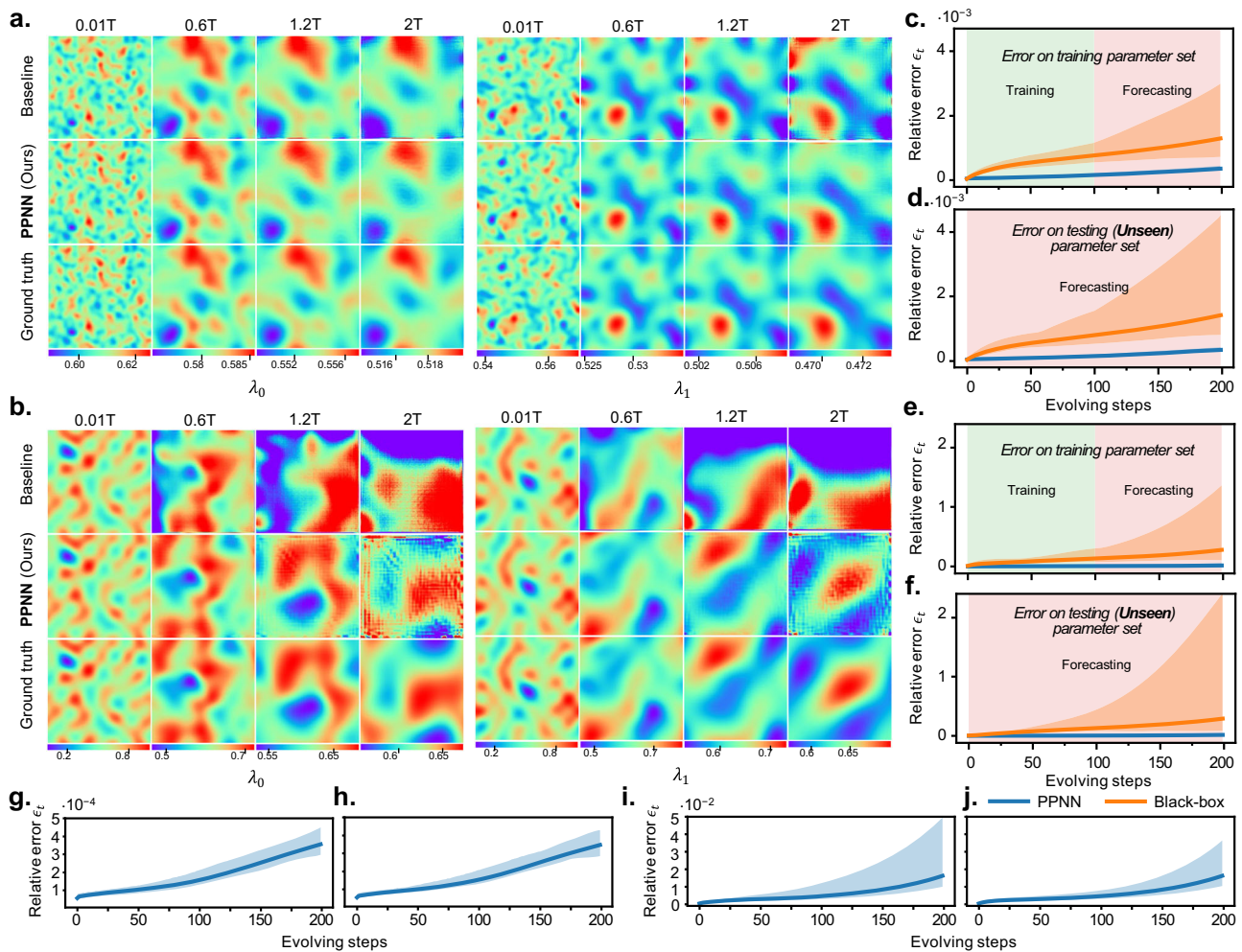
Fig. 2 Prediction comparison in the reaction-diffusion (RD) case and viscous Burgers' case. a, b Predicted solution snapshots of u for the RD equations (**a**) and the velocity magnitude $\|u\|_2$ for the Burgers' equations (**b**) at different time steps and unseen parameters, obtained by black-box ConvResNet (baseline model, first rows), and partial differential equation preserved neural network (PPNN, our method, second rows), compared against ground truth (high-resolution numerical simulation, third rows). λ_0, λ_1 are randomly selected testing (unseen) parameters in each system. **c–f** Relative prediction error ϵ_t of PPNN (blue lines——) and black-box ConvResNet baseline (orange lines——) for the RD dynamics (**c, d**) and Burgers' equations (**e, f**) averaged on 100 randomly sampled training parameters λ (**c–e**) and testing (unseen) parameters (**d–f**). The shaded area shows the maximum and minimum relative errors of all testing trajectories. **g, h** Zoom in views of the relative error curve of PPNN shown in (**c, d–g, h**) respectively. **i, j** Zoom in views of the relative error curve of PPNN shown in (**e, f–i, j**) respectively.

Figure 2a shows the PPNN-predicted solution snapshots of the RD equations at four randomly selected test parameters (i.e., randomly generated ICs and unseen diffusion coefficients). The prediction results of baseline black-box ConvResNet (first row) and the proposed PPNN (second row) are compared against the ground truth reference (third row). It can be seen that both models agree with the reference solutions for $t < 0.6T$, showing good generalizability on testing ICs and γ for a short-term model rollout. However, the error accumulation becomes noticeable for the black-box baseline when $t > T$, and the spatial patterns of the baseline predictions significantly differ from the reference at $t = 2T$, which is an expected issue for the next-step predictors. In contrast, the results of our PPNN have an good agreement with the reference solutions over the entire time span $[0, 2T]$ on all testing parameters, showing great robustness, predictability, and generalizability in both the spatiotemporal domain and parameter space. Predicted solutions on more testing parameters are presented in Fig. S12.

To further examine the error propagation in time for both models, the relative testing errors ϵ_t averaged over 100 randomly

selected parameters in training and testing sets are computed and plotted in Fig. 2, where Fig. 2c shows the averaged model-rollout error evaluated on 100 training parameters and the Fig. 2d shows the error averaged on 100 randomly generated testing parameters. (Zoom in views of Fig. 2c and d can be found in Fig. 2g and h, respectively.) The model is only trained within the range of $1T$ ($100\Delta t$), and it is clearly seen that the rollout error of the black-box model significantly grows in the extrapolation range $[T, 2T]$ (from $100\,\Delta t$ to $200\,\Delta t$), where $\Delta t = 200\delta t$ is the learning step size which is 200 numerical timesteps δt. The error accumulation becomes more severe for the unseen testing parameters. However, our PPNN predictions maintain an impressively low error, even when extrapolating twice the length of the training range. Besides, the scattering of the error ensemble is significantly reduced compared to the black-box baseline, indicating great robustness of the PPNN for various testing parameters.

Viscous Burgers' equation. For the second case, we study the spatiotemporal dynamics governed by the viscous Burgers'

equations on a 2D domain with periodic boundary conditions,

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = \nu \nabla^2 \boldsymbol{u}, \quad t \in [0, T], \tag{6}$$

where $\boldsymbol{u} = [u(x, y, t), v(x, y, t)]^T \in \mathbb{R}^2$ is the velocity vector, $T = 2s$ is the time length we simulated, and ν represents the viscosity. The initial condition (IC) $\boldsymbol{u}_0$ is generated according to,

$$\boldsymbol{u}_0 = \begin{cases} u_0 = \sum_{i=-4}^{4} \sum_{j=-4}^{4} r_{i,j}^{(1)} \sin(ix + jy) + r_{i,j}^{(2)} \cos(ix + jy) \\ v_0 = \sum_{i=-4}^{4} \sum_{j=-4}^{4} r_{i,j}^{(3)} \sin(ix + jy) + r_{i,j}^{(4)} \cos(ix + jy) \end{cases} \quad r_{i,j}^{(k)} \sim \mathcal{N}(0,1); k = 1, 2, 3, 4, \tag{7}$$

where x, y are spatial coordinates of grid points, and $r_{i,j}^{(k)}; k \in 1, 2, 3, 4$ are random variables sampled independently from a normal distribution. The IC is normalized in the same way as mentioned in the RD case. We attempt to learn the dynamics given different ICs and viscosities. Similar to the RD cases, the parameter space $\mathbb{R}^d$ is also high-dimensional ($d = 324$), as the IC is parameterized by 4×9^2 independent random variables and the scalar viscosity can also vary in range $[0.02, 0.07]$. The reference solution is generated by solving the Burgers' equations on the domain of $(x, y) \in [0, 3.2]^2$, discretized by a fine mesh of 256×256 grids using finite difference method.

The velocity magnitude contours of the 2D Burgers' equation with different testing parameters are shown in Fig. 2b, obtained by the black-box baseline, PPNN, and reference numerical solver, respectively. Note that all the testing parameters are not seen during training. (More predicted solutions on different testing parameters are presented in Fig. S13.) Similar to the RD case, PPNN shows a significant improvement over the black-box baseline in terms of long-term rollout error accumulation and generalizability on unseen ICs and viscosity ν. Due to the strong convection effect, black-box baseline predictions deviate from the reference very quickly, and significant discrepancies in spatial patterns can be observed as early as $t < 0.6T$. In general, the black-box baseline suffers from the poor out-of-sample generalizability for unseen parameters, making the predictions useless. Our PPNN significantly outperforms the black-box baseline, and its prediction results agree with the reference for all testing samples. Although slight prediction noises are present after a long-term model rollout ($t > 1.2T$), the overall spatial patterns can be accurately captured by the PPNN even at the last learning step ($t = 2T$). The error propagation of both models is given in Fig. 2, where the rollout errors ϵ_t at each time step, averaged over 100 randomly selected parameters from training and testing sets, are plotted. Figure 2e shows the averaged model rollout error evaluated on 100 training parameters, while Fig. 2f shows the error averaged on 100 randomly generated parameters, which are not used for training. Zoom in views of Fig. 2e and f can be found in Fig. 2i and j, respectively. As both models are only trained with the $1T$ ($100\Delta t$) time steps for each parameter in the training set, it is clear that the error of the black-box model grows rapidly once stepping into the extrapolation range $[T, 2T]$. The error accumulation effect of the black-box model becomes more obvious for those parameters which are not in the training set due to the poor generalizability. In contrast, the error of PPNN predictions remains surprisingly low even in the extrapolation range for both training and testing regimes, and there is nearly no error accumulation. In addition, the error scattering significantly shrinks compared to that of the black-box model, indicating significantly better accuracy, generalizability and robustness of the PPNN compared to the black-box baseline.

Naiver-Stokes equations. The last case investigates the performance of PPNN to learn an unsteady fluid system exhibiting complex vortex dynamics, which is governed by the 2D parametric unsteady Naiver-Stokes (NS) equations:

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\nabla p + \nu \nabla^2 \boldsymbol{u}, \quad t \in [0, T],$$
$$\nabla \cdot \boldsymbol{u} = 0, \tag{8}$$

where $\boldsymbol{u} = [u(x, y, t), v(x, y, t)]^T \in \mathbb{R}^2$ is the velocity vector, $p(x, y, t) \in \mathbb{R}$ is the pressure, and $\nu = 1/\text{Re}$ represents the kinematic viscosity (Re is the Reynolds number). The NS equations are solved in a 2D rectangular domain $(x, y) \in [0, 4] \times [0, 1]$, where a jet with dynamically-changed jet angle is placed at the inlet. Namely, the inflow boundary is defined by a prescribed velocity profile $\boldsymbol{u}(0, y, t)$,

$$\boldsymbol{u}(0, y, t) = \begin{bmatrix} u(0, y, t) \\ v(0, y, t) \end{bmatrix} = \begin{bmatrix} \exp\left(-50(y - y_0)^2\right) \\ \sin(t) \cdot \exp\left(-50(y - y_0)^2\right) \end{bmatrix} \tag{9}$$

where y_0 represents the vertical position of the center of the inlet jet. The outflow boundary condition is set as pressure outlet with a reference pressure of $p(4, y, t) = 0$. No-slip boundary conditions are applied on the upper and lower walls. In this case, the neural network models are expected to learn the fluid dynamics with varying Reynolds number Re and jet locations y_0. Namely, a two-dimensional physical parameter vector $\boldsymbol{\lambda} = [\text{Re}, y_0]^T$ is considered. In training set, we use five different Re evenly distributed in the range $[2 \times 10^3, 1 \times 10^4]$ and 9 different jet locations y_0 uniformly selected from 0.3 to 0.7. Figure 3a–b shows the snapshots of velocity magnitude of the NS equations at two representative testing parameters, which are not seen in the training set. To be specific, $\boldsymbol{\lambda}_0 = [2500, 0.325]^T$ represents a relatively low Reynolds number $Re = 2500$ with the jet located at $y_0 = 0.325$, while $\boldsymbol{\lambda}_1 = [8500, 0.575]^T$ is a higher Reynolds number case ($Re = 8500$) with the jet located at $y_0 = 0.325$. The rollout prediction results of the PPNN and baseline black-box ConvResNet are compared with the ground truth reference. Although both models can accurately capture the spatiotemporal dynamics at the beginning stage (when $t \leq 0.4T$), showing good predictive performance for the unseen parameters for a short-term rollout, the predictions by the black-box model are soon overwhelmed by the noises due to the rapid error accumulation ($t > T$). However, the proposed PPNN significantly outperforms the black-box baseline as it managed to provide accurate rollout predictions even at the last testing steps ($t = 3T$), which extrapolate as three times long as the training range, indicating preserving the PDE structure can effectively suppress the error accumulation, which is unavoidable in most auto-regressive neural predictors. To further investigate the error propagation in time for both models, we plot the relative testing errors ϵ_t against time in Fig. 3c–d, which are averaged over 5 randomly selected parameters in both training (Fig. 3c) and testing sets (Fig. 3d). We can clearly see that PPNN managed to maintain low rollout error in both training and extrapolation ranges, in contrast to the significantly higher error accumulation in the black-box baseline results. In particular, the black-box model relative error visibly grows only after a short-term model rollout and increases rapidly once it enters the extrapolation range even for testing on the training parameter set (Fig. 3c), and the errors are accumulated even faster for the testing on unseen parameters (Fig. 3d). On the contrary, our PPNN has almost no error accumulation and performs much more consistently between the training and extrapolation ranges, with significantly lower rollout errors. The results again demonstrate outstanding predictive accuracy and generalizability of the
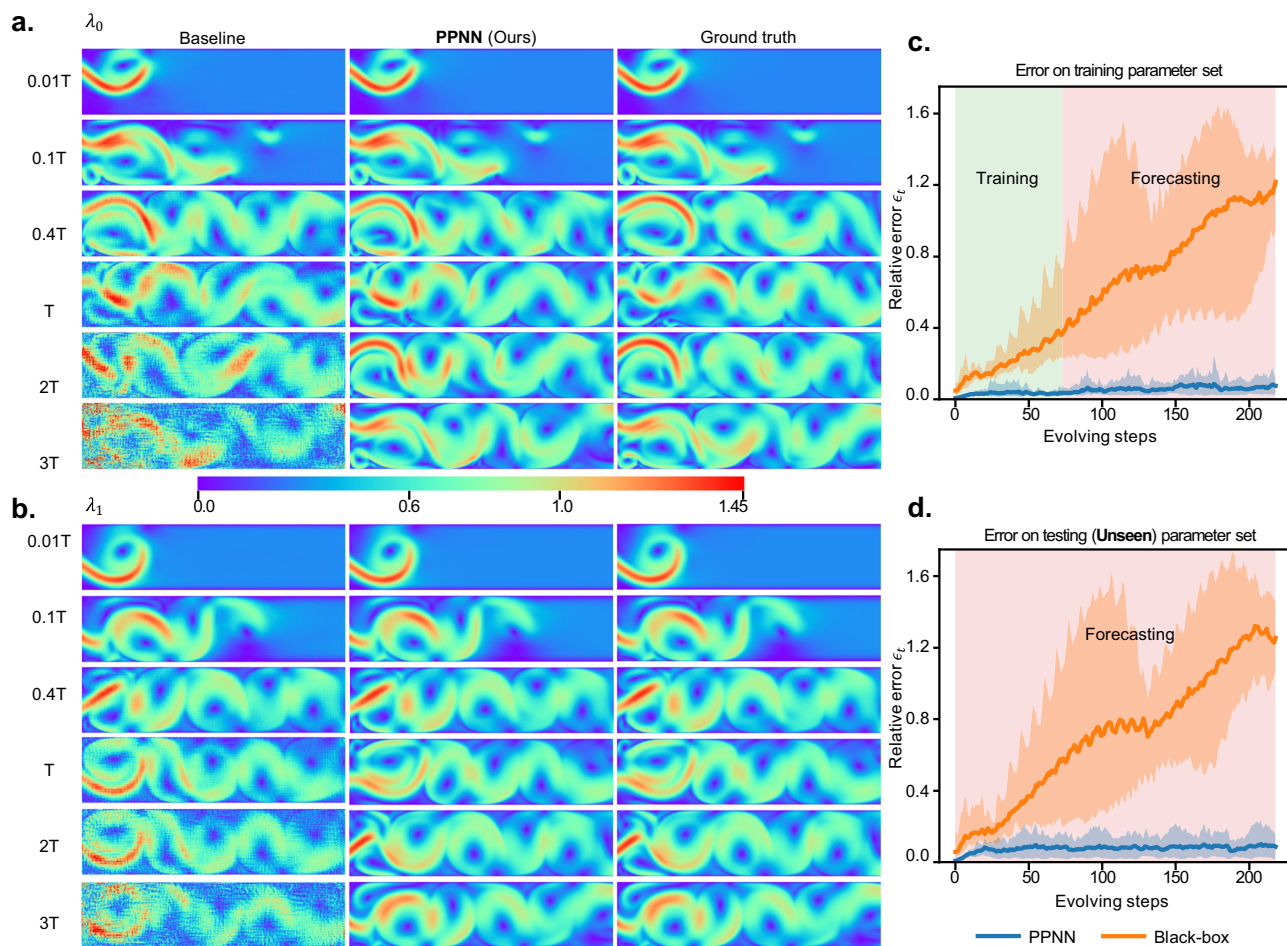
Fig. 3 Prediction comparison in the case governed by Naiver-Stocks (NS) equations. a, **b** Predicted solution snapshots of velocity magnitude $\|\boldsymbol{u}\|_2$ for the NS equations obtained by black-box ConvResNet (baseline); partial differential equation preserved neural network (PPNN, Ours), compared against the ground truth (high-resolution numerical simulation), where λ_0 is (Re $= 2500, y_0 = 0.325$, shown in (**a**)), and λ_1 is high Reynolds number (Re $= 8500, y_0 = 0.575$, shown in (**b**)). **c**, **d** Relative prediction error ϵ_t of PPNN (blue lines——) and black-box ConvResNet baseline (orange lines——) at different timesteps for the NS equation, averaged on 5 randomly sampled (**c**) training parameters and (**d**) testing (unseen) parameters. The shaded areas show the scattering of the relative errors over all testing trajectories.

proposed method. Besides, PPNN also shows a significantly smaller uncertainty range, indicating great robustness among different testing parameters.
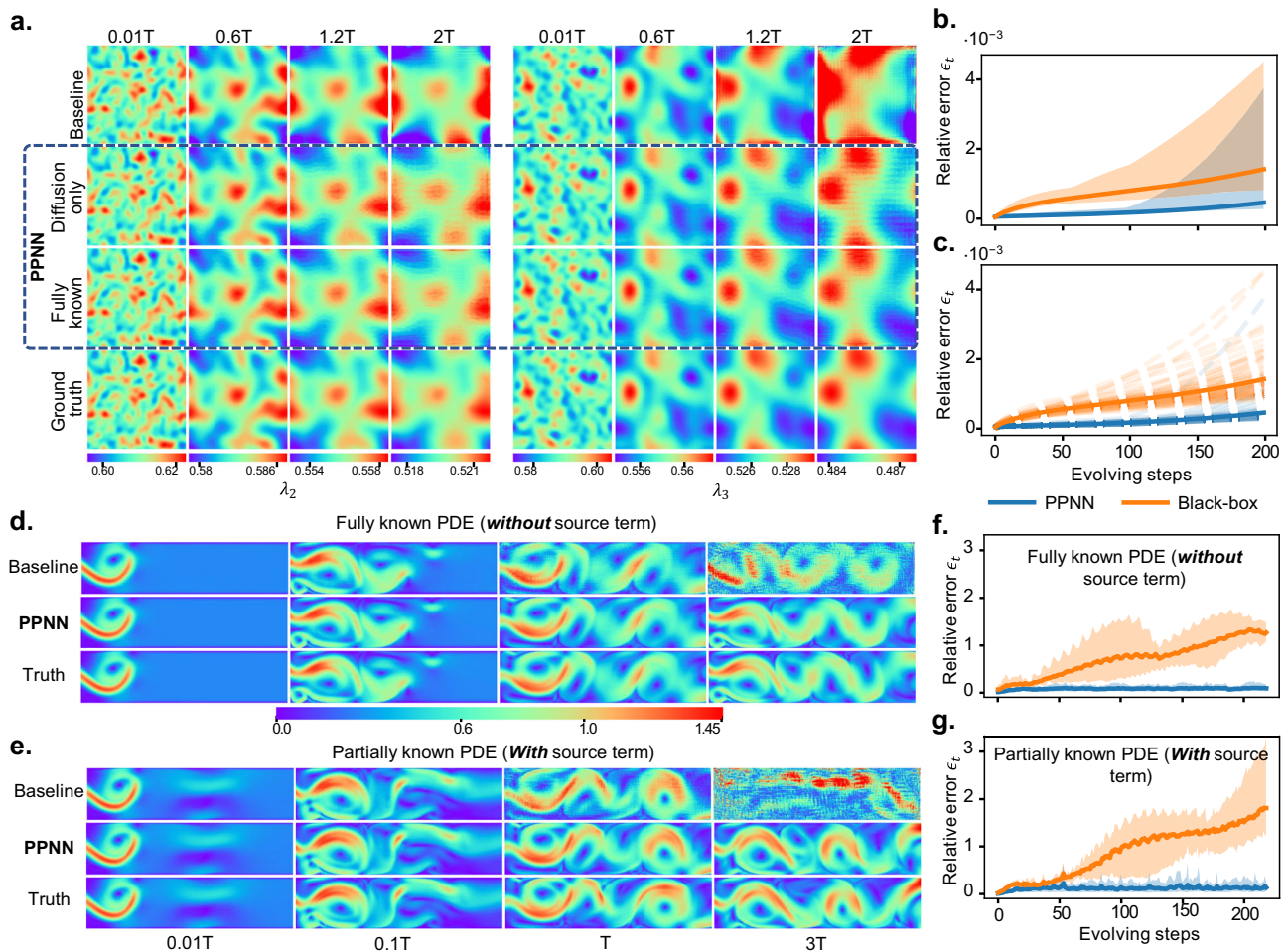
When the governing PDEs are partially known. In real-world applications, the underlying physics behind complex spatio-temporal phenomena might not be fully understood, and thus the governing equations can be incomplete, e.g., with unknown source terms, inaccurate physical parameters, or uncertain I/BCs. Such partially-known physics poses great challenges to the traditional simulation paradigm since the governing equations are partially known. Nonetheless, the incomplete prior knowledge can be well utilized in our proposed PPNN framework, where preserving partially-known governing PDE structures can still bring significant merits to data-driven spatiotemporal learning and prediction, which will be discussed in this subsection.

Reaction diffusion equations with unknown reaction term. We first revisit the aforementioned FitzHugh-Nagumo RD equations. Here, we consider the scenario where only the diffusion phenomenon is known in the FitzHugh-Nagumo RD dynamics. Namely, the reaction source terms remain unknown and PPNN only preserves the incomplete RD equations, i.e., 2D diffusion equations,

$$\frac{\partial \boldsymbol{u}}{\partial t} = \gamma \nabla^2 \boldsymbol{u}. \tag{10}$$

All the case settings remain the same as those discussed previously. Although incomplete/inaccurate prior knowledge about the RD system is preserved, our PPNN still shows a significant advantage over the black-box baseline. Figure 4a compares the snapshots of reactant u at two randomly selected unseen parameters λ_2 and λ_3 predicted by black-box baseline model (first rows), PPNN with the diffusion terms preserved only (second rows), PPNN with the complete RD equation preserved (third rows), against the ground truth (fourth rows). The PPNNs preserving either complete or incomplete RD equations accurately capture the overall patterns and well agree with the reference solutions, while the black-box baseline shows notable discrepancy and large errors, particularly at $t = 2T$, which is the twice of the training phase length. At the last extrapolation step, the prediction results of black-box baseline show some visible noise and are less smooth compared to the results by preserving the complete RD equation, indicating that lack of the prior information on the reactive terms could slightly reduce the improvement by PPNN. Figure 4b–c shows the relative model rollout errors averaged over 100 test trajectories, which are not

Fig. 4 Prediction comparison in the cases where the governing equations are partially known. a Predicted solution snapshots of u for the reaction-diffusion (RD) equations at different time steps and unseen parameters, obtained by black-box ConvResNet (baseline model), and partial differential equation (PDE)-preserved neural network (PPNN, preserving diffusion terms only), and PPNN (preserving complete FitzHugh-Nagumo RD equations), compared against ground truth. λ_2 and λ_3 are two randomly selected testing (unseen) parameters. **b, c** Averaged relative testing error ϵ_t of the PPNN with incomplete PDE (blue lines——) and Black-Box ConvResNet baseline (orange lines——) for the RD dynamics evaluated on 100 randomly generated testing (unseen) parameters (same parameters as shown in Fig. 2**c**). Shaded areas in (**b**) indicate envelopes of the maximum and minimum relative errors of all testing trajectories, while the dash lines in (**c**) indicate the relative error of each test trajectory. **d, e** Predicted solution snapshots of flow velocity magnitude $\|\boldsymbol{u}\|_2$ obtained by black-box ConvResNet (baseline), PPNN (ours), compared against ground truth (high-resolution numerical simulation) of the NS equations without (**d**) and with (**e**) an unknown magnetic source term, respectively. The PPNN only preserves a NS equation portion for both scenarios, which are at the same testing (unseen) parameter $\boldsymbol{\lambda} = [9000, 0.475]^T$, which is not in the training set. **f, g** Relative prediction errors ϵ_t of the PPNN (blue line——) and black-box ConvResNet baseline (orange line——) for the NS equation with (**e**) and without (**f**) a unknown magnetic body force, averaged on five randomly sampled unseen parameters. The shaded area shows the scattering of relative errors for all testing trajectories.

seen in the training set. The shaded area in the upper panel shows the error distribution range of these 100 test trajectories. Even the preserved PDEs are not complete/accurate, the mean relative error (blue line——) remains almost the same as the PPNN with fully-known PDEs (see Fig. 2a), which is significantly lower than that of the black-box baseline (orange line——), showing a great advantage of preserving governing equation structures even if the prior physics knowledge is imperfect. Compared to the PPNN with fully-known PDEs, the error distribution range by preserving partially-known PDEs is increased and error ensemble is more scattered, implying slightly decreased robustness. Although the envelope of the error scattering for incomplete PDEs is much larger than that of the case with fully-known PDEs, this is due to a single outlier trajectory, which can be seen in Fig. 4c. This indicates embedding a incomplete PDE terms will leads to restricted performance of PPNN when the disregarded term plays an important role in the dynamic system. In general, the standard

deviation of the error ensemble from the PPNN with partially-known PDE ($\sigma = 1.123 \times 10^{-4}$) is still significantly lower than that of the black-box baseline ($\sigma = 3.412 \times 10^{-4}$). In comparison, the standard deviation of errors in PPNN with fully-known PDEs over the 100 test trajectories is 0.854×10^{-4}.

Naiver-Stokes equations with an unknown magnetic field. In the second case, we consider the the complex magnetic fluid dynamic system governed by Naiver-Stokes equations with an unknown magnetic field:

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\nabla p + \nu \nabla^2 \boldsymbol{u} + \boldsymbol{F},$$
$$\nabla \cdot \boldsymbol{u} = 0, \tag{11}$$

where $\boldsymbol{u} = [u, v]^T$ is the velocity vector; p is the pressure; while ν represents the kinematic viscosity. Here $\boldsymbol{F} = [F_x, F_y]^T$ represents

the body force introduced by a magnetic field:

$$F_x = mH\frac{\partial H}{\partial x}, \qquad F_y = mH\frac{\partial H}{\partial y}$$
$$H(x,y) = \exp\left[-8\left((x - L/2)^2 + (y - W/2)^2\right)\right]$$

(12)

where $m = 0.16$ is the magnetic susceptibility, and H is a time-invariant magnetic intensity. The contour of the magnitude of the body force source term is shown in the supplementary information (see Fig. S11). In this case, the magnetic field remains unknown and PPNN only preserves the NS equation without the magnetic source term. All the other case settings remain unchanged as described in the Naiver-Stokes equation case.

Similar to what we observed in the example of RD equations with the unknown reaction term, the PPNN still remains a significant advantage over the black-box baseline even by preserving an incomplete physics of the flow in a magnetic field. Figure 4d–e, shows the velocity magnitude $\|u\|_2$ results of the flow with (Fig. 4d) or without (Fig. 4e) a magnetic field at the same testing parameter $(\lambda = [\text{Re} = 9000, y_0 = 0.475]^T)$, predicted by the PPNN and black-box ConvResNet, compared against the reference solution. For both scenarios, only the NS equation portion is preserved in the PPNN, i.e., the magnetic field remains unknown. Figure 4d shows the solution snapshots for the flow without the magnetic field (i.e., PPNN preserving the complete physics), while Fig. 4e shows the predictions of the flow with the magnetic field (i.e., PPNN preserving an incomplete physics). Comparing the reference solutions at upper and lower panels, the spatiotemporal patterns of the flow fields exhibit notable differences for the cases with and without magnetic fields. In both scenarios, the black-box baseline model suffers from the long-term model rollout, particularly for the flow within the magnetic field, the black-box baseline completely fails to capture the physics when $t > 2T$. In both scenarios, the PPNN outperforms the black-box baseline. In particular at the last time step $t = 3T$, which is three times the training phase length, the black-box predictions are totally overwhelmed by noise, while our PPNN predictions still agree with the reference very well. Compared to the case preserving the complete physics (Fig. 4d), a slight deviation from the reference solution can be observed in the PPNN predictions of the flow with an unknown magnetic field (Fig. 4e), indicating that incomplete prior knowledge could slightly affect the PPNN performance negatively. Nonetheless, preserving the partially-known PDE structure still brings significant merit. The error propagation is shown in Fig. 4f–g. The relative model rollout errors are averaged over 5 randomly selected unseen parameters for the systems with (Fig. 4f) and without (Fig. 4g) the magnetic field. Comparing to the PPNN with completely-known PDEs, the PPNN preserving incomplete/inaccurate prior knowledge does show a slight increment in the mean relative error ϵ_t as well as the error scattering, which implies a slight decrease in the robustness. However, the significant advantage over the black-box baseline remains, and almost no error accumulation is observed in PPNN for both scenarios.

When encoding completely mis-specified PDE terms. In the scenarios we have presented so far, the preserved PDE operators are incomplete but not entirely incorrect, which allows the PPNN model to outperform the black-box baseline. However, in certain situations, our prior knowledge about the target system may sometimes be entirely incorrect. In this section, we consider an extreme case where the preserved physics are completely mis-specified.

To investigate this, we consider a system governed by the viscous Burgers' equation (Eq.(6)), but we preserving a reaction
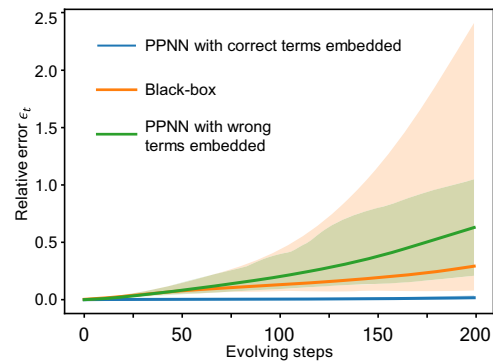
Fig. 5 Relative error ϵ_t comparison when wrong terms are embedded in partial differential equation preserved neural network (PPNN), tested on 2D Burgers' equation. The relative error of PPNN (blue line——), its black-box counterpart (orange line——), and PPNN with completely wrong partial differential equation terms (green line——) tested on unseen parameters is shown in the figure. Solid lines show the mean relative error, while the shaded areas show the distribution of all the 100 sample trajectories.

term (Eq.(4)) in the PPNN that does not reflect the actual physical processes at all. This experiment aims to assess our model's performance when the physics are completely mis-specified and determine how this mismatch affects the overall model performance.

These results show the model's behavior under the extreme conditions, when the underlying physics might be either completely unknown or inaccurately specified. As depicted in Fig. 5, the performance of the PPNN model suffers when the embedded PDE terms diverges significantly from the actual physics. In such cases, the performance of the PPNN model is adversely affected, with its predictions being worse than those of the black-box method. As expected, this result suggests that an certain level of alignment between the embedded PDEs and the underlying physics is essential for optimal performance. Particularly, the error distribution range of the PPNN model is significantly narrower than that of the black-box baseline, indicating that mis-specified embedded PDEs also impose an inductive bias to the deep learning model.

Training and inference cost. We have demonstrated that the proposed PPNN significantly improves the accuracy, generalizability, and robustness of next-step neural predictors by preserving the mathematical structure of the governing PDEs. Since the PPNN has a more complex network structure than the black-box baseline, it is worthwhile to discuss the training and inference costs of the PPNN and its comparison with the corresponding black-box baseline and the reference numerical solvers.

Training cost. As shown in Fig. 6a–c, the averaged relative (rollout) prediction error ϵ_T on n testing parameters λ at the last time step T in the training process ($n = 8$ in RD, $n = 6$ in Burgers and $n = 5$ in NS). For all the cases, PPNN features a significantly (orders of magnitude) lower error than the black-box model from a very early training stage. This means that, to achieve the same (if not higher) level of accuracy, our PPNN requires significantly less training cost compared to the black-box baseline. In addition, under the same training budget, the PPNN is much more accurate than the black-box baseline, demonstrating the merit of PPNN by leveraging the prior knowledge for network architecture design.

Inference cost. The inference costs of different neural networks and numerical solvers on the three testing cases (see section
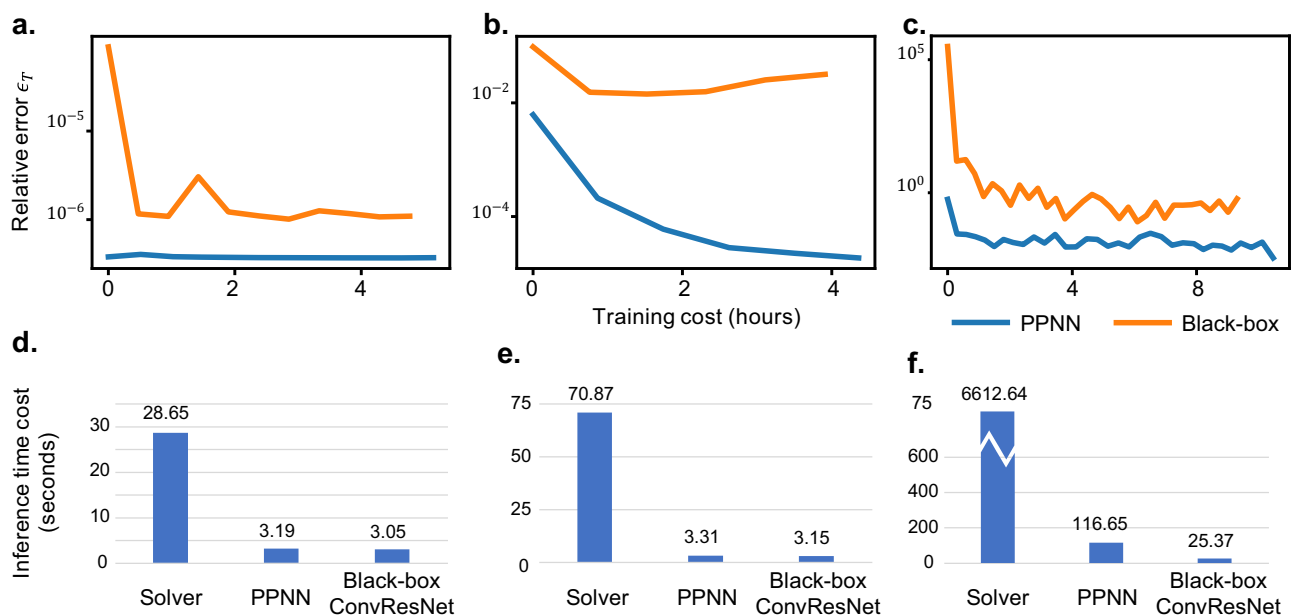
Fig. 6 Testing error ϵ_T during training and the inference cost of partial differential equation (PDE)-preserved neural network (PPNN), black-box baseline and numerical solvers. a–c Averaged relative test error at the last time step ϵ_T of PPNN (blue lines) and black-box ConvResNet (orange lines) in training process of different cases in the section When the governing PDEs are fully known (governed by reaction-diffusion (**a**) Burgers' equation (**b**) and Naiver-stokes equations (**c**)). **d–f** inference time cost of numerical solver, PPNN and black-box ConvResNet in the case (governed by reaction-diffusion (**d**) Burgers' equation (**e**), and Naiver-stokes equations (**f**)). The reaction-diffusion and Burgers cases are inferred (simulated) on a NVIDIA RTX 3070 GPU and the time is measured for infer/simulate 10 trajectories for 200 time steps. The Navier-Stocks case is inferred/simulated on a single of Intel Xeon Gold 6138 CPU and the time is measured for infer/simulate 1 trajectory for 219 time steps.

When the governing PDEs are fully known) with the model rollout length of T are summarized in Fig. 6b–f. Due to the fast inference speed of neural networks, both next-step neural models show significant speedup compared to the high-fidelity numerical solvers. In particular, the speedup by the PPNN varies from $10\times$ to $60\times$ without significantly sacrificing the prediction accuracy. Such speedup will become more tremendous considering a longer model rollout and enormous repeated model queries on a large number of different parameter settings, which are commonly required in many-query applications such as optimization design, inverse problems, and uncertainty quantification. Note that all models are compared on the same hardware (GPU or CPU) to eliminate the difference introduced by hardware. However, as most legacy numerical solvers can only run on CPUs, the speedup by neural models can be much more significant if they leverage massive GPU parallelism. Admittedly, adding the PDE-preserving part inevitably increases the inference cost compared to the black-box baseline, but the huge performance improvement by doing so outweighs the slight computational overhead, as demonstrated in section When the governing PDEs are fully known. We have to point out that the computation of the PDE-preserving portion is not fully optimized, particularly in the NS case, where low-speed I/O interactions reduce the overall speedup ratio compared to the numerical solver based on the mature CFD platform OpenFOAM. Further performance improvements are expected by customized code optimizations in future work.
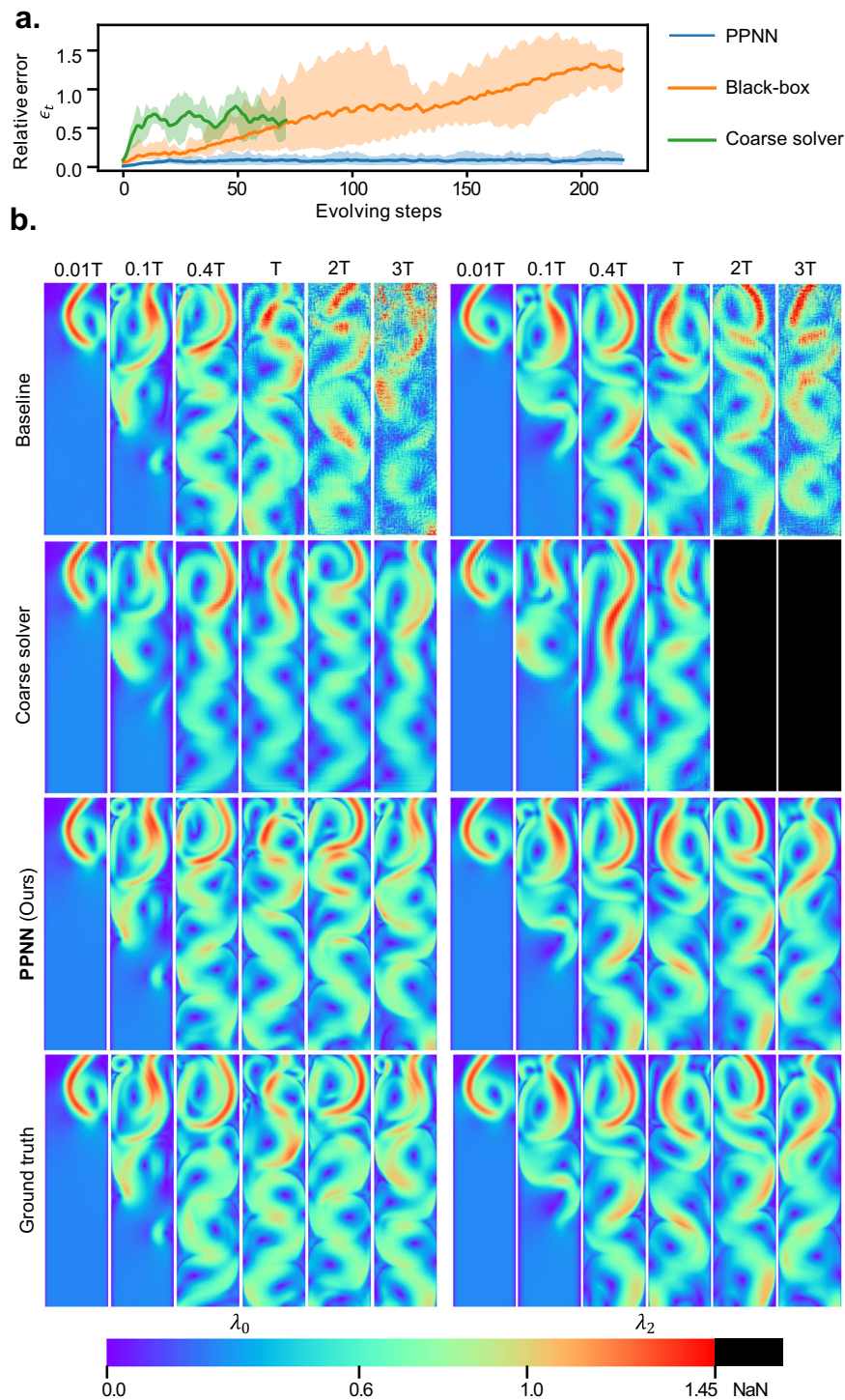
Relationship between the PDE-preserving portion and numerical solvers. The advantages of the proposed PPNN over the pure black-box baseline mainly come from "baking" the prior knowledge into the network architecture. As discussed above, the mathematical structures of the governing physics are encoded into the PPNN based on the relationship between neural network structures and differential equations. From the

numerical modeling perspective, if our understanding of the underlying physics is complete and accurate (i.e., complete governing PDEs are available), the PDE-preserving portion in PPNN can be interpreted as a numerical solver with the explicit forward Euler scheme defined on a coarse mesh. For simplicity, we here refer to this numerical solver derived from the fully-known PDE preserving part as the "coarse solver". It is interesting to see how well it performs by the coarse solver only when governing equations and IC/BCs/physics properties are fully known.

We use the NS case as an example. Figure 7b shows the magnitude of velocity $\|\boldsymbol{u}\|_2$ predicted by the PPNN, black-box ConvResNet and coarse solver, respectively, compared against the reference solution. Two representative testing parameters are studied here, one is at a lower Reynolds number Re $= 2500, y_0 = 0.325$ (Fig. 7b, λ_0), and the other is at a higher Reynolds number Re $= 9000, y_0 = 0.475$ (Fig. 7b, λ_2). It is clear that the predictions by the coarse solver noticeably deviate from the reference solution from $0.4T$, and most vortices are damped out due to the coarse spatial discretization. This becomes worse in the higher Reynolds number scenario, where the coarse solver predicted flow field is unphysical at $0.1T$ and the simulation completely diverged at $t = 1.16T$, because of the large learning timestep making traditional numerical solvers fail to satisfy the stability constraint.

As shown in the error propagation curves in Fig. 7a, the coarse solver has large prediction errors over the testing parameter set from the very beginning, which is much higher than that of the black-box data-driven baseline. Since several of the testing trajectories by the coarse solver diverges quickly after 70 evolving steps, the error propagation curve stops.

This figure again empirically demonstrates that the PPNN structure not only overcomes the error accumulation problem in black-box methods, but also significantly outperforms numerical solvers by simply coarsen the spatiotemporal grids. On the other
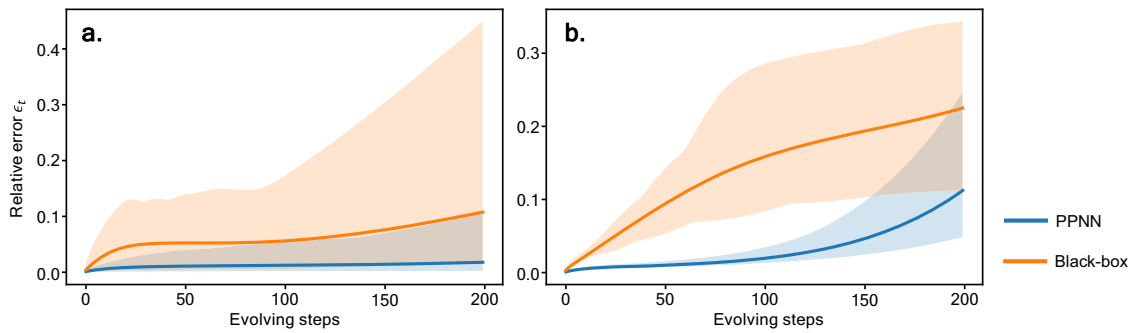
Fig. 7 Prediction comparison between partial differential equation (PDE)-preserved neural network (PPNN), the PDE-preserving part of PPNN (numerical solver results on a coarse mesh), the black-box baseline, and the label data. a Relative error at different time steps of PPNN (blue line——), Black-Box neural network (orange line——) and the coarse solver (green line——) compared to the ground truth results obtained by icoFoam on a fine mesh. The relative error is an averaged value of 5 test trajectories with randomly sampled parameters, these parameters are not in the training set. The shaded area shows the maximum and minimum relative error of these testing trajectories. In coarse solver, 2 of the testing trajectories diverged (NaN) at the 72nd step thus the green curve (——) stops at the 71st step. **b** The contours show predicted solution snapshots of velocity magnitude $\|u\|_2$ for the NS equations, obtained by black-box ConvResNet (baseline), PDE-preserving part only(coarse solver), and PPNN (ours); compared against ground truth (high-resolution numerical simulation), where $\lambda_0 = [2500, 0.325]^T$ and $\lambda_2 = [9000, 0.475]^T$, which are unseen in the training set. The black color indicates NaN, i.e., solution blow up.

hand, for those trajectories that do not diverge, the coarse solver's relative errors are limited to a certain level, which is in contrast to black-box, data-driven methods where the error constantly grows due to the error accumulation. This phenomenon implies that

preserving PDEs plays a critical role in addressing the issue of error accumulation, which does not simply provide a rough estimation of the next step, but carries underlying physics information that guides the longer-term prediction.

Fig. 8 Prediction comparison when partial differential equation preserved neural network (PPNN) using different deep neural networks as the trainable part. The relative error ϵ_t of 100 randomly sampled testing parameters λ is showed in this figure. The solid lines shows the averaged error over these 100 samples with shaded area shows the maximum and minimum relative errors of all testing trajectories. Blue (——) indicates the error of PPNN while orange (——) represents the corresponding black-box method. **a** shows the relative error of ViT and its PPNN variant. **b** shows the relative error of U-Net and its PPNN counterpart.

PPNN as a general framework for embedding known physics. In the previous sections, we have demonstrated the performance enhancement achieved by PPNN based on ConvResNet architecture. However, the proposed approach is not restricted to a particular neural network structures. In this section, we showcase the flexibility of PPNN as a general framework by integrating the PDE-preserving part into various DNN architectures, specifically U-Net and Vision Transformer (ViT). More details on the particular U-Net and ViT architectures employed in our study are provided in the supplementary information. To illustrate the versatility of PPNN, we tested it alongside its corresponding baseline in the context of the viscous Burgers' equation, as discussed in Section Viscous Burgers' equation.

Figure 8 presents the relative error ϵ_t gathered from 100 randomly selected unseen input parameters λ over 200 testing time steps for both the U-Net and ViT scenarios. In both cases, the PPNN variant considerably outperforms its black-box counterpart, achieving much lower relative errors. Furthermore, the error distributions of the PPNN exhibit narrower ranges in comparison to those of the baseline models. When compared to the ConvResNet (see Figure 2f), both the U-Net and ViT baselines exhibit significantly enhanced performance in terms of the average relative error. It is noteworthy that while the baselines exhibit improved performance, the PPNN variant demonstrates does not have the same performance gain, albeit still superior to their corresponding baseline models. This observation suggests a potential overfitting issue in the PPNN variant, warranting further investigation.

By successfully incorporating PPNN with a variety of DNN architectures and exhibiting its superior performance in the setting of the viscous Burgers' equation, we furnish compelling evidence that PPNN operates as a flexible framework for integrating known physics into deep neural networks. This underlines its potential for enhancing the predictive accuracy and robustness across various neural architectures.Moreover, our approach not only demonstrates compatibility with different neural networks but also shows impressive generalizability across varying boundary conditions. For additional insights into the application of PPNN on diverse boundary value problems, we invite readers to refer to the Section Supplementary Note 1 in the supplementary information.

Comparison with existing SOTA methods for neural operator learning. The backbone of the proposed PPNN method is a next-step auto-regressive model, which learns the transition dynamics of a spatiotemporal process, by mapping the solution fields from previous time steps to the next ones, and the whole trajectory

prediction is obtained by rolling out the learned transition model autoregressively. Since the PPNN prediction is also conditioned on parameters λ, the proposed model can be interpreted as learning an operator $\mathcal{G}$ in a discrete manner,

$$\mathcal{G} : \boldsymbol{u}(\boldsymbol{\xi}; \boldsymbol{\lambda}) \mapsto \mathcal{G}[\boldsymbol{u}(\boldsymbol{\xi}; \boldsymbol{\lambda})], \tag{13}$$

where $\boldsymbol{\xi} = [\boldsymbol{x}, t]$ represents spatial and temporal coordinates and $\boldsymbol{u} \in \mathbb{R}^n$ is the n-dimensional state variable. In addition to the auto-regressive formulation, one can directly learn the operator $\mathcal{G}$ using deep neural networks in a continuous manner, generally referred to as neural operators. In the past few years, several continuous neural operator learning methods have been proposed, e.g., DeepONet[25,68] and Fourier Neural Operator (FNO)[26]. Although many of them have shown great success for a handful of PDE-governed systems, it remains unclear how these methods perform compared to our proposed PPNN on the challenging scenarios studied in this work,

- Problems with high-dimensional parameter space, i.e., $\boldsymbol{\lambda} \in \mathbb{R}^d, d \gg 1$.
- Limited training data for good generalizability in parameter space and temporal domain.

Therefore, we conduct a comprehensive comparison of PPNN with existing state-of-the-art (SOTA) neural operators, including physics-informed neural network (PINN)[12], DeepONet[25,68], and Fourier Neural Operator (FNO)[26], on one of the previous test cases, Viscous Burgers' equation, where the PDE is fully known. (Strictly speaking, original PINN by Rassi et al.[12] is not an operator learner, but can be easily extended to achieving so by augmenting the network input layer with the parameter dimension, as shown in ref. [13]). For a fair comparison, the problem setting and training data remain the same for all the methods and the number of trainable parameters of each models are comparable (PINN: 1.94M parameters; DeepONet: 1.51M parameters; PPNN: 1.56M parameters. Please note in DeepONet, we used two separate but identical neural networks to learn the two components u_x, u_y of velocity $\boldsymbol{u}$ respectively to achieve optimal performance; each network contains 0.755M trainable parameters). Except for FNO, which has 0.58M trainable parameters due to the spatial Fourier transformation in FNO is too memory-hungry for a larger model to fit into the GPU used for training (RTX A6000 48GB RAM). It is worth noting that FNO could be formulated either as a continuous operator or as an autoregressive model. Here we show the performance of the continuous FNO. The performance of autoregressive FNO (named as aFNO) is shown in the Section Supplementary Note 3.6 supplementary
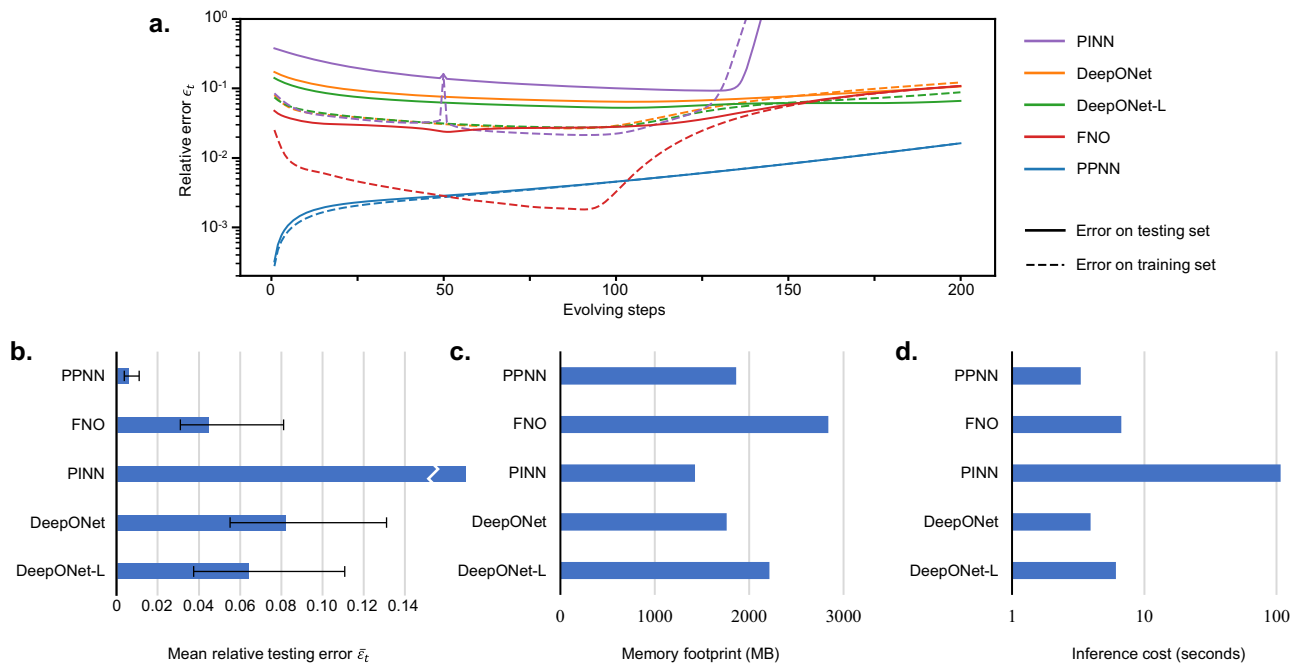
Fig. 9 Comparison between partial differential equation preserved neural network (PPNN) and various neural operators. a Comparison of the relative error ϵ_t of the physics-informed neural network (PINN, purple lines ——), deep operator networks (DeepONet, orange lines ——), DeepONet-L (green lines ——), Fourier neural operator (FNO, red lines ——), and PPNN (blue lines ——) of the velocity $\boldsymbol{u}$ in the viscous Burgers' equation evaluated on 100 randomly generated testing (unseen) parameters (solid lines), and 100 randomly selected training parameters (dashed lines). Only the first 100 time steps are used for training. Note that the y axis is in log scale. **b** Relative testing error averaged over all testing parameters and time steps $\bar{\epsilon}_t$ with error bar. Note that PINN has a much higher error than the other models which cannot be completely shown in this figure. The error bar here indicates the lowest and highest relative testing error ϵ_t among all the testing parameters, while the blue bar shows the mean relative testing error $\bar{\epsilon}_t$. **c** The memory footprint of different methods when testing 10 trajectories. Please note that, the time cost and memory footprint measured of PINN is the amount required for inferring a single row of the target mesh at a time. Inferring the whole field requires more memory, which exceeds the inference device's capacity. **d** The inference cost of testing 10 trajectories on a NVIDIA RTX 3070 GPU. Please note the inference time of PINN is measured for the inference-optimized variants of the original models, which are significantly faster than the original form used for training.

information, which is slightly better compared to continuous FNO in terms of testing error with unseen parameters. Besides, we also include a DeepONet with significantly more trainable parameters (79.19M) to show the highest possible performance DeepONet would achieve, which is named as DeepONet-L. Note that since some of these models' original forms cannot be directly applied to learn parametric spatiotemporal dynamics in multi-variable settings, necessary modifications and improvement has been made. The implementation details and hyperparameters of these models are provided in the supplementary information (see Supplementary Note 3).

Predictive performance comparison. All the models are used to predict the spatiotemporal dynamics of 100 randomly generated initial fields which are not seen in training. The relative prediction errors ϵ_t of the existing SOTA neural operators and PPNN are compared in Fig. 9. As shown in Fig. 9a, b, PPNN significantly outperforms all the other SOTA baselines for all the time steps in both training and testing parameter regimes. All the existing SOTA neural operators have much higher prediction errors (several orders of magnitude higher) compared to PPNN, especially when entering the extrapolation range (after 100 time steps), where the error grows rapidly. In contrast, the relative error of PPNN predictions remains very low (10^{-3}) and barely accumulated evolving with time (shown in Fig. 9a). The prediction errors of most continuous neural operators do not grow monotonically since their predictions do not rely on autoregressive model rollout and thus does not have error

accumulation issue. However, the overall accuracy of all continuous neural operators (particularly in extrapolation range) is much lower than that of PPNN. Besides, PPNN exhibits a much smaller error scattering over different testing samples (shown in Fig. 9b), indicating significantly higher robustness compared to existing SOTA methods. All of these observations suggest the obvious superiority of the PPNN in terms of extrapolability in time.

The comparison of the generalizability in parameter space of all methods is shown in Fig. 9a, where the dashed lines represent the averaged testing errors on the training parameter set, while the solid lines indicate the errors on the testing parameter set. It is clear that all the continuous neural operators have a significantly higher prediction error on the testing set than that on the training set, while the PPNN's prediction errors are almost the same on both the testing and training sets, which are much lower than all the other methods, indicating a much better generalizability.

It is worth mentioning that the notable overfitting issue is observed in DeepONet with increased trainable parameters, i.e., DeepONet-L. It can be seen that although the prediction errors of DeepONet-L and FNO are relatively lower on training parameter sets and interpolation regimes, they rapidly increase when stepping into extrapolation ranges and unseen parameter regimes. We would like to point out that there are "physics-informed" variants of DeepONet and FNO[27,69], which regulate the DNN training by minimizing the residual of governing PDEs, in conjunction with data loss. However, these approaches typically

necessitate the knowledge of the complete equation forms to formulate the physics-informed loss, while our method excel at integrating partially known physics, such as individual PDE operators, into the neural network structures. Moreover, the challenge of balancing DNN training with equation loss and label data is a well-documented issue, often requiring sophisticated hyperparameter tunning to adjust the weights between equation loss and data loss[30,70,71]. In addition, the use of equation loss for problems with a high-dimensional parameter space poses a significant challenge in minimizing the composed loss function, leading to marginal and often unstable improvement over purely data-driven methods. We provide a more detailed comparison and discussion regarding the performance of these physics-informed variants of FNO/DeepONet in the supplementary information (see Supplementary Note 5).

Cost comparison. Figure 9c and d show the time cost and memory footprint in the inference phase. Even compared to the fastest baseline DeepONet, PPNN is still about 20% faster and the memory footprint of PPNN is very close to the model with the smallest memory footprint: DeepONet. It should be noted that all the models, including our PPNN are not exhaustively fine-tuned. Although carefully tuning hyperparameters may further improve the performance of each model, issues such as generalizability or robustness presented above cannot be addressed by hyperparameter tuning.

Limitations and future directions. Despite the significant advancements of the PPNN model, it is important to acknowledge some inherent limitations and potential areas of improvement. One such limitation is the minor accumulation of error when extrapolating over a significantly long duration. This can be attributed to the one-step prediction formulation currently utilized by the PPNN model. It is worth noting that the trainable aspect of PPNN is mathematically equivalent to learning closure models under the current design. However, a crucial distinction between our proposed method and existing closure model approaches lies in the ability to propagate gradients through PPNN, encompassing both the PDE-preserving segment and the trainable portion. This capability arises from our representation of physics priors through convolutional neural networks, making it differentiable. This differentiable feature empowers us to conduct end-to-end training with long-term model rollouts, an achievement unattainable in traditional closure model learning, where the physics priors (i.e., numerical solvers) lack differentiability, often necessitating direct labels for the discrepancy of such prior models. More importantly, this study underscores the connection between numerical PDE operators and neural architecture components, which can pave the way for innovative neural solver designs that go beyond classic PDE solvers augmented with DNN closures. From this viewpoint, conventional numerical PDE solvers can be conceptualized as a specific instance of neural networks. Their architecture details, including elements like convolution kernels, residual connections, or recurrent structure, are completely determined by the governing PDEs and their associated numerical schemes. In contrast, fully trainable neural networks are completely flexible and derive their structural parameters purely from data. Nonetheless, it is important to note that in this work, we did not fully explore the potential of PPNN through extended model rollouts during training. Conventional time-marching solvers operate based on predefined time integration schemes. In contrast, the PPNN framework offers the potential to weave various numerical PDE operators and trainable components to construct a modern DL architecture such as LSTM or transformer, clearly a departure

from standard PDE solvers. This paper represents our initial step into the field of differentiable hybrid neural modeling, primarily aiming to explore and demonstrate the merit of PDE-integrated neural models. As such, the design and comparison of various hybrid PDE-neural architectures fall outside the scope of this work.

Our present work primarily focuses on structured data or meshes, a choice driven by their simplicity and computational efficiency. These are commonly used in many computational physics problems with relatively simple geometries. The novelty of our approach lies in the innovative use of structured meshes to design a PDE-preserving neural network. This is accomplished by mapping known PDE operators onto convolutional filters, thereby transposing the laws of physics into the language of deep learning. However, this focus on structured data does not mean that our model is inherently limited to such data. We see our demonstration on structured meshes as an essential first step towards extending the approach to more complex geometries and unstructured data. Addressing concerns about the applicability of our method to unstructured data and irregular geometries, we note that our current method could be extended using graph neural networks. The graph convolution operation, interpreted as a localized spectral filtering on unstructured data, can be viewed as a generalization of the CNN's convolution operation. By carefully designing spectral filters, the concept of "PDE-preserving" can be incorporated into desired spatial PDE operators through finite-volume-based or finite-element-based kernel functions. Although such an extension would require rigorous mathematical derivations and extensive empirical studies, we believe it serves as an intriguing direction for future research.

Furthermore, the ConvResNet formulation in the current version of PPNN is not mesh-invariant due to the discrete convolution operation, suggesting it cannot directly process data represented on different meshes without interpolations. However, the proposed PPNN framework can be extended to accommodate mesh invariance. One potential way to achieve this is to use mesh-invariant convolutional layers, which apply the same operations to the input data regardless of the underlying mesh structure. This could be realized, for instance, by employing geodesic convolutions or graph convolution kernel in spectral domain, allowing the model to adapt to variations in the mesh resolutions. Additionally, integrating adaptive mesh refinement techniques into the training process might provide another route towards mesh invariance. This strategy would involve dynamically adjusting the mesh resolution by incorporating mesh info Δx into the model, allowing the model to capture the mesh variations.

In real-world applications, training data can be gathered from experiments or in-situ sensing, where data uncertainty may arise due to measurement noises in both inputs and training labels. Our current PPNN model does not include an uncertainty quantification (UQ) capability, but uncertainty propagation and quantification represent fascinating directions for future research. Extending the PPNN model to incorporate Bayesian learning could be a potential solution. Techniques like Bayesian neural networks using variational inference[72–75] or deep ensemble methods[76–78] may offer promising avenues for expanding the PPNN model to include UQ capabilities.

Spatiotemporal dynamics constitute a fundamental aspect of numerous physics systems, ranging from classical fields like fluid dynamics, acoustics, and electromagnetics to the intricate realm of Quantum mechanics. The governing equations for such dynamics often fall within the domain of partial differential equations. Consequently, the ability to effectively solve these PDEs is imperative for comprehending, modeling, and

controlling the underlying physical processes. By integrating the PDE structure into deep neural networks, PPNN represents a powerful tool for modeling such PDEs. In the context of various physics applications, PPNN exhibits considerable potential. Contrasted with traditional numerical solvers or earlier physics-informed neural networks, PPNN demonstrates lower training and inferring cost and the capacity to assimilate unknown physics from data. Additionally, when compared to purely data-driven methods, PPNN provides enhanced accuracy in out-of-sample scenarios while maintaining stability over prolonged model rollouts. The versatile nature of PPNN makes it a promising candidate for applications in modeling and predicting dynamic physics systems, including heat transfer, turbulent flow, and electromagnetic fields. While not delving into the specifics of each application, it is evident that PPNN holds significant promise for speeding up the study and understanding of complex spatiotemporal dynamics across various physics domains.

Conclusion

In this work, we proposed a physics-inspired deep learning framework, PDE-preserved neural network (PPNN), aiming to learn parametric spatiotemporal physics, where the (partially) known governing PDE structures are preserved via fixed convolutional residual connection blocks in a multi-resolution setting. The PDE-preserving ConvResNet blocks together with trainable blocks in an encoding-decoding manner bring the PPNN significant advantages in long-term model rollout accuracy, spatiotemporal/parameter generalizability, and training efficiency. The effectiveness and merit have been demonstrated over a handful of challenging spatiotemporal prediction tasks, including the FitzHugh-Nagumo reaction diffusion equations, viscous Burgers equations and Naiver-Stokes equations, compared to the existing baselines, including ConvResNet, U-Net, Vision transformer, PINN, DeepONet, and FNO. The proposed PPNN shows satisfactory predictive accuracy in testing regimes and significantly lower error-accumulation effect for long-term model rollout in time, even if the preserved physics is incomplete or inaccurate. Finally, the discussion on the inference and training costs shows the great potential of the proposed model to serve as a reliable and efficient surrogate model for spatiotemporal dynamics in many applications that require repeated model queries, e.g., design optimization, data assimilation, uncertainty quantification, and inverse problems. While Direct Numerical Simulations (DNS) are used as the source of labeled training data in our study, the data could just as well originate from experimental results or field observations. A unique feature of PPNN, and one of its significant advances, lies in its ability to generalize to different physical parameters and initial/boundary conditions. Unlike most label-free PINN techniques, which act as PDE solvers for a given set of parameters and conditions, PPNN's ability to adapt to varying parameters and conditions underscores its capability to learn the PDE system. In general, this work explored a creative design of leveraging physics-inductive bias in scientific machine/deep learning and showcased how to use physical prior knowledge to inform the learning architecture design, shedding new light on physics-informed deep learning from a different aspect. Therefore, this work represents a inventive PiDL development and a significant advance in the realm of SciML.

METHODS

Problem formulation. We are interested in predictive modeling of physical systems with spatiotemporal dynamics, which can be described by a set of parametric coupled PDEs in the general form,

$$\frac{\partial \boldsymbol{u}}{\partial t} + \mathscr{F}\left[\boldsymbol{u}, \boldsymbol{u}^2, \dots, \nabla_x \boldsymbol{u}, \nabla_x^2 \boldsymbol{u}, \nabla_x \boldsymbol{u} \cdot \boldsymbol{u}, \dots ; \boldsymbol{\lambda}\right] = \boldsymbol{0}, \qquad \boldsymbol{x}, t \in \Omega \times [0, T], \boldsymbol{\lambda} \in \mathbb{R}^d,$$

(14a)

$$\mathcal{I}\left[\boldsymbol{x}, \boldsymbol{u}, \nabla_x^2 \boldsymbol{u}, \nabla_x \boldsymbol{u} \cdot \boldsymbol{u}; \boldsymbol{\lambda}\right] = \boldsymbol{0}, \qquad \boldsymbol{x} \in \Omega, t = 0, \boldsymbol{\lambda} \in \mathbb{R}^d,$$

(14b)

$$\mathcal{B}\left[t, \boldsymbol{x}, \boldsymbol{u}, \nabla_x^2 \boldsymbol{u}, \nabla_x \boldsymbol{u} \cdot \boldsymbol{u}; \boldsymbol{\lambda}\right] = \boldsymbol{0}, \qquad \boldsymbol{x}, t \in \partial\Omega \times [0, T], \boldsymbol{\lambda} \in \mathbb{R}^d,$$

(14c)

where $\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{x}, t; \boldsymbol{\lambda}) \in \mathbb{R}^n$ is the n-dimensional state variable; t denotes time and $\boldsymbol{x} \in \Omega$ specifies the space; $\mathscr{F}[\cdot]$ is a complex nonlinear functional governing the physics, while differential operators $\mathcal{I}[\cdot]$ and $\mathcal{B}[\cdot]$ describe the initial and boundary conditions (I/BCs) of the system, respectively; $\boldsymbol{\lambda} \in \mathbb{R}^d$ is a d-dimensional vector, representing physical/modeling parameters in the governing PDEs and/or I/BCs. Solving this parametric spatiotemporal PDE system typically relies on traditional FD/FV/FE methods, which are computationally expensive in most cases. This is due to the spatiotemporal discretization of the PDEs into a high-dimensional algebraic system, making the numerical simulation time-consuming, particularly considering that a tiny step is often required for the time integration to satisfy the numerical stability constraint. Moreover, as the system solution $\boldsymbol{u}(\boldsymbol{x}, t; \boldsymbol{\lambda})$ is parameter-dependent, we have to start over and conduct the entire simulation given a new parameter $\boldsymbol{\lambda}$, making it infeasible for application scenarios that require many model queries, e.g., parameter inference, optimization, and uncertainty quantification. Therefore, our objective is to develop a data-driven neural solver for rapid spatiotemporal prediction, enabled by efficient time-stepping with coarse-gaining and fast inference speed of neural networks. In particular, this study focuses on the learning architecture design by preserving known PDE structures for improving the robustness, stability, and generalizability of data-driven auto-regressive predicting models.

**Next-step prediction models based on convolutional ResNets**. The next-step DNN predictors are commonly used for emulating spatiotemporal dynamics in an autoregressive manner,

$$\boldsymbol{u}_t = f_\theta(\boldsymbol{u}_{t-1}, \boldsymbol{\lambda} | \boldsymbol{\theta}),$$

(15)

where the state solution $\boldsymbol{u}_t$ at time step t is approximated by a neural network function $f_\theta : \mathbb{R}^n \times \mathbb{R}^d \to \mathbb{R}^n$, taking the previous state $\boldsymbol{u}_{t-1}$ and physical parameters $\boldsymbol{\lambda}$ as the input features. The function $f_\theta(\cdot | \boldsymbol{\theta})$ is parameterized by trainable weight vector $\boldsymbol{\theta}$ that can be optimized based on training labels. Once the model is fully trained, it can be used to predict spatiotemporal dynamics by autoregressive model rollouts given only the initial condition $\boldsymbol{u}_0$ and a specific set of physical parameters $\boldsymbol{\lambda}$. In general, the next-step model is built based on residual network (ResNet) blocks, which have recently improved the state-of-the-art (SOTA) in many benchmarks learning tasks[9]. Given the input features $\boldsymbol{z}_0$, a ResNet block with N layers outputs $\boldsymbol{z}_N$ as,

$$\boldsymbol{z}_{j+1} = \boldsymbol{z}_j + f^{(j)}(\boldsymbol{z}_j | \boldsymbol{\theta}^{(j)}), j = 0, \cdots N - 1,$$

(16)

where $f^{(j)}$ represents the generic neural network function of jth layer and $\boldsymbol{\theta}^{(j)}$ are corresponding weights. For end-to-end spatiotemporal learning, $f^{(j)}$ are often formulated by (graph) convolutional neural networks with trainable convolution stencils and biases. In a ResNet block, the dimension of the feature vectors (i.e., the image resolution and the number of channels) should remain the same across all layers. The ResNet-based next-step models have been demonstrated powerful and effective for predicting complex spatiotemporal physics. One of the examples is the MeshGraphNet[9], which is a GNN-based ResNets and shows
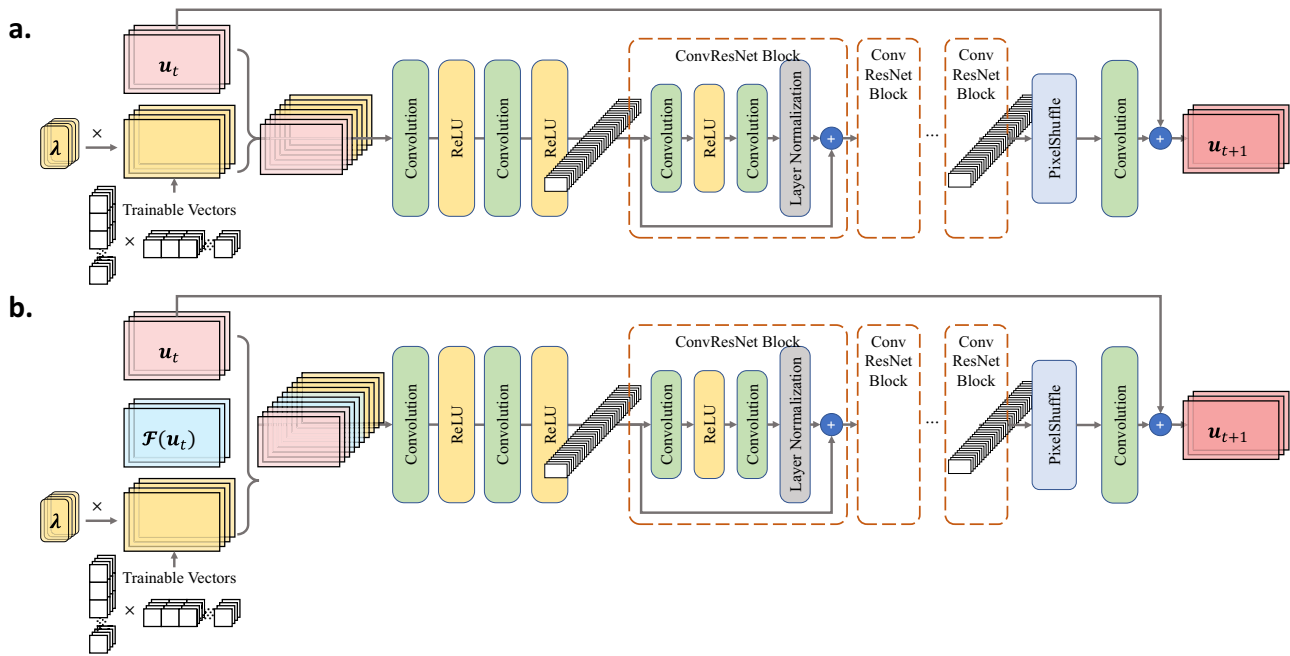
Fig. 10 Schematics of the deep neural network structures used in this work. a Network architecture of the baseline black-box ConvResNet-based next-step model. **b** Network architecture of the trainable portion of partial differential equation (PDE)-preserved neural network (PPNN). The only difference between them is trainable portion of PPNN has an extra input variable $\mathcal{F}(\boldsymbol{u}_t)$, provided by the PDE-preserving portion of PPNN.

the SOTA performance in spatiotemporal learning with unstructured mesh data.

In this work, as we limit ourselves to structured data within regular domains, a CNN variant of the MeshGraphNet, Convolutional ResNet (ConvResNet)-based next-step model, is used as one of the baseline black-box models in this work, whose network structure is shown in Fig. 10a. The ConvResNet takes the previous state and physical parameters as the input and predicts the next-step state using a residual connection across the entire hidden ConvResNet layers after a pixel shuffle layer. The hidden layers consist of several ConvResNet blocks, constructed based on standard convolution layers with residual connections and ReLU activation functions, followed by layer normalization. To learn the dependence of physical parameters $\boldsymbol{\lambda}$, each scalar component of the physical parameter vector is multiplied by a trainable matrix, which is obtained by vector multiplication of trainable weight vectors.

Neural network architecture and differential equations. Recent studies have shown the relationship between DNN architectures and differential equations: ResNets can be interpreted as discretized forms of ODEs/PDEs, while differential equations can be treated as a continuous interpretation of ResNet blocks with infinite depth.

Residual connections and ODEs. As discussed in[44,50], the residual connection as defined in Eq. (16) can be seen as a forward Euler discretization of a ODE,

$$\frac{\partial z(t)}{\partial t} = F(z(t)|\boldsymbol{\theta}(t)), \text{for} t \in (0, T], \qquad (17)$$

where $z(t=0) = z_0$ and T is total time. In ResNets, a fixed time step size of $\Delta t = 1$ is set for the entire time span and $N \cdot \Delta t = T$. Namely, the depth of the residual connection (i.e., the number of layers in a ResNet block) can be controlled by changing the total time T. On the other hand, an ODE as given by Eq. (17) can be interpreted as a continuous ResNet block with infinite number of

layers (i.e., infinite depth). Based on this observation, the classic ResNet structure can be extended by discretizing an ODE using different time-stepping schemes (e.g., Euler, Runge-Kutta, leap-frog, etc.). Moreover, we can also define a residual connection block by directly coupling a differentiable ODE solver with a multi-layer perception (MLP) representing $F(\cdot)$, where the hybrid ODE-MLP is trained as a whole differentiable program using back-propagation, which is known as a neural-ODE block[50].

Convolution operations and PDEs. In the neuralODE, MLP is used to define $F(\cdot)$, which, however, can be any neural network structure in a general setting. When dealing with structured data (e.g., images, videos, physical fields), the features $z(t, \boldsymbol{x})$ can be seen as spatial fields, and convolution operations are often used to construct a CNN-based $F(\cdot)$. A profound relationship between convolutions and differentiations has been presented in refs. [45,79,80]. Following the deep learning convention, a 2D convolution is defined as,

$$\text{conv}\left(\boldsymbol{z}, h^{(\theta)}\right) = \int \boldsymbol{z}(\boldsymbol{x}' - \boldsymbol{x}) h^{(\theta)}(\boldsymbol{x}) d\boldsymbol{x}, \qquad (18)$$

where h represent convolution kernel parameterized by $\boldsymbol{\theta}$. Based on the order of sum rules, the kernel h can be designed to approximate any differential operator with prescribed order of accuracy[81], and thus the convolution in Eq. (18) can be expressed as[49],

$$\text{conv}\left(\boldsymbol{z}, h^{(\theta)}\right) = \mathscr{D}\left[\boldsymbol{u}, \dots, \nabla_{\boldsymbol{x}}\boldsymbol{u}, \nabla_{\boldsymbol{x}}^2\boldsymbol{u}, \nabla_{\boldsymbol{x}}\boldsymbol{u} \cdot \boldsymbol{u}, \dots; \boldsymbol{\theta}\right], \qquad (19)$$

where $\mathscr{D}$ is a discrete differential operator based FD/FV/FE methods. For example, from the point of view of FDM, convolution filters can be seen as the finite difference stencils of certain can be interpreted as the discrete forms of certain PDEs, and thus the PDEs can be used to inform ConvResNet architecture design.

Multi-resolution PDE-preserved Neural Network (PPNN) architecture. It is well known that auto-regressive models suffer from error accumulation, which is particularly severe for the next-step formulation. Although remedies such as using training noises[82] or sequence models[10] have been explored, the error accumulation issue cannot be easily mitigated, and the model usually fails to operate in a long-span rollout. Inspired by the relationship between network architectures and differential equations, we hypothesize that the performance of an auto-regressive ConvResNet model for spatiotemporal learning can be significantly improved if the network is constructed by preserving (partially) known governing physics (i.e., PDEs) of the spatio-temporal dynamics. Therefore, we propose a multi-resolution PDE-preserved neural network (PPNN) framework, where the discrete governing PDEs are preserved in residual connection blocks using grids with multiple resolutions.

As shown in Fig. 1, the PPNN has the same backbone ResNet structure as the black-box next-step baseline model, where a residual connection is applied across the entire hidden ConvResNet layers. The hidden ConvResNet consists of two portions: PDE-preserving ConvRes layers and trainable ConvRes layers, coupled in an encoding-decoding manner. In the PDE-preserving portion, the ConvRes connection is constructed based on the convolution operators defined by the discrete differential operators of the governing PDEs using finite-difference stencils. The preserved-PDE ConvRes layers are operated on low-resolution grids by taking in the downsampled input solution fields using bi-linear algorithm and the output is upsampled back to the original resolution using bi-cubic algorithm, which improves the model rollout stability with large evolving steps, meanwhile reducing the cost overhead during the model inference. This structure resembles the multgrid method which significantly improves the speed and reduce the cost by solving PDEs on different mesh resolutions. The trainable portion takes the high-resolution solution fields, together with the output of the PDE preserving part, as the input and contains a few classic ConvResNet blocks. For a fair comparison, the network architecture of the trainable portion is exactly the same as that of the black-box ConvResNet baseline, except that the trainable portion of PPNN takes the output of the PDE-preserving portion (see Fig. 10). The PDE-preserving part and trainable part are connected via bi-cubic up-sampling operation. Overall, the PDE-preserving part enhance the trainable part by (a) preserving a time integration scheme (b) providing input feature enrichment. An ablation study of these two components can be found in the section Supplementary Note 2 in supplementary information. Note that a smaller time step $\Delta t'$ can be used within the PDE-preserving portion via inner-iteration to stabilize model rollout. In general, the combination of the two portions can be seen as a ConvResNet architecture that preserves the mathematical structure of the underlying physics behind the spatiotemporal dynamics to be modeled.

Data availability

All the used datasets in this study can be generated by the openly available Python scripts on GitHub at https://github.com/jx-wang-s-group/ppnn upon publication.

Code availability

All the source codes to reproduce the results in this study will be openly available on GitHub at https://github.com/jx-wang-s-group/ppnn upon publication.

References

1. Lui, HugoF. S. & Wolf, W. R. Construction of reduced-order models for fluid flows using deep feedforward neural networks. *J. Fluid Mech.* **872**, 963–994 (2019).
2. San, O., Maulik, R. & Ahmed, M. An artificial neural network framework for reduced order modeling of transient flows. *Commun. Nonlin. Sci. Numer. Simulat.* **77**, 271–287 (2019).
3. Gao, H., Wang, Jian-Xun & Zahr., M. J. Non-intrusive model reduction of large-scale, nonlinear dynamical systems using deep learning. *Phys. D Nonlin. Phenom.* **412**, 132614 (2020).
4. Fresca, S. & Manzoni, A. Pod-dl-rom: enhancing deep learning-based reduced order models for nonlinear parametrized pdes by proper orthogonal decomposition. *Comp. Methods Appl. Mech. Eng.* **388**, 114181 (2022).
5. Murata, T., Fukami, K. & Fukagata, K. Nonlinear mode decomposition with convolutional neural networks for fluid dynamics. *J. Fluid Mech.* **882**, A13 (2020).
6. Mohan, A. T., Tretiak, D., Chertkov, M. & Livescu, D. Spatio-temporal deep learning models of 3d turbulence with physics informed diagnostics. *J. Turbul.* **21**, 484–524 (2020).
7. Maulik, R., Lusch, B. & Balaprakash, P. Reduced-order modeling of advection-dominated systems with recurrent neural networks and convolutional autoencoders. *Phys. Fluids* **33**, 037106 (2021).
8. Fukami, K., Hasegawa, K., Nakamura, T., Morimoto, M. & Fukagata, K. Model order reduction with neural networks: Application to laminar and turbulent flows. *SN Comp. Sci.* **2**, 1–16 (2021).
9. Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A. & Battaglia, P. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations* 4521–4622 (ICLR, 2020).
10. Han, X., Gao, H., Pfaff, T., Wang, J. X. & Liu, L. Predicting physics in mesh-reduced space with temporal attention. In *International Conference on Learning Representations* 4632–4653 (ICLR, 2022).
11. Baker, N. et al. Workshop report on basic research needs for scientific machine learning: *Core Technologies for Artificial Intelligence* (USDOE Office of Science, 2019).
12. Raissi, M., Perdikaris, P. & Karniadakis, G. E. Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* **378**, 686–707 (2019).
13. Sun, L., Gao, H., Pan, S. & Wang, Jian-Xun Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Comp. Methods Appl. Mech. Eng.* **361**, 112732 (2020).
14. Zhang, R., Liu, Y. & Sun, H. Physics-informed multi-lstm networks for metamodeling of nonlinear structures. *Comp. Methods Appl. Mech. Eng.* **369**, 113226 (2020).
15. Haghighat, E., Raissi, M., Moure, A., Gomez, H. & Juanes, R. A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics. *Comp. Methods Appl. Mech. Eng.* **379**, 113741 (2021).
16. Sun, L. & Wang, Jian-Xun Physics-constrained bayesian neural network for fluid flow reconstruction with sparse and noisy data. *Theoret. Appl. Mech. Lett.* **10**, 161–169 (2020).
17. Arzani, A., Wang, Jian-Xun & D'Souza, R. M. Uncovering near-wall blood flow from sparse data with physics-informed neural networks. *Phys. Fluids* **33**, 071905 (2021).
18. Lu, L. et al. Physics-informed neural networks with hard constraints for inverse design. *SIAM J. Sci. Comp.* **43**, B1105–B1132 (2021).
19. Zhang, E., Dao, M., Karniadakis, GeorgeEm & Suresh, S. Analyses of internal structures and defects in materials using physics-informed neural networks. *Sci. Adv.* **8**, eabk0644 (2022).
20. Han, J., Jentzen, A. & E, W. Solving high-dimensional partial differential equations using deep learning. *Proc. Natl Acad. Sci.* **115**, 8505–8510 (2018).
21. Zhang, D., Lu, L., Guo, L. & Karniadakis, GeorgeEm Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems. *J.Comput. Phys.* **397**, 108850 (2019).
22. Yang, Y. & Perdikaris, P. Adversarial uncertainty quantification in physics-informed neural networks. *J. Comput. Phys.* **394**, 136–152 (2019).
23. Kharazmi, E., Zhang, Z. & Karniadakis, GeorgeEm hp-vpinns: Variational physics-informed neural networks with domain decomposition. *Comp. Methods Appl. Mech. Eng.* **374**, 113547 (2021).
24. Jagtap, A. D., Kharazmi, E. & Karniadakis, GeorgeEm Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. *Comp. Methods Appl. Mech. Eng.* **365**, 113028 (2020).
25. Lu, L., Jin, P., Pang, G., Zhang, Z. & Karniadakis, GeorgeEm Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nat. Mach. Intell.* **3**, 218–229 (2021).
26. Li, Z. et al. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations* 4596–4621 (ICLR, 2020).

27. Wang, S., Wang, H. & Perdikaris, P. Learning the solution operator of parametric partial differential equations with physics-informed deeponets. *Sci. Adv.* **7**, eabi8605 (2021).

28. Goswami, S., Yin, M., Yu, Y. & Karniadakis, GeorgeEm A physics-informed variational deeponet for predicting crack path in quasi-brittle materials. *Comp. Methods Appl. Mech. Eng.* **391**, 114587 (2022).

29. Jagtap, A. D., Kawaguchi, K. & Karniadakis, GeorgeEm Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. *J. Comput. Phys.* **404**, 109136 (2020).

30. Wang, S., Yu, X. & Perdikaris, P. When and why pinns fail to train: a neural tangent kernel perspective. *J. Comput. Phys.* **449**, 110768 (2022).

31. Wang, S., Sankaran, S. & Perdikaris, P. Respecting causality is all you need for training physics-informed neural networks. *arXiv* https://doi.org/10.48550/arXiv.2203.07404 (2022).

32. Gao, H., Sun, L. & Wang, Jian-Xun PhyGeoNet: physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain. *J. Comput. Phys.* **428**, 110079 (2021).

33. Ren, P., Rao, C., Liu, Y., Wang, Jian-Xun & Sun, H. Phycrnet: Physics-informed convolutional-recurrent network for solving spatiotemporal pdes. *Comp. Methods Appl. Mech. Eng.* **389**, 114399 (2022).

34. Geneva, N. & Zabaras, N. Modeling the dynamics of pde systems with physics-constrained deep auto-regressive networks. *J. Comput. Phys.* **403**, 109056 (2020).

35. Gao, H., Sun, L. & Wang, Jian-Xun Super-resolution and denoising of fluid flow using physics-informed convolutional neural networks without high-resolution labels. *Phys. Fluids* **33**, 073603 (2021).

36. Wandel, N., Weinmann, M. & Klein, R. Teaching the incompressible navier–stokes equations to fast neural surrogate models in three dimensions. *Phys. Fluids* **33**, 047117 (2021).

37. Ranade, R., Hill, C. & Pathak, J. Discretizationnet: A machine-learning based solver for navier–stokes equations using finite volume discretization. *Comp. Methods Appl. Mech. Eng.* **378**, 113722 (2021).

38. Yao, H., Gao, Y. & Liu, Y. Fea-net: A physics-guided data-driven model for efficient mechanical response prediction. *Comp. Methods Appl. Mech. Eng.* **363**, 112892 (2020).

39. Mitusch, S. K., Funke, S. W. & Kuchta, M. Hybrid fem-nn models: combining artificial neural networks with the finite element method. *J. Comput. Phys.* **446**, 110651 (2021).

40. Wang, Z., Huan, X. & Garikipati, K. Variational system identification of the partial differential equations governing microstructure evolution in materials: Inference over sparse and spatially unrelated data. *Comp. Methods Appl. Mech. Eng.* **377**, 113706 (2021).

41. Minglang, Y., Zhang, E., Yu, Y. & Karniadakis, GeorgeEm Interfacing finite elements with deep neural operators for fast multiscale modeling of mechanics problems. *Comp. Methods Appl. Mech. Eng.* **402**, 115027 (2022).

42. Gao, H., Zahr, M. J. & Wang, Jian-Xun Physics-informed graph neural galerkin networks: a unified framework for solving pde-governed forward and inverse problems. *Comp. Methods Appl. Mech. Eng.* **390**, 114502 (2022).

43. Liu, Xin-Yang & Wang, Jian-Xun Physics-informed dyna-style model-based deep reinforcement learning for dynamic control. *Proc. Royal Soc. A Math. Phys. Eng. Sci.* **477**, 20210618 (2021).

44. Haber, E. & Ruthotto, L. Stable architectures for deep neural networks. *Inver. Probl.* **34**, 014004 (2017).

45. Lu, Y., Zhong, A., Li, Q. & Dong, B. Beyond finite layer neural networks: bridging deep architectures and numerical differential equations. In *International Conference on Machine Learning* 3276–3285 (PMLR, 2018).

46. Rousseau, François, Drumetz, L. & Fablet, R. Residual networks as flows of diffeomorphisms. *J. Math. Imag. Vision* **62**, 365–375 (2020).

47. Ruthotto, L. & Haber, E. Deep neural networks motivated by partial differential equations. *J. Math. Imag. Vision* **62**, 352–364 (2020).

48. Chamberlain, B. et al. Grand: graph neural diffusion. In *International Conference on Machine Learning* 1407–1418 (PMLR, 2021).

49. Eliasof, M., Haber, E. & Treister, E. PDE-GCN: Novel architectures for graph neural networks motivated by partial differential equations. *Adv. Neur. Inform. Process. Syst.* **34**, 3836–3849 (2021).

50. Chen, R., Rubanova, Y., Bettencourt, J. & Duvenaud, D. K. Neural ordinary differential equations. *Advances in Neural Information Processing Systems* 31 (NeuralPS, 2018).

51. He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* 770–778 (IEEE, 2016).

52. Gholami, A., Keutzer, K. & Biros, G. Anode: unconditionally accurate memory-efficient gradients for neural odes. *arXiv* https://doi.org/10.48550/arXiv.1902.10298 (2019).

53. Shi, Z., Gulgec, N. S., Berahas, A. S., Pakzad, S. N. & Takac, M. Finite difference neural networks: fast prediction of partial differential equations. In *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)* 130–135 (IEEE, 2020).

54. Innes, M. et al. A differentiable programming system to bridge machine learning and scientific computing. *arXiv* https://doi.org/10.48550/arXiv.1907.07587 (2019).

55. Rackauckas, C. et al. Universal differential equations for scientific machine learning. *arXiv* https://doi.org/10.48550/arXiv.2001.04385 (2020).

56. Sun, Y., Zhang, L. & Schaeffer, H. Neupde: neural network based ordinary and partial differential equations for modeling time-dependent data. In *Mathematical and Scientific Machine Learning* 352–372 (PMLR, 2020).

57. Hochlehnert, A., Terenin, A., Saemundsson, S. & Deisenroth, M. Learning contact dynamics using physically structured neural networks. In *International Conference on Artificial Intelligence and Statistics* 2152–2160 (PMLR, 2021).

58. Heiden, E., Millard, D., Coumans, E., Sheng, Y. & Sukhatme, G. S. Neuralsim: augmenting differentiable simulators with neural networks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)* 9474–9481 (IEEE, 2021).

59. Hackenberg, M. et al. Using differentiable programming for flexible statistical modeling. *The American Statistician* 111–131 (AMS, 2021).

60. Kochkov, D. et al. Machine learning–accelerated computational fluid dynamics. *Proc. Natl Acad. Sci.* **118**, e2101784118 (2021).

61. Belbute, P., Filipe, D. A., Economon, T. & Kolter, Z. Combining differentiable pde solvers and graph neural networks for fluid flow prediction. In *International Conference on Machine Learning* 2402–2411 (PMLR, 2020).

62. Um, K., Brand, R., Fei, YunRaymond, Holl, P. & Thuerey, N. Solver-in-the-loop: Learning from differentiable physics to interact with iterative pde-solvers. *Adv. Neur. Inform. Process. Syst.* **33**, 6111–6122 (2020).

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

64. San, O. & Maulik, R. Neural network closures for nonlinear model order reduction. *Adv. Comput. Math.* **44**, 1717–1750 (2018).

65. Beck, A., Flad, D. & Munz, Claus-Dieter Deep neural networks for data-driven les closure models. *J. Comput. Phys.* **398**, 108910 (2019).

66. Ronneberger, O., Fischer, P. & Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI on 18th International Conference* 234–241 (Springer, 2015).

67. Dosovitskiy, A. et al. An image is worth 16x16 words: transformers for image recognition at scale. *arXiv* https://doi.org/10.48550/arXiv.2010.11929 (2020).

68. Lu, L. et al. A comprehensive and fair comparison of two neural operators (with practical extensions) based on FAIR data. *Comp. Methods Appl. Mech. Eng.* **393**, 114778 (2022).

69. Li, Z. et al. Physics-informed neural operator for learning partial differential equations. *arXiv* https://doi.org/10.48550/arXiv.2111.03794 (2021).

70. Chen, Z., Badrinarayanan, V., Lee, C. Y. & Rabinovich, A. Gradnorm: gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning* 794–803. (PMLR, 2018).

71. McClenny, L. & Braga, N. U. Self-adaptive physics-informed neural networks using a soft attention mechanism. *arXiv* https://doi.org/10.48550/arXiv.2009.04544 (2020).

72. Kucukelbir, A., Tran, D., Ranganath, R., Gelman, A. & Blei, D. M. Automatic differentiation variational inference. *J. Mach. Learn. Res.* https://doi.org/10.10932Fgbe2Fevad099 (2017).

73. Graves, A. Practical variational inference for neural networks. *Advances in Neural Information Processing Systems* 24 (NeuralPS, 2011).

74. Hoffman, M. D., Blei, D. M., Wang, C. & Paisley, J. Stochastic variational inference. *J. Mach. Learn. Res.* https://jmlr.org/papers/v14/hoffman13a.html (2013).

75. Gal, Y. & Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning* 1050–1059 (PMLR, 2016).

76. Lakshminarayanan, B., Pritzel, A. & Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in Neural Information Processing Systems* 30 (NeuralPS, 2017).

77. Ovadia, Y. et al. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *Advances in Neural Information Processing Systems* 32 (NeuralPS, 2019).

78. Rahaman, R. et al. Uncertainty quantification and deep ensembles. *Adv. Neur. Inform. Process. Syst.* **34**, 20063–20075 (2021).

79. Dong, B., Jiang, Q. & Shen, Z. Image restoration: Wavelet frame shrinkage, nonlinear evolution pdes, and beyond. *Multisc. Model. Simult.* **15**, 606–660 (2017).

80. Long, Z., Lu, Y., Ma, X. & Dong, B. Pde-net: Learning pdes from data. In *International Conference on Machine Learning* 3208–3216 (PMLR, 2018).

81. Long, Z., Lu, Y. & Dong, B. Pde-net 2.0: Learning pdes from data with a numeric-symbolic hybrid deep network. *J. Comput. Phys.* **399**, 108925 (2019).

82. Gonzalez, S. A. et al. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning* 8459–8468 (PMLR, 2020).

Author contributions

X.Y.L., H.S. and J.X.W. contributed to the ideation and design of the research; X.Y.L. and J.X.W. performed the research (implemented the model, conducted numerical experiments, analyzed the data, contributed materials/analysis tools); X.Y.L, M.Z. and L.L. contributed to the comparison study with other models; X.Y.L. and J.X.W. wrote the manuscript; H.S. and L.L. contributed to manuscript editing.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s42005-024-01521-z.

Correspondence and requests for materials should be addressed to Jian-Xun Wang.

Peer review information *Communications Physics* thanks the anonymous reviewers for their contribution to the peer review of this work. A peer review file is available.

Reprints and permission information is available at http://www.nature.com/reprints

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.