nature machine intelligence

Article

https://doi.org/10.1038/s42256-023-00685-7

Encoding physics to learn reaction–diffusion processes

Received: 6 September 2022

Accepted: 5 June 2023

Published online: 17 July 2023

Check for updates

Chengping Rao[1,2,7], Pu Ren[3,7], Qi Wang[1], Oral Buyukozturk[4], Hao Sun[1,5] & Yang Liu[6]

Modelling complex spatiotemporal dynamical systems, such as reaction–diffusion processes, which can be found in many fundamental dynamical effects in various disciplines, has largely relied on finding the underlying partial differential equations (PDEs). However, predicting the evolution of these systems remains a challenging task for many cases owing to insufficient prior knowledge and a lack of explicit PDE formulation for describing the nonlinear process of the system variables. With recent data-driven approaches, it is possible to learn from measurement data while adding prior physics knowledge. However, existing physics-informed machine learning paradigms impose physics laws through soft penalty constraints, and the solution quality largely depends on a trial-and-error proper setting of hyperparameters. Here we propose a deep learning framework that forcibly encodes a given physics structure in a recurrent convolutional neural network to facilitate learning of the spatiotemporal dynamics in sparse data regimes. We show with extensive numerical experiments how the proposed approach can be applied to a variety of problems regarding reaction–diffusion processes and other PDE systems, including forward and inverse analysis, data-driven modelling and discovery of PDEs. We find that our physics-encoding machine learning approach shows high accuracy, robustness, interpretability and generalizability.

Spatiotemporal dynamics is ubiquitous in nature. For example, reaction–diffusion processes exhibit interesting phenomena and are commonly seen in many disciplines such as chemistry, biology, geology, physics and ecology. The autonomous formation mechanism of stripe (Turing) patterns in the skin of tropical fishes can be revealed by diffusion and reaction. Like many other systems, understanding their complex spatiotemporal dynamics, governed by the inherent partial differential equations (PDEs), is a central task. Nevertheless, the principled laws in the context of closed-form governing equations for many underexplored systems remain uncertain or partially unknown (for example, the reaction mechanism is typically nonlinear and intractable to model). Even for some dynamical systems, for example, other than reaction–diffusion processes, whose PDEs are already known (such as Naiver–Stokes equations), the computational cost of accurate numerical simulation is also prohibitive for scientific applications involving large-scale spatiotemporal domains. Yet, machine learning has opened up new avenues for scientific modelling or discovery of the aforementioned systems in a data-driven manner.

In fact, the history of knowledge discovery from data (or observation) can be dated back to the time of Kepler, who discovered the well-known law of planetary motion from massive documented data. Recently, the revived machine learning methods have pushed the

[1]Gaoling School of Artificial Intelligence, Renmin University of China, Beijing, China. [2]Department of Mechanical and Industrial Engineering, Northeastern University, Boston, MA, USA. [3]Department of Civil and Environmental Engineering, Northeastern University, Boston, MA, USA. [4]Department of Civil and Environmental Engineering, MIT, Cambridge, MA, USA. [5]Beijing Key Laboratory of Big Data Management and Analysis Methods, Beijing, China. [6]School of Engineering Science, University of Chinese Academy of Sciences, Beijing, China. [7]These authors contributed equally: Chengping Rao, Pu Ren. e-mail: haosun@ruc.edu.cn; liuyang22@ucas.ac.cn

renaissance of the data-driven scientific computing, such as modelling of complex systems[1–9], super-resolution of scientific data[10–12], material property prediction[13], system identification and equation discovery[14–17], among others. These successful applications are largely attributed to the extraordinary expressiveness of deep learning models, which enables the automatic learning of the nonlinear mapping among variables from rich labelled data[18]. In particular, the latest research has shown that deep learning[19–22] could accelerate the discovery of underlying governing PDEs given sparse or noisy data. However, the pure data-driven methods rooted on deep learning typically learn representations from and highly rely on big data (for example, from experiment or simulation), which are often insufficient in most scientific problems. The resulting model often fails to satisfy physical constraints (for example, conservation laws, invariants), whose generalizability cannot be guaranteed either[23]. To tackle this issue, physics-informed neural networks (PINNs)[24–26] have taken a remarkable leap in scientific machine learning and become a major paradigm, which leverages our prior knowledge of underlying physics to enable learning in small data regimes.

PINNs have shown effectiveness in a wide range of scientific applications, including solving general PDEs[24,27,28], reduced-order modelling[29,30], uncertainty quantification[31,32], inverse problems[24,33], data-driven knowledge discovery[21] and others. In particular, the paradigm has been demonstrated to be effective in modelling a variety of physical systems, such as fluid dynamics[25,34], subsurface transport[35,36] and engineering mechanics[33,37–39]. However, the dominant physics-informed learning model, the PINN, generally represents a continuous learning paradigm as it employs fully connected neural networks (FCNNs) for the continuous approximation of the solution to the physical system. The resultant continuous representation of the system's prediction brings several limitations, including poor computational efficiency due to the nature of the FCNN, inaccurate physical constraints due to the soft penalty in the loss function and a lack of capability to hard-encode prior physics into the learning model. Fortunately, the latest studies in discrete learning models, such as convolutional neural networks[32,40–42], graph neural networks[43] and transformers[44], show promise in overcoming some of the above limitations. Compared with the continuous learning model, the discrete learning approaches have a distinct advantage of hard encoding the initial conditions (ICs) and boundary conditions (BCs), as well as the incomplete PDE structure, into the learning model. This practice could avoid the ill-posedness of the optimization even without any labelled data as shown in very recent studies[41,45,46]. Therefore, we are motivated to establish an effective, interpretable and generalizable discrete learning paradigm that can be leveraged for predicting the nonlinear physical systems, which remains a substantial challenge in scientific machine learning. Recent advances have shown that operator learning can naturally achieve this goal, for example, DeepONet[47] and Fourier Neural Operator[48]. However, a rich set of labelled data should be supplied to train reliable operators for these methods. Although adding prior physics to constrain DeepONet helps alleviate the need of large data[49], the explicit expression of PDE(s) must be given, which falls short in dealing with systems whose governing equations are partially or completely unknown.

To this end, we propose the physics-encoded model that encodes the prior physics knowledge in the network architecture, in contrast to 'teaching' models the physics through a penalized loss function commonly seen in physics-informed learning. In particular, our model has four major characteristics. (1) Compared with the dominant method of PINN that utilizes an FCNN as a continuous approximator to the solution, the physics-encoded model is discrete (that is, the solution is spatially mesh based and defined on discrete time steps) and hard encodes the given physics structure into the network architecture. (2) Our model employs a unique convolutional network (that is, a Π-block, discussed in Methods) to capture the spatial patterns of the system while the time marching is performed by the recurrent unit. This unique network has been demonstrated (with mathematical proof and numerical experiments) to promote the expressiveness of our model on nonlinear spatiotemporal dynamics. (3) Owing to the discretization with time, our network is able to incorporate well-known numerical time integration methods (for example, forward Euler scheme, Runge–Kutta scheme) for encoding incomplete PDEs into the network architecture. Throughout this article, we demonstrate the capabilities of the proposed network architecture by applying it to various tasks in scientific modelling of spatiotemporal dynamics such as reaction–diffusion processes.

Results

Physics-encoded spatiotemporal learning

The motivation of the physics-encoded spatiotemporal learning paradigm is to establish a generalizable and robust model for predicting the physical system state based on very limited low-resolution and noisy measurement data. The established model is expected to deliver good extrapolation capability over the temporal horizon and generalization to different ICs. These demands essentially require the proposed model to learn the underlying spatiotemporal dynamics from data. To this end, we propose a novel network, namely, the physics-encoded recurrent convolutional neural network (PeRCNN), as shown in Fig. 1. The network is designed to preserve the given physics structure, for example, structure or specific terms of the governing PDEs, ICs and BCs. The prior physics knowledge is forcibly 'encoded', which makes the network possess interpretability. More details are given in Methods.

Reaction–diffusion systems

Reaction–diffusion (RD) equations have found wide applications in the analysis of pattern formation[50], such as population dynamics[51], chemical reactions[52], cell proliferations[53] and so on. In this article, we specifically consider three different RD systems of the lambda–omega (λ–Ω), FitzHugh–Nagumo (FN) and Gray–Scott (GS) types to verify the proposed approach. In general, the RD system can be described by the following governing equation

$$\mathbf{u}_t = \mathbf{D}\Delta\mathbf{u} + \mathbf{R}(\mathbf{u}) \tag{1}$$

where $\mathbf{u} \in \mathbb{R}^n$ is the vector of concentration variables, the subscript t denotes time derivative, n represents the system dimension, $\mathbf{D} \in \mathbb{R}^{n\times n}$ is the diagonal diffusion coefficient matrix, Δ is the Laplacian operator and $\mathbf{R}(\mathbf{u})$ is the reaction vector that represents the interactions among components of $\mathbf{u}$. Without loss of generality, let us assume the RD system features two components, that is, $\mathbf{u} = [u, v]^{\mathrm{T}}$. Specifically, the λ–Ω RD system is governed by

$$\begin{aligned} u_t &= \mu_u\Delta u + (1-u^2-v^2)\,u + \beta\,(u^2+v^2)\,v \\ v_t &= \mu_v\Delta v - \beta\,(u^2+v^2)\,u + (1-u^2-v^2)\,v \end{aligned} \tag{2}$$

while the FN RD system can be described by

$$\begin{aligned} u_t &= \mu_u\Delta u + u - u^3 - v + \alpha \\ v_t &= \mu_v\Delta v + (u-v)\beta \end{aligned} \tag{3}$$

where α and β are the coefficients prescribing the reaction process and take different values. Similarly, the GS RD system can be described by

$$\begin{aligned} u_t &= \mu_u\Delta u - uv^2 + F(1-u) \\ v_t &= \mu_v\Delta v + uv^2 - (F+\kappa)v \end{aligned} \tag{4}$$

where κ and F denote the kill and feed rate, respectively. For the FN and GS RD systems, we consider both two-dimensional (2D) and

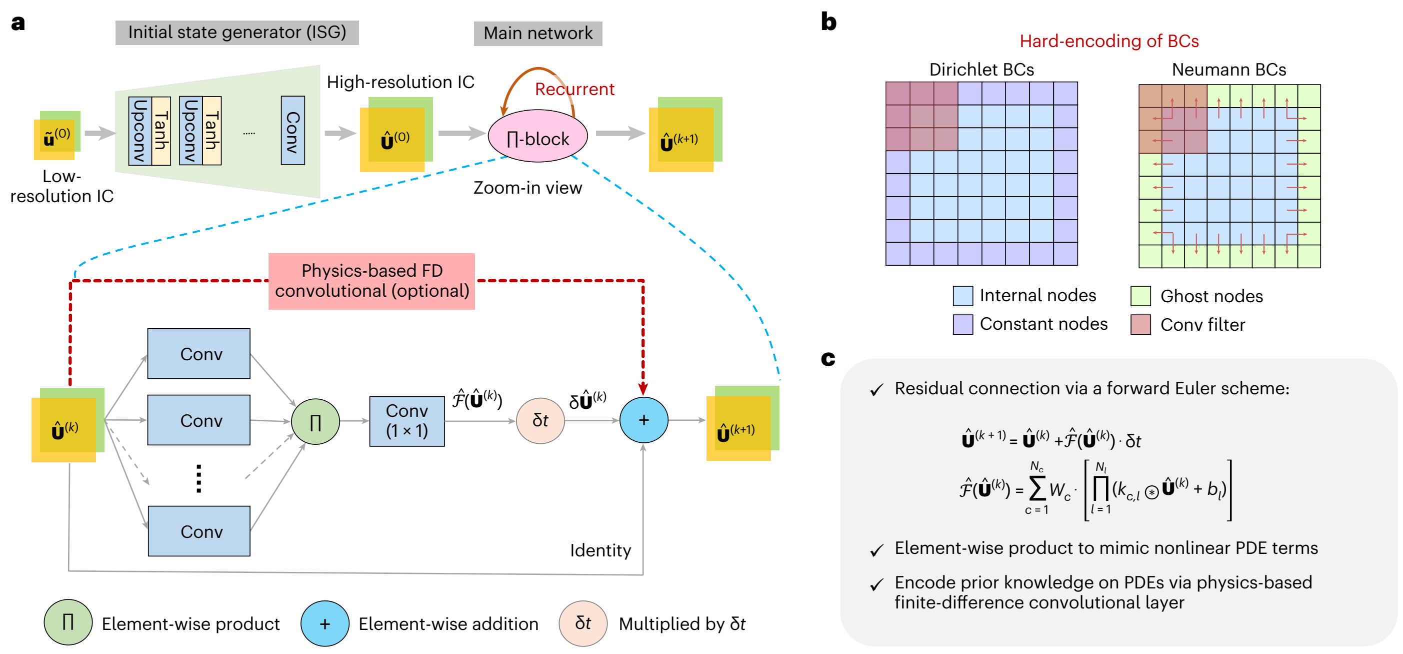

a
Initial state generator (ISG)
Main network
High-resolution IC
Recurrent
Upconv
Tanh
Upconv
Tanh
Conv
$\tilde{\mathbf{u}}^{(0)}$
Low-resolution IC
$\hat{\mathbf{U}}^{(0)}$
Π-block
$\hat{\mathbf{U}}^{(k+1)}$
Zoom-in view
Physics-based FD convolutional (optional)
Conv
Conv
Conv
Π
Conv (1 × 1)
$\hat{\mathcal{F}}(\hat{\mathbf{U}}^{(k)})$
δt
$\delta\hat{\mathbf{U}}^{(k)}$
+
$\hat{\mathbf{U}}^{(k)}$
$\hat{\mathbf{U}}^{(k+1)}$
Identity
Element-wise product
Element-wise addition
Multiplied by δt
b
Hard-encoding of BCs
Dirichlet BCs
Neumann BCs
Internal nodes
Constant nodes
Ghost nodes
Conv filter
c
✓ Residual connection via a forward Euler scheme:
$\hat{\mathbf{U}}^{(k+1)} = \hat{\mathbf{U}}^{(k)} + \hat{\mathcal{F}}(\hat{\mathbf{U}}^{(k)}) \cdot \delta t$
$\hat{\mathcal{F}}(\hat{\mathbf{U}}^{(k)}) = \sum_{c=1}^{N_c} W_c \cdot \left[\prod_{l=1}^{N_l} (k_{c,l} \circledast \hat{\mathbf{U}}^{(k)} + b_l)\right]$
✓ Element-wise product to mimic nonlinear PDE terms
✓ Encode prior knowledge on PDEs via physics-based finite-difference convolutional layer

Fig. 1 | Schematic architecture of the PeRCNN. a, The network with a Π-block for the recurrent computation. **b**, Embedding of BCs. **c**, Key features of the network. Here, $\tilde{\mathbf{u}}^{(0)}$ denotes the low-resolution and noisy measurement of the initial state, while $\hat{\mathbf{U}}^{(k)}$ denotes the predicted fine-resolution solution at time t_k. The decoder (ISG) is used to downscale/upsample the low-resolution initial state. Conv denotes 'convolution'.

three-dimensional (3D) cases in this article. To generate the numerical solution as the ground-truth reference, we discretize the regular physical domain with Cartesian grid and utilize a high-order finite difference (FD) method to simulate the evolution of the RD system. The computational and discretization parameters for each case are provided in Extended Data Table 1.

Forward analysis of PDE systems

Solving general PDEs is undoubtedly the cornerstone of scientific computing. We herein demonstrate the capability of the PeRCNN for forward analysis of PDE systems (that is, solving PDEs), in particular, the aforementioned RD systems. It is assumed that the governing PDEs, together with the necessary ICs and BCs, of the physical system are completely known. Therefore, the prescribed initial state is fed to the network for the recurrent computation without resorting to the initial state generator (ISG). Once the forward recurrent computation is finished, the snapshot (or prediction) at each time step is collected. The FD is then applied on the discrete snapshots for computing the partial derivatives involved in the governing PDE. The mean squared error of the equation residual is used as the optimization objective (or loss function) for obtaining a set of model's parameters. Multiple RD systems, including the 2D λ–Ω, 2D/3D FN and 2D GS RD equations, are considered herein as the numerical examples. The ground-truth reference solution is generated by the high-order FD method. The governing PDEs, computational domain and discretization settings for each system are provided in Extended Data Table 1. Detailed discussions on the network settings are given in Supplementary Note C. Details on how to properly select the spatial and temporal grid sizes, to ensure the model's numerical stability and achieve desired solution resolution, are given in Supplementary Note H.

Figure 2 shows the snapshots predicted by the PeRCNN for each system at a given time. To compare the performance of the proposed approach with existing methods, we also provide the result of two baseline models, namely, the convolutional long-short term memory (ConvLSTM)[54] and the PINN[24]. It can be seen that the solution obtained by the PeRCNN agrees well with the reference for all four cases. In contrast, the ConvLSTM and the PINN perform differently on 2D and 3D cases; in particular, they get a fairly good prediction for 2D cases while considerably deviate from the reference for 3D cases. To examine the accuracy of each method as a PDE solver, we compute the accumulative root mean square error (RMSE) of the prediction. It shows that the PeRCNN achieves a significantly lower error throughout the considered time-marching interval. Although existing techniques (for example, finite difference, volume and element methods) for solving PDEs are already mature nowadays, the result in this part demonstrates the promise of the PeRCNN on modelling and simulation of complex systems.

Inverse analysis of PDE systems

Calibrating the unknown parameters of a given model against experimental data is a commonly seen problem in scientific discovery, for example, one might be interested in uncovering the scalar coefficients in the governing PDEs given very limited observed snapshots of the system. As the proposed PeRCNN has the capability of encoding the PDE structure (for example, known terms) into the network architecture for predicting spatiotemporal systems, we can apply it to identify the unknown coefficients by treating them as trainable variables. To verify the effectiveness of the PeRCNN in inverse analysis of PDEs, we consider the 2D GS RD system governed by the following two coupled equations: $u_t = \mu_u \Delta u - c_1 uv^2 + c_F(1-u)$ and $v_t = \mu_v \Delta v + c_2 uv^2 - (c_F + c_\kappa)v$, where μ_u, μ_v, c_1, c_2, c_F and c_κ are unknown coefficients. As the explicit form of the governing PDEs is known, we construct the physics-based FD convolutional connections (that is, diffusion and other polynomial terms) according to the right-hand side of the governing equation. Note that no element-wise product layer is involved in the network. Meanwhile, each unknown coefficient is treated as an individual trainable variable in the computational graph for the forwards and backwards computations.

To examine the capability of the model in scenarios of various data availability, we consider two different sets of measurement data. In the first scenario (S1), the available measurement includes multiple noisy and low-resolution snapshots of the system, which means the available data are scarce spatially while somewhat abundant in the temporal dimension. In the second scenario (S2), we assume only the first and last snapshots of the system with decent resolution are

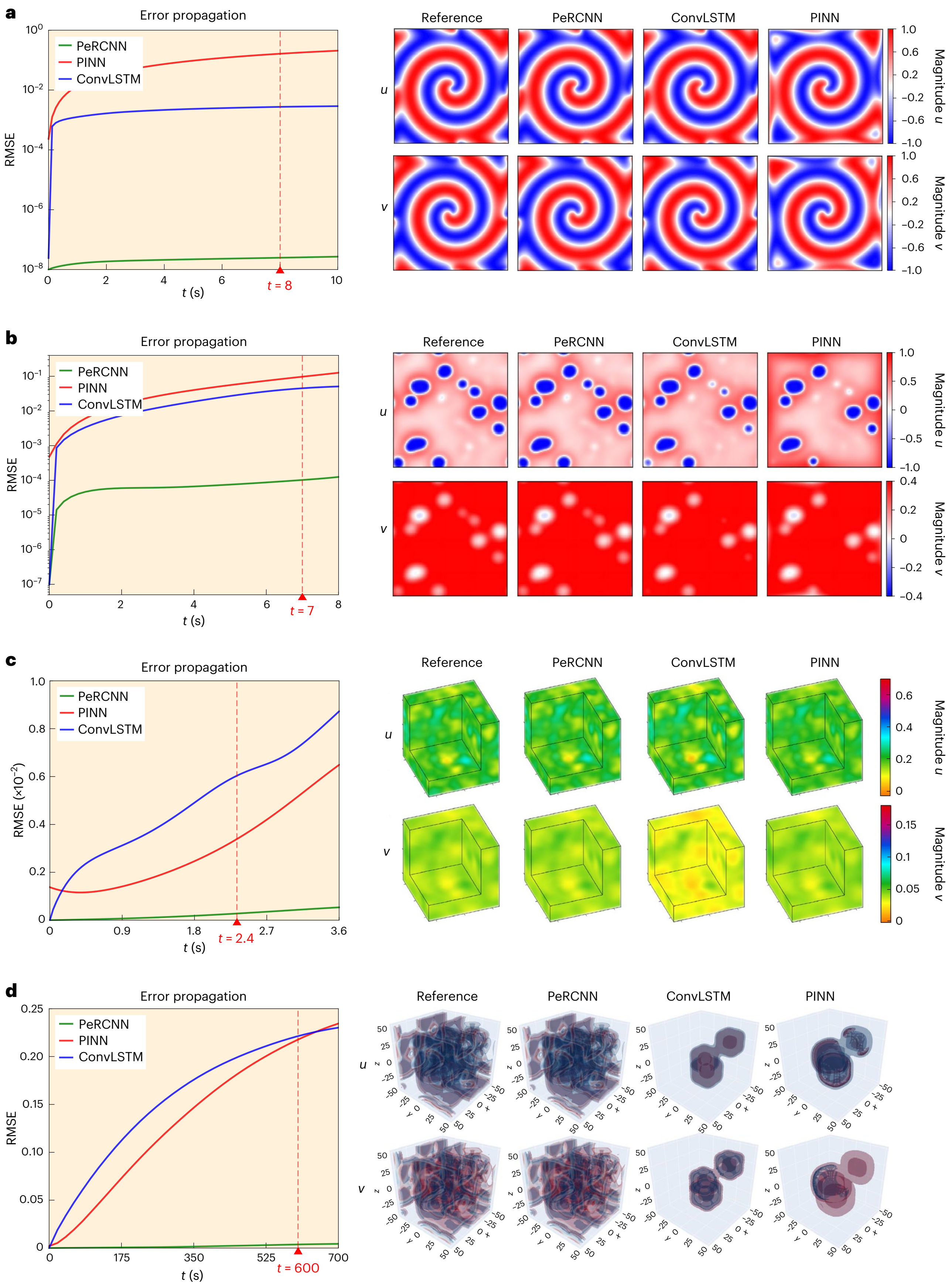


Fig. 2 | Error propagation curve and predicted snapshots by PeRCNN, ConvLSTM and PINN on various RD systems. **a**, 2D λ–Ω RD equation. **b**, 2D FN RD equation. **c**, 3D FN RD equation. **d**, 3D GS RD equation. Note that only a corner of the solution field is shown for the FN RD systems in **c** for better visualization of the internal solution distribution.

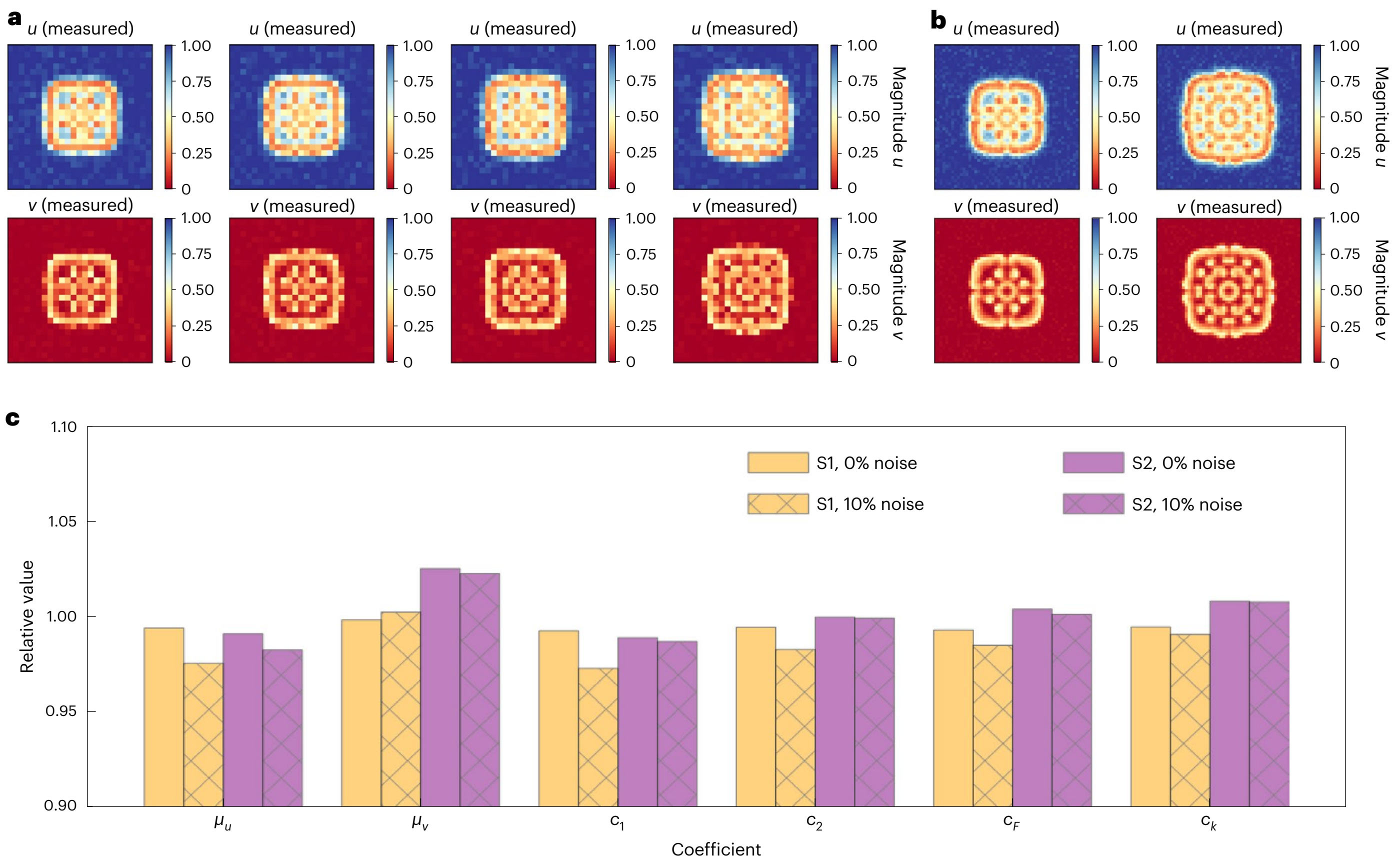


Fig. 3 | Snapshots of the measurement data employed in the experiment and the identified coefficients. a, Data availability in S1 (multiple snapshots with 26 × 26 resolution). **b**, Data availability in S2 (initial and last snapshots with 51 × 51 resolution). **c**, Identified coefficient at various noise levels and data availability.

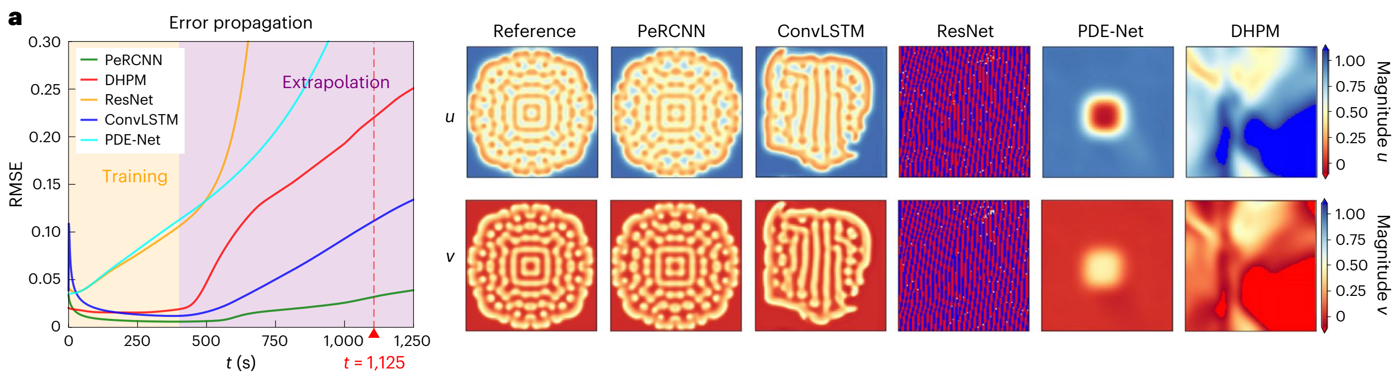
available. These two scenarios reflect the trade-off between the spatial and temporal resolution of the existing measurement. These synthetic measurements accompanied with 10% Gaussian noise are shown in Fig. 3a,b. The misfit error between the prediction and the measurement data is computed as the loss function for optimizing the unknowns. To prevent the overfitting to noise, early stopping is employed by splitting the dataset into training and validation sets. The details of computational parameters for dataset generation, network architecture, initialization of coefficients and optimization settings are presented in Supplementary Note D.

To test the effects of noise, the experiment is also performed on clean data. Each case encompasses ten runs with various random seed for coefficient initialization. The identified coefficients are presented in Fig. 3c and Extended Data Table 2. It can be seen that, in all cases, the PeRCNN is able to uncover the unknown coefficients with satisfactory accuracy. Compared with the identified coefficients in the noise-free case, the result deteriorates only slightly with 10% noise. In the absence of the noise, the identified coefficients feature high accuracy, with the mean absolute relative error for all the coefficients being 0.6%. For the case with 10% noise, the mean absolute relative error is 1.61% in spite of 10% Gaussian noise in the measurement. The PeRCNN also shows superiority to the PINN (see the result reported in Supplementary Table 4). Moreover, the potential of employing the PeRCNN to identify space-varying coefficients is further demonstrated in Supplementary Note D.5). The numerical results in this section illustrates the good capability of our model on the inverse analysis of PDE systems.

Data-driven modelling of spatiotemporal dynamics

PDEs play an an essential role in modelling physical systems. However, there still exist a considerable portion of systems, such as those in epidemiology, climate science and biology, whose underlying governing PDEs are either completely unknown or only partially known. Owing to the ever-growing data availability as well as recent advances in scientific machine learning, data-driven modelling nowadays becomes an effective way to establish predictive models for physical systems. As the proposed network is characterized with excellent expressiveness for representing nonlinear dynamics (see 'Universal polynomial approximation property for the Π-block' in Methods) and the capability of encoding an incomplete governing PDE, it has great potential to serve as a generalizable and robust data-driven model for predicting high-resolution nonlinear spatiotemporal dynamics. In this part, we primarily focus on data-driven modelling of spatiotemporal dynamics by the proposed physics-encoded learning paradigm given limited, noisy measurement data.

Let us assume some low-resolution and potentially noisy snapshots of the system are measured, that is, $\tilde{\mathbf{u}} \in \mathbb{R}^{n'_t \times n \times H' \times W'}$ where n'_t is the number of snapshots, n is the number of state variable components and $H' \times W'$ is the resolution of each snapshot. We seek to establish a predictive model that gives the most likely high-resolution solution $\hat{\mathbf{U}} \in \mathbb{R}^{n_t \times n \times H \times W}$ where $n'_t < n_t$, $H' < H$ and $W' < W$, and possesses satisfactory extrapolation ability over the temporal horizon (for example, for $t > t_{n_t}$). As we have seen, one salient characteristic of the PeRCNN is the capability of encoding prior knowledge (for example, the general PDE structure and/or the ICs/BCs) into the learning model. In particular, we assume the basic PDE form as shown in equation (4) is given, where the diffusion term is known a priori whose coefficients are however unknown. To compare the PeRCNN with existing methods, we also perform experiments on several baseline models, namely, the recurrent ResNet[55,56], ConvLSTM[54], PDE-Net[5] and the deep hidden physics model[57]. Once the training is done, we infer the high-resolution prediction from



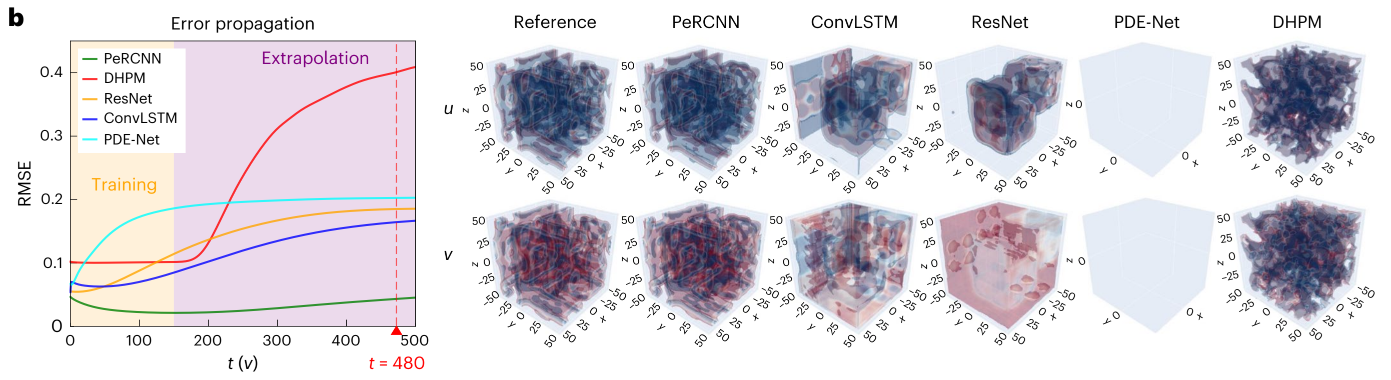


Fig. 4 | Error propagation curve of the prediction and the extrapolated snapshots from each data-driven model compared with the reference solution. Note that extrapolation is performed beyond the time horizon of the measurement data while the time of each snapshot is marked by a dashed red line in the error propagation curve. **a**, 2D GS RD equation. **b**, 3D GS RD equation.

the predictive data-driven models. The accumulative RMSE and the physics residual (see Supplementary Note E.3 for definition) are utilized to evaluate the accuracy of the established data-driven models.

We verify the performance of the PeRCNN using synthetic datasets of the 2D and 3D GS RD equation systems, whose computational parameters and discretization setting are provided in Extended Data Table 1. In the experiments, we fix the amount of data, training, validation and testing dataset splitting, the number of prediction steps, the Gaussian noise level (10%), and the random seed for each method. The hyperparameters for each case are selected through hold-out cross-validation. The synthetic measurement data (that is, some low-resolution snapshots) are downsampled (in both spatial and temporal dimensions) from the numerical solution. Once the model is finalized, extrapolation beyond temporal horizon, for example, for $t > t_{n_t'}$, would be performed to examine the extrapolation ability of each model. Note that comprehensive sensitivity tests of the PeRCNN in the context of some major hyperparameters (that is, filter size, number of convolutional layers and number of channels) are presented in Supplementary Note E.4.

2D GS RD equation. In this case, we consider a data availability scenario where the resolution of measurement data is relatively low in space but decent in time. The available measurement data in this case encompasses 41 noisy snapshots of on a 26×26 grid, ranging from $t = 0$ to $t = 400$ s. As we assume the dynamical system of interest features the ubiquitous diffusion phenomenon, the governing PDE (that is, $\mathcal{F}$) is known to have a diffusion term ($\Delta \mathbf{u}$) whose scalar coefficients are still unknown. Therefore, we encode the diffusion term into the PeRCNN by creating a physics-based FD convolutional connection with the discrete Laplace operator as the convolutional kernel (Supplementary Note B.3). Furthermore, the diffusion coefficient ($\tilde{\mu}$) is first estimated by solving a linear regression problem of $u_t = \tilde{\mu}\Delta u$ with the available data. Then a lower bound of 0 and upper-bound of $2\tilde{\mu}$ are applied to ensure the stability of diffusion. Each model is responsible for predicting 801 fine-resolution snapshots during the training phase, while 1,700 extra snapshots are predicted for extrapolation.

Snapshots at different time instants are presented in Fig. 4a, which reveal the complex maze-like pattern of the GS RD system. We report that the recurrent ResNet and PDE-Net are unable to reconstruct the fine-resolution snapshots even in the training due to the limited noisy training data, after trying all the hyperparameter combinations within a range (Supplementary Note E.5.1). Apart from that, it can be seen from the snapshots that the PeRCNN is the only model working well for long-time extrapolation in spite of minor discrepancies. It is also interesting to note that PeRCNN with filter size of 1 works as good as the model with larger filters (for example, 3 or 5). This is because the reaction term of the GS RD system contains no spatial derivatives, making the 1×1 filter sufficient for representing the nonlinear reaction terms. This implies that prior knowledge on the governing PDE can also be employed while designing the data-driven model. To quantitatively measure the extrapolation capability of our model, we also plot the evolution of accumulative RMSE in Fig. 4a. It is observed that the PeRCNN outperforms the competitors at all stages in the context of error propagation, which further confirms the extrapolation ability of the PeRCNN. We may notice that the accumulative RMSE starts from an initial high value. This is due to the fact that the training data are corrupted by 10% Gaussian noise and the metric is computed from one single snapshot at the beginning. The effect of the unrelated noise gradually fades out as more time steps are considered.

3D GS RD equation. In this example, we test our method on the 3D GS RD system. As the computational intensity of this higher-dimensional example brings challenges to the existing methods, we aim to scrutinize the performance of our PeRCNN regarding the scalability and computational efficiency. The training data used to establish the data-driven

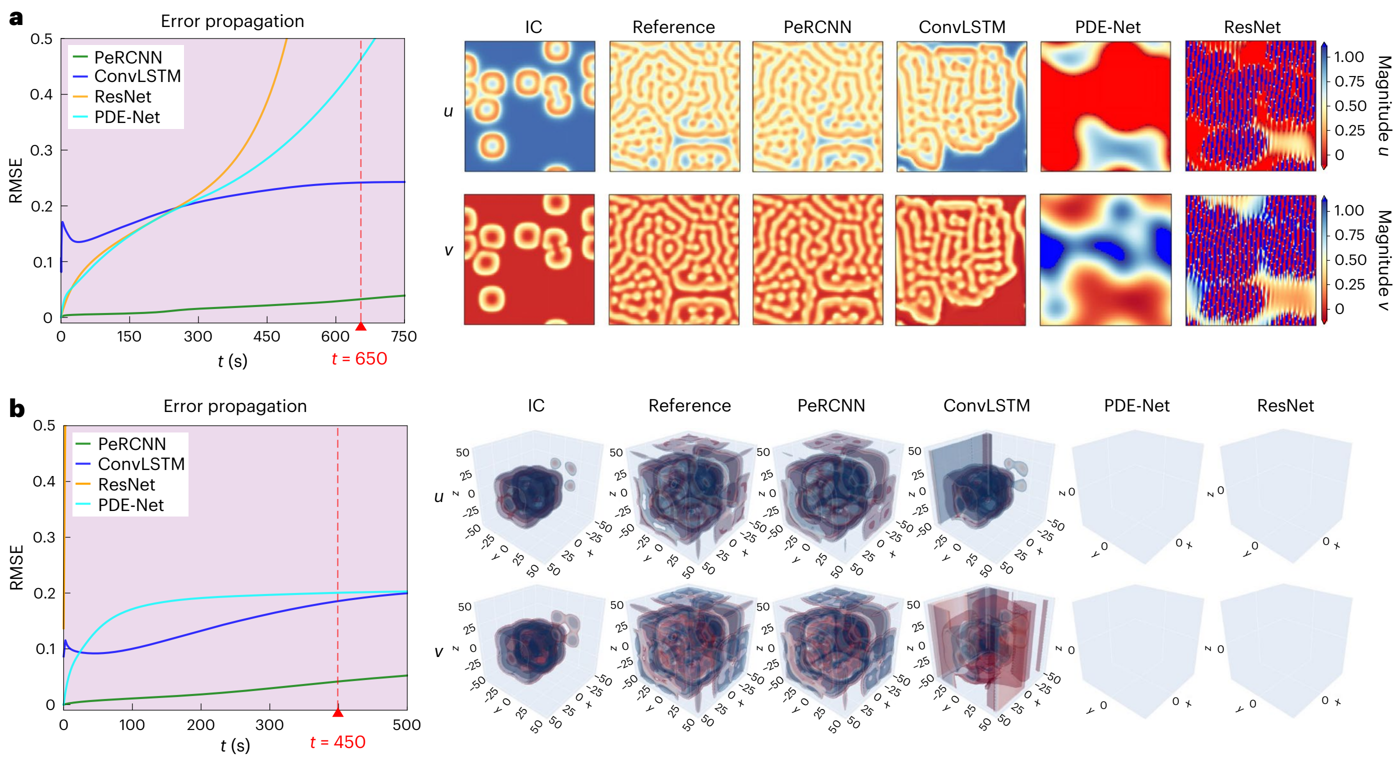


Fig. 5 | Error propagation curve and the snapshots of the inference result. The results are obtained by performing inference on the data-driven model from the section 'Data-driven modelling of spatiotemporal dynamics' using a different IC. **a**, 2D GS RD equation. **b**, 3D GS RD equation.

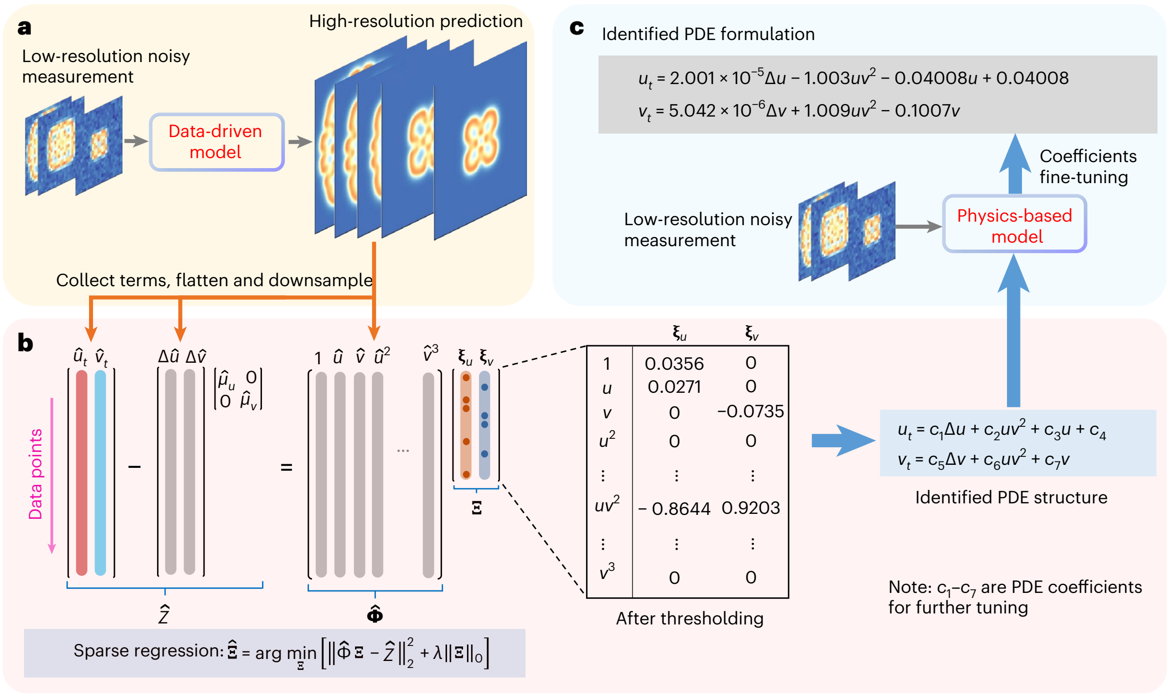
model include 21 noisy low-resolution snapshots (25^3) uniformly sampled from $t = 0$ to $t = 150$ s. The prior knowledge on the system and the estimation of the diffusion coefficients as discussed in the previous 2D GS RD example are adopted here as well. Each trained model produces 301 high-resolution (49^3) snapshots during the learning stage, while 700 extrapolation steps are predicted once each model is finalized. The predicted isosurfaces of two levels are plotted in Fig. 4b. It should be noted that the plot of PDE-Net is blank because the prediction range falls out of the two selected isosurface levels. Similar to the previous case, we observe that PeRCNN is the only model that gives a satisfactory long-term prediction. The flat error propagation curve of the PeRCNN, as shown in Fig. 4b, also demonstrates the remarkable generalization capability of PeRCNN.

In Supplementary Note E.5, we compare the number of trainable parameters, the training time per epoch, and the RMSE of both training and extrapolation for each model. It is found that the PeRCNN is characterized with good model efficiency as it uses the least amount of training parameters. For the 3D case where the training efficiency of the network is of great concern, the elapsed time for training one epoch by PeRCNN is comparable to that of the ResNet, which is widely acknowledged to be an efficient network architecture. As for the accuracy of the training and extrapolation, our model outperforms the baselines consistently across different examples. In a nutshell, the PeRCNN outperforms the other three baselines with much fewer trainable parameters and higher accuracy.

Generalization to different ICs. It is evident that the trained model has good extrapolation capability along the time horizon. Here we further explore how the trained model generalizes to different ICs. To set up the experiment, we employ the above trained model to perform inference with a different IC. It should be noted that the baseline deep hidden physics model is ineligible for inference with different ICs as it is based on an FCNN. The prediction result is depicted in Fig. 5. It is seen that the PeRCNN gives consistent prediction compared with the ground-truth reference solution. On the contrary, the considered baseline models (for example, recurrent ResNet, ConvLSTM and PDE-Net) are unable to generalize to an unseen IC. They give wild prediction because of their incapability of learning the underlying physics (for example, caused by the black-box property of the model). In addition, the error propagation of the prediction in Fig. 5 indicates clearly the excellent generalization capability of the proposed model. In the later section 'Interpretability of the learned model', we show that the extracted expression from the trained PeRCNN model is very close to the genuine $\mathcal{F}$, which to a large degree explains the remarkable generalization capability of our model given the fact that the trained PeRCNN model parameterizes the spatiotemporal dynamics well.

Data-driven discovery of PDEs with scarce and noisy data

In previous sections, we primarily investigate the scientific modelling tasks (for example, forward simulation or data-driven modelling) using the proposed model, which exhibits excellent accuracy and extrapolation ability as identified from the numerical results. However, the process of knowledge discovery does not end at modelling the physical phenomena of interests. More importantly, it is the translation of the learned patterns from the data (for example, formulated PDEs or empirical relationships) that lead scientists to understand the cause–effect relationship among physical variables, and further make inference on similar problems. Therefore, in this section, we extend the proposed physics-encoded learning model for discovering the closed-form governing PDEs[58]. To formulate this problem, let us again consider the nonlinear system described by equation (7). The objective of the equation discovery is to recover the closed form of the governing PDEs given the scarce and noisy measurement of the system. To this end, we integrate the sparse regression technique[15] with our PeRCNN model for solving this problem. The proposed framework for the PDE discovery is presented in Fig. 6 with the example of 2D GS RD equation.


a
Low-resolution noisy measurement
Data-driven model
High-resolution prediction
Collect terms, flatten and downsample
b
Data points
Sparse regression: $\hat{\Xi} = \arg\min_{\Xi}\left[\|\hat{\Phi}\Xi - \hat{Z}\|_2^2 + \lambda\|\Xi\|_0\right]$
After thresholding
Identified PDE structure
$u_t = c_1\Delta u + c_2uv^2 + c_3u + c_4$
$v_t = c_5\Delta v + c_6uv^2 + c_7v$
Note: c_1–c_7 are PDE coefficients for further tuning
c
Identified PDE formulation
$u_t = 2.001\times10^{-5}\Delta u - 1.003uv^2 - 0.04008u + 0.04008$
$v_t = 5.042\times10^{-6}\Delta v + 1.009uv^2 - 0.1007v$
Coefficients fine-tuning
Low-resolution noisy measurement
Physics-based model

	ξ_u	ξ_v
1	0.0356	0
u	0.0271	0
v	0	−0.0735
u^2	0	0
⋮	⋮	⋮
uv^2	− 0.8644	0.9203
⋮	⋮	⋮
v^3	0	0

Fig. 6 | Flowchart of the discovery of governing PDEs. a, Data reconstruction: data-driven model constructed from low-resolution and noisy measurement is used to generate high-resolution prediction for sparse regression. **b**, Sparse regression: the STRidge algorithm is used to obtain the sparse coefficient matrix $\boldsymbol{\Xi}$. **c**, Fine-tuning of coefficients: the PeRCNN built based on the identified PDE structure is employed to fine-tune the coefficient from the sparse regression.

The entire procedure consists of three steps, data reconstruction (Fig. 6a), sparse regression (Fig. 6b) and fine-tuning of coefficients (Fig. 6c), as discussed in 'Equation discovery' in Methods.

To validate the effectiveness of the our method, we perform the equation discovery on two RD systems (for example, 2D GS and λ–Ω RD systems) using synthetic datasets, which are obtained by downsampling the noise-corrupted numerical solution. Two different Gaussian noise levels (5% and 10%), as well as the noise-free case, are considered in the experiment. As we assume the ubiquitous diffusion phenomenon exists in the concerned system, a short-cut diffusion convolutional layer is encoded into the network for data reconstruction (Fig. 1a). Accordingly, the coefficients corresponding to $\Delta(\mathbf{u})$ are exempted from being filtered in the Sequential Threshold Ridge regression (STRidge) algorithm. Once the equation discovery is finished, we measure the performance of the proposed method using the metrics of precision, recall and relative ℓ_2 error of the coefficient vector. Technical details of the generation of synthetic measurement data and the evaluation metrics can be found in Supplementary Note F.3.

The discovered PDEs by our method are provided in Extended Data Table 3. It is seen that our method is able to recover the governing PDEs completely when the measurement data are clean or mildly polluted by noise. Even though the noise level grows to 10%, our approach still exhibits competitive performance, that is, it uncovers the majority of terms in the PDEs. Empirical study in Supplementary Note F.5 shows that our method could handle even a much larger noise level, that is, 30% Gaussian noise. In Supplementary Note F.4, we also compare our approach with some existing methods (or baselines) for governing PDE discovery, including PDE-FIND[16], sparse regression coupled with an FCNN or PDE-Net[5]. The comparison shows that our approach outperforms (if not performs as good as) the considered baselines consistently under different noise levels and data richness. Visualizing the reconstructed high-fidelity data from each method (Supplementary Fig. 17), we observe that our method has a much smaller reconstruction error as a result of fully utilizing the prior physics knowledge and the powerful expressiveness of the model. This would give rise to more accurate derivative terms in the linear system, facilitating the discovery of the governing PDEs. In summary, the effectiveness of the proposed approach is demonstrated for solving the data-driven equation discovery problem, especially when the measurement data are characterized by poor resolution and noise.

Interpretability of the learned model

Compared with the traditional deep neural networks, which are usually considered to be 'black box', the proposed network architecture is designed to possess good interpretability. As each channel of the input for the Π-block (that is, $\hat{\mathbf{U}}^{(k)}$ where k denotes the discrete time instant) corresponds to a solution component (that is, u and v), the multiplicative form of the Π-block (equation (8)) makes it possible to extract an explicit form of learned $\mathcal{F}$ from the learned weights and biases via symbolic computations. This section is dedicated to the discussion on how the learned model can be interpreted as an analytical expression, which is useful for people to understand the underlying cause–effect relationships among physical variables.

To demonstrate how to interpret the learned model, we first use the learned model from 3D GS RD case in 'Data-driven modelling of spatiotemporal dynamics' as an example. In this case, the parallel convolutional layers in the Π-block have the filter size of 1, which implies that each output channel is the linear combination of u, v and a constant. With the element-wise product operation among three convolutional layers, a third-degree polynomial will be produced to account for the reaction term of the system. With the help of the SymPy[59]—a symbolic computation Python package—we can extract the learned reaction term:

$$\mathbf{R}(\mathbf{u}) = \begin{bmatrix} -0.0074u^3 - 0.0051u^2v - 0.2uv^2 - 0.0386v^3 - 0.0018u^2 \\ -0.11uv - 0.055v^2 - 0.016u - 0.022v + 0.025; \\ 0.0005u^3 - 0.013u^2v + 0.54uv^2 - 0.087v^3 - 0.0076u^2 \\ +0.023uv + 0.046v^2 + 0.017u - 0.036v - 0.0097 \end{bmatrix} \tag{5}$$

Meanwhile, the identified diffusion term can be also extracted from the trainable variables in diffusion connections, which reads $\mathbf{D}(\mathbf{u}) = \left[0.18\Delta u, 0.08\Delta v\right]^{\mathrm{T}}$. Comparing the extracted term with the ground-truth PDEs, we observe some distracting terms due to the 10% noise in the training data and the redundancy of the network. Further pruning on the raw expression can be done to make it more parsimonious.

The above example is a special case where the convolutional layer in the Π-block has filter size of 1, which indicates no spatial derivatives are involved in reaction terms. However, we can extend the network architecture design to make it applicable to general cases. To interpret terms involving partial derivatives (for example, $u\Delta u$, uu_x), we could completely freeze or impose moment matrix constraints on part of the convolutional filters[5]. Here, an experiment is conducted on the 2D Burgers' equation, which has wide applications in applied mathematics such as fluid and traffic flow modelling, given by $\mathbf{u}_t + \mathbf{u}\cdot\nabla\mathbf{u} = \nu\Delta\mathbf{u}$, where $\mathbf{u} = [u, v]^{\mathrm{T}}$ denotes the fluid velocities and ν is the viscosity coefficient. The network employed in the experiment has two convolutional layers with two channels. The first convolutional layer is associated with derivative operators $\partial(\cdot)/\partial x$ and $\partial(\cdot)/\partial y$, respectively, by fixing the filters with corresponding FD stencils. The synthetic dataset is generated on 101×101 Cartesian grid using high-order FD method with $\nu = 0.005$. The noise-free synthetic measurement data used for constructing the model include 11 low-resolution (51×51) snapshots uniformly selected from the time period of $t\in[0, 0.1]$.

After the model is trained, we interpret the expression from the PeRCNN model, which reads:

$$\mathbf{u}_t = \begin{bmatrix} 0.0051\Delta u - 0.95u_x(1.07u - 0.0065v - 0.17) \\ +0.98u_y(0.0045u - 1.01v + 0.17) + 0.053; \\ 0.0051\Delta v - 0.82v_x(1.22u + 0.0078v - 0.18) \\ -0.91v_y(0.0063u + 1.08v - 0.17) + 0.058 \end{bmatrix} \tag{6}$$

It can be observed that the equivalent expression of the learned model matches well the genuine governing PDEs, except for some minor terms whose coefficients are close to zero. In addition, the extracted expression helps explain the extraordinary extrapolation and generalization capabilities of our model. Although the selection of differential operators to be embedded is crucial for identifying the genuine form of the $\mathcal{F}$, the above two examples demonstrate the interpretability of PeRCNN over common black-box models.

Discussion

This paper introduces a novel deep learning architecture, namely, PeRCNN, for modelling and discovery of nonlinear spatiotemporal dynamical systems based on sparse and noisy data. One major advantage of the PeRCNN is that the prior physics knowledge can be encoded into the network, which guarantees that the resulting network strictly obeys given physics (for example, ICs and BCs, general PDE structure, and known terms in PDEs). This brings distinct benefits for improving the convergence of training and accuracy of the model. Through extensive numerical experiments, we show the efficacy of the PeRCNN for forward and inverse analysis of RD-type PDEs. The comparison with several baseline models demonstrates that the proposed physics-encoded learning paradigm uniquely possesses remarkable extrapolation ability, generalizability and robustness against data noise and/or scarcity. Although we demonstrate the effectiveness of the PeRCNN on various RD systems, the model is in theory applicable to other types of spatiotemporal PDE (for example, the 2D Burgers' equation with the convection term shown in Supplementary Note F.4, and the Kolmogorov turbulent flows at Reynolds number 1,000, discussed in Supplementary Note J).

Equally important, the PeRCNN shows good interpretability due to the multiplicative form of the Π-block. An analytical expression that governs the underlying physics can be further extracted from the learned model via symbolic computation. In particular, we successfully marry PeRCNN to the sparse regression algorithm to solve the crucial PDE discovery issues. The coupled scheme enables us to iteratively optimize the network parameters, and fine-tune the discovered PDE structures and coefficients, essentially leading to the final parsimonious closed-form PDEs. The resulting framework will serve as an effective, interpretable and flexible approach to accurately and reliably discover the underlying physical laws from imperfect and coarse-meshed measurements.

Although the PeRCNN shows promise in data-driven modelling of complex systems, it is restricted by the computational bottleneck due to the high dimensionality of the discretized system, especially when it comes to systems in a large 3D spatial domain with long-term evolution. However, this issue is expected to be addressed via temporal batch and multi-graphics-processing-unit training. In addition, the current model is rooted in standard convolution operations, which limits its applicability to irregular meshes of arbitrary computational geometries. This issue might be resolved by introducing graph convolution into the network architecture. Lastly, as the PeRCNN network is designed based on the assumption that the underlying governing PDEs have a polynomial form (commonly seen in standard PDEs for modelling of physics such as diffusion, reaction, convection and rotation), it might be less capable or too redundant (if many channels are used to achieve a high polynomial degree) of modelling unique spatiotemporal dynamics whose governing PDEs are parsimonious but involve other advanced symbolic operators such as division, sin, cos, exp, tan, sinh, log and so on. Although the PeRCNN shows success in data-driven modelling of a PDE system with a non-polynomial term in Supplementary Note I, how to design a network that properly incorporates a limited number of mathematical operators as symbolic activation functions to improve the representation ability still remains an open question. We aim to systematically address these issues in our future study.

Methods

We herein introduce the method of the proposed PeRCNN model. More details can be found in Supplementary Note B.

Network architecture

Let us first consider a spatiotemporal dynamical system described by a set of nonlinear, coupled PDEs as

$$\mathbf{u}_t = \mathcal{F}\left(\mathbf{x}, t, \mathbf{u}, \mathbf{u}^2, \nabla_{\mathbf{x}}\mathbf{u}, \mathbf{u}\cdot\nabla_{\mathbf{x}}\mathbf{u}, \nabla^2\mathbf{u}, \cdots\right) \tag{7}$$

where $\mathbf{u}(\mathbf{x}, t) \in \mathbb{R}^n$ denotes the state variable with n components defined over the spatiotemporal domain $\{(\mathbf{x}, t)\} \in \Omega\times\mathcal{T}$. Here, Ω and $\mathcal{T}$ represent the spatial and temporal domains, respectively; $\nabla_{\mathbf{x}}$ is the nabla operator with respect to the spatial coordinate $\mathbf{x}$; and $\mathcal{F}(\cdot)$ is a nonlinear function describing the right-hand side of the PDEs. The solution to this problem is subject to the IC $\mathcal{I}(\mathbf{u}; t = 0, \mathbf{x}\in\Omega) = 0$ and the BC $\mathcal{B}(\mathbf{u}, \nabla_{\mathbf{x}}\mathbf{u}, \cdots; \mathbf{x}\in\partial\Omega) = 0$. Since we mainly focus on regular physical domains in this paper, the state variable $\mathbf{u}$ is defined on a discretized Cartesian grid.

Borrowing the concepts of numerical discretization, we build the physics-encoded spatiotemporal learning model on the basis of a forward Euler scheme. That said, the state variable $\mathbf{u}$ would be updated by a recurrent network given by $\hat{\mathbf{u}}^{(k+1)} = \hat{\mathbf{u}}^{(k)} + \hat{\mathcal{F}}\left(\hat{\mathbf{u}}^{(k)}; \theta\right)\delta t$, where δt is the time spacing, $\hat{\mathbf{u}}^{(k)}$ is the prediction at time t_k and $\hat{\mathcal{F}}$ is the approximated $\mathcal{F}$ parameterized by $\boldsymbol{\theta}$ that ensembles a series of operations for computing the right-hand side of equation (7). Similar ideas of applying numerical discretization (for example, backwards Euler or Runge–Kutta) to designing deep learning architectures can be found in some recent literature[5,60–64].

Following the above intuition, here we introduce the proposed network, namely, the PeRCNN. The architecture of this network (as shown in Fig. 1) consists of two major components: a fully convolutional network as the ISG and a novel convolutional block called the Π-block (product) used for recurrent computation. The ISG is introduced to produce the high-resolution initial state $\hat{\mathbf{U}}^{(0)}$ in the case that only low-resolution initial state (or measurement) $\tilde{\mathbf{u}}^{(0)}$ is available as the IC. Note that $\tilde{\mathbf{u}}$ is used to denote the low-resolution snapshots (or measurement) while the superscript '0' indicates the first one. Similarly, $\hat{\mathbf{U}}$ is used to represent the high-resolution prediction from the model. Within the Π-block, the core of PeRCNN, the state variable $\hat{\mathbf{U}}^{(k)}$ from the previous time step goes through multiple parallel convolutional layers. The feature maps produced by these layers are then fused through an element-wise product layer. The 1 × 1 convolutional layer is subsequently used to linearly combine multiple channels into the desired output (that is, approximated $\mathcal{F}$). Mathematically, the Π-block seeks to approximate the function $\mathcal{F}$ via polynomial combination of solution $\hat{\mathbf{U}}^{(k)}$ and its spatial derivatives, given by

$$\hat{\mathcal{F}}\left(\hat{\mathbf{U}}^{(k)}\right) = \sum_{c=1}^{N_c} W_c \cdot \left[\prod_{l=1}^{N_l}\left(\mathcal{K}_{c,l} \circledast \hat{\mathbf{U}}^{(k)} + b_l\right)\right] \tag{8}$$

where N_c and N_l are the numbers of channels and parallel convolutional layers, respectively; $\circledast$ denotes the convolutional operation; $\mathcal{K}_{c,l}$ denotes the weight of the convolutional filter of the lth layer and the cth channel, while b_l represents the bias of the lth layer; and W_c is the weight corresponding to the cth channel in the 1 × 1 convolutional layer while the bias is omitted for simplicity. This multiplicative representation promotes the network expressiveness for nonlinear functions $\mathcal{F}$, compared with the additive representation commonly seen in related work[5,65]. For detailed discussion on the design of the Π-block, please refer to Supplementary Note B.

Due to the discretized scheme of the learning model, it is possible to encode prior physics knowledge of the system into network architecture, which contributes to a well-posed optimization problem. Given some existing terms in the PDE, we could encode these terms into the network by creating a short-cut connection, namely the physics-based FD convolutional connection, from $\hat{\mathbf{U}}^{(k)}$ to $\hat{\mathbf{U}}^{(k+1)}$, as shown in Fig. 1. The convolutional kernel in this physics-based convolutional layer would be fixed with the corresponding FD stencil to account for the known terms. A major advantage of this encoding mechanism over the soft penalty in physics-informed learning models is the capability to leverage the incomplete PDE in the learning. In the numerical examples, we demonstrate that such a highway connection could accelerate the training speed and improve the model inference accuracy significantly. In a nutshell, the physics-based convolutional connection is built to account for the known physics, while the Π-block is designed to learn the complementary unknown dynamics.

In addition to the incomplete PDE, the boundary conditions (for example, Dirichlet or Neumann type) can also be encoded into the learning model. Inspired by the idea from the FD method, we apply the physics-based padding to the model's prediction at each time step, as shown by Fig. 1b. Specifically, for the Dirichlet BCs, we pad the prediction with prescribed values. Likewise, the padding value of Neumann or Robin BCs will be computed based on the boundary values and the gradient information. A comprehensive discussion on the padding mechanism for various BCs (for example, Dirichlet, Neumann, Robin and periodic) can be found in Supplementary Note B.3. In particular, we show the effectiveness of the proposed padding method in Supplementary Note G, where the Neumann BCs are considered for example.

Motivation of the network architecture design

In the 'Physics-encoded spatiotemporal learning' section, we have introduced the proposed network architecture for learning spatiotemporal dynamical systems. Here a further discussion on the design philosophy is presented to showcase the primary motivations. A distinct characteristic of this architecture is the usage of the Π-block as a universal polynomial approximator to nonlinear functions, instead of utilizing a sequence of linear layers intertwined with nonlinear activation layers commonly seen in traditional deep networks. The motivations for introducing the element-wise product operation in the Π-block are threefold:

- Although the nonlinear activation function is crucial to the universal approximation property of the deep neural network, it is also a source of poor interpretability. For example, the conventional deep neural network would form a prolonged nested function that is usually intractable to humans. We consider it unfavourable to use these nonlinear functions to build a recurrent block that aims to generalize the unknown physics.
- The element-wise product operation makes a better approximation to $\mathcal{F}$ in the form of multivariate polynomial (for example, $\mathbf{u} \cdot \nabla u + u^2 v$), which covers a wide range of well-known dynamical systems, such as Navier–Stokes, RD, Lorenz, Schrödinger equations, to name only a few. Since the spatial derivatives can be computed by convolutional filters[66], a Π-block with n parallel convolutional layers of appropriate filter size is able to represent a polynomial up to the nth order.
- Compared with the regression models (for example, linear or symbolic regression) relying on predefined basis functions or prior knowledge (for example, the highest order) on $\mathcal{F}$ (refs. 5,15), the Π-block is flexible at generalizing the nonlinear function $\mathcal{F}$. For example, a Π-block with two parallel layers of appropriate filter size ensembles a family of polynomials up to the second order (for example, u, Δu, uv, $\mathbf{u} \cdot \nabla u$), with no need to explicitly define the basis.

Since the network architecture roots on numerical discretization, nice mathematical properties (see the next section) exist to guarantee the universal polynomial approximation property of the Π-block. In addition, the flexibility to deal with a variety of problems in scientific modelling is another advantage possessed by the proposed network architecture. The Π-block acts as a universal polynomial approximator to unknown nonlinear function while the physics-based convolutional layer accounts for the prior knowledge on the governing equation. Such a way of encoding the prior physics knowledge into the network architecture could effectively narrow down the space of feasible model parameters, hence leading to the reduced training effort required (for example memory, floating point operations per second and so on). Furthermore, in many prediction tasks involving nonlinear system, a mixture of partial physics knowledge and a scarce amount of measurement data of the system is available, which is when the proposed PeRCNN has a huge advantage over the traditional deep network. As shown in the 'Data-driven modelling of spatiotemporal dynamics' section, we demonstrated the capability of the proposed PeRCNN at handling the modelling tasks given limited physics knowledge and some low-resolution snapshots of the system. An extreme case is when the analytical form of a physical system is completely known except some scalar coefficients. The physics-based convolutional layers associated with trainable variables can be created to exactly express the physical system up to discretization error. In such a case, the PeRCNN can be assumed to recover to FD method with some trainable variables. In the 'Inverse analysis of PDE systems' section, we showed how the proposed PeRCNN can be applied to identify the system coefficients from the noisy and scarce data in such a scenario.

Noteworthy, we mainly consider the nonlinear function $\mathcal{F}$ in the form of polynomial, which is very commonly seen in PDEs. Other terms such as trigonometric and exponential functions are not considered in this work for simplicity. However, incorporating them would require no more effort than adding a particular symbolic activation (for example, sin, cos, exp and so on) layer following the convolutional operation. Furthermore, these functions can be approximated by polynomials based on a Taylor series as argued in ref. 15.

Universal polynomial approximation property for the Π-block

In the proposed PeRCNN, the Π-block acts as an universal polynomial approximator to unknown nonlinear functions while the physics-based FD convolutional layer (that is, with FD stencil as the convolutional filter) accounts for the prior knowledge on the governing equation. Notably, the Π-block achieves its nonlinearity through the element-wise product operation (equation (8)), which renders the network better expressiveness compared with the additive form representation $\hat{\mathcal{F}}(\mathbf{u}) = \sum_{1\le i\le N} f_i \cdot (\mathcal{K}_i \circledast \mathbf{u})$ seen in related work[5,65] where N is the number of convolutional layers, $\mathcal{K}_i$ is the convolutional kernel of the ith layer and f_i is the weight of the ith layer's output. To support this claim, we propose the following Theorem and Lemmas to prove that any dynamical system described by equation (7) whose $\mathcal{F}$ is continuous (for example, preferably in the form of polynomial) can be approximated by the proposed network. Without loss of generality, we consider the state variable $\mathbf{u}$ with one components u. Lemma 1 and Lemma 2 guarantee the accuracy of the approximation of $\hat{\mathcal{F}}$ (equation (8)) and the forward computation $\hat{u}^{(k+1)} = \hat{u}^{(k)} + \hat{\mathcal{F}}(\hat{u}^{(k)}; \theta)\,\delta t$, respectively.

Lemma 1. *The trainable convolutional filter $\mathcal{K}$ can approximate any differential operator with prescribed order of accuracy.*

Proof. Consider a bivariate differential operator $\mathcal{L}(\cdot)$, we have[20]:

$$\begin{aligned}\mathcal{L}(u) &= \sum_{k_1,k_2=-\frac{p-1}{2}}^{\frac{p-1}{2}} \mathcal{K}[k_1,k_2] \sum_{i,j=0}^{p-1} \frac{\partial^{i+j}u}{\partial^i x \partial^j y}\Big|_{(x,y)} \frac{k_1^i k_2^j}{i!j!}\delta x^i \delta y^j + \mathcal{O}\left(|\delta x|^{p-1} + |\delta y|^{p-1}\right)\\ &= \sum_{k_1,k_2=-\frac{p-1}{2}}^{\frac{p-1}{2}} \mathcal{K}[k_1,k_2] u(x+k_1\delta x, y+k_2\delta y) + \mathcal{O}\left(|\delta x|^{p-1} + |\delta y|^{p-1}\right)\\ &= \mathcal{K} \circledast u + \mathcal{O}\left(|\delta x|^{p-1} + |\delta y|^{p-1}\right)\end{aligned} \tag{9}$$

where p is the size of the filter indexed by k_1 and k_2. Letting the filter's entry $\mathcal{K}[k_1,k_2]$ be the corresponding Taylor series coefficient, we can see the error of approximation is bounded by $\mathcal{O}(|\delta x|^{p-1} + |\delta y|^{p-1})$.

Lemma 2. *The local truncation error of the forward computation (that is, $\hat{u}^{(k+1)} = \hat{u}^{(k)} + \hat{\mathcal{F}}(\hat{u}^{(k)}; \theta)\,\delta t$) diminishes as δt decreases.*

Proof. With the Taylor expansion of $u^{(k+1)} = u^{(k)} + \mathcal{F}(u^{(k)})\,\delta t + \mathcal{O}(\delta t^2)$, we can see the truncation error of the forward computation converges to zero as δt decreases.

Theorem 1. *Suppose $\mathcal{F}: \mathbb{R}^s \to \mathbb{R}$ is a continuous real-valued function of multidimensional variables $\eta \in \mathbb{R}^s$, where $\boldsymbol{\eta}$ denotes the set of system state $\mathbf{u}$ and its derivative terms, consisting of s elements in total. For any small positive number ϵ, there exist positive integers M and N, real numbers w_j, γ_{ij} and b_j ($i = 1, 2, ..., N$ and $j = 1, 2, ..., M$), and variable set $\mathbf{E} \in \mathbb{R}^{N\times M}$, such that:*

$$\left|\mathcal{F}(\eta) - \sum_{j=1}^{M} w_j \cdot \left[\prod_{i=1}^{N}(\gamma_{ij}E_{ij} + b_i)\right]\right| < \epsilon \tag{10}$$

Proof. Let us first denote the set of system state $\mathbf{u}$ and its derivative terms, consisting of s elements in total, as $\eta = [\,\mathbf{u}, \mathcal{L}_1(\mathbf{u}), \mathcal{L}_2(\mathbf{u}), \ldots, \mathcal{L}_{s-1}(\mathbf{u})\,]^{\mathrm{T}} \in \mathbb{R}^s$. For example, $\eta = [u, v, u_x, v_y, \ldots]^{\mathrm{T}}$. The right-hand side of the PDEs in equation (7) can then be represented by $\mathcal{F}(\eta)$. Based on the multivariate Taylor's theorem, for any small positive number ϵ, there is a real-valued polynomial function $\mathcal{T}$ such that

$$|\mathcal{F}(\eta) - \mathcal{T}(\eta)| < \epsilon \tag{11}$$

Here, $\mathcal{T}(\eta)$ can be expressed as:

$$\mathcal{T}(\eta) = \sum_{n_1=0}^{n}\sum_{n_2=0}^{n}\cdots\sum_{n_s=0}^{n}\mathcal{N}(\eta) \tag{12}$$

where

$$\mathcal{N}(\eta) = c_{n_1}c_{n_2}\cdots c_{n_s}(\eta_1 - \bar{b}_1)^{n_1}(\eta_2 - \bar{b}_2)^{n_2}\cdots(\eta_s - \bar{b}_s)^{n_s} \tag{13}$$

Here, c terms denote the real-valued coefficients, $\bar{b}$ terms denote the biases and n is the maximum polynomial order. For simplicity, we omit the subscripts $\{n_1, ..., n_s\}$ in $\mathcal{N}(\eta)$.

Lemma 3. *For real numbers η, b and c, and integer n', there exist real-valued vector $\boldsymbol{\alpha} \in \mathbb{R}^{n+1}$, real number $\tilde{b}$ and integer $n' \le n$ such that $c(\eta - b)^{n'} = \prod_{i=0}^{n}(\alpha_i\eta - \tilde{b})$ if $\|\boldsymbol{\alpha}\|_0 = n'$.*

Based on Lemma 3, there are real number α terms and $\tilde{b}$ terms such that $\mathcal{N}(\eta)$ can be re-written as:

$$\mathcal{N}(\eta) = \prod_{i=0}^{n}\left[(\alpha_{i1}\eta_1 - \tilde{b}_1)(\alpha_{i2}\eta_2 - \tilde{b}_2)\cdots(\alpha_{is}\eta_s - \tilde{b}_s)\right] \quad \text{s.t.} \quad \|\alpha_k\|_0 = n_k \tag{14}$$

where $\alpha_k = [\alpha_{0k}, \alpha_{1k}, \ldots, \alpha_{nk}]^{\mathrm{T}} \in \mathbb{R}^{n+1}$ is the kth vector of the α terms ($k = 0, 1, ..., s$); $\|\cdot\|_0$ denotes the ℓ_0 norm of a vector. By defining proper weights ($\boldsymbol{\beta} \in \mathbb{R}^{(n+1)s}$) and biases ($\hat{\mathbf{b}} \in \mathbb{R}^{(n+1)s}$), we can further express $\mathcal{N}(\eta)$ by:

$$\mathcal{N}(\eta) = \prod_{i=1}^{(n+1)s}\left[\beta_i\hat{\eta}_i + \hat{b}_i\right] \tag{15}$$

where $\hat{\eta} := \boldsymbol{\ell} \otimes \eta \in \mathbb{R}^{(n+1)s}$ is the Kronecker transformation of $\boldsymbol{\eta}$; $\boldsymbol{\ell} \in \mathbb{R}^{n+1}$ is a column vector with all elements equal to 1; and $\otimes$ denotes the Kronecker product. Note that $\boldsymbol{\beta}$ is sparse. Substituting equation (15) into equation (12), we obtain the equivalent formulation for $\mathcal{T}(\eta)$ as follows:

$$\mathcal{T}(\eta) = \sum_{j=1}^{M} w_j \cdot \left[\prod_{i=1}^{N}(\gamma_{ij}E_{ij} + b_i)\right] \tag{16}$$

where $\mathbf{E} := \tilde{\ell} \otimes \hat{\eta} \in \mathbb{R}^{N\times M}$ is the Kronecker transformation of $\hat{\mathbf{E}}$; $\tilde{\ell} \in \mathbb{R}^M$ is a row vector with all elements equal to 1; $\mathbf{w} \in \mathbb{R}^M$ and $\boldsymbol{\gamma} \in \mathbb{R}^{N\times M}$ denote some properly defined real-valued coefficients; and $\mathbf{b} \in \mathbb{R}^N$ is the bias vector. Note that the formulation of equation (16) can be guaranteed when $M \ge (n+1)^s$ and $N \ge (n+1)s$. Substituting equation (16) into equation (11) can thus prove Theorem 1.

It is noted that the term $\gamma_{ij}E_{ij}$ in equation (16) can be approximated by a series convolutional filters as shown in Lemma 1, inspiring the design of the universal polynomial approximator shown in equation (8). Although Theorem 1 still holds by selecting different values of M, N, $\mathbf{w}$ and $\mathbf{b}$, the approximation capability might be affected (with varying approximation errors). In particular, a small value of n (thus M and N) that represents less polynomial terms used for approximation will probably lead to a large truncation error. Nevertheless, since the proposed universal polynomial approximator is fully learnable, an equivalent model can be achieved by adapting the channel number (see Supplementary Note I). If the underlying form of $\mathcal{F}$ is a polynomial type, Theorem 1 holds with much fewer parameters and terms required for satisfactory accuracy.

Loss functions

We employ different loss functions depending on the problem at hand. In the case of forward analysis of nonlinear systems, we assume the full knowledge on the system (for example, governing equation, ICs, BCs) is available. The most straightforward approach to construct a predictive model would be utilizing the numerical discretization, that is, customizing all the physics-based connections to realize the FD time updating. That is to say, the network architecture parameterizes an explicit solver of PDE system which hence has a requirement on δt for numerical stability. To avoid this issue, we construct the predictive model in an implicit manner. To be more concrete, we employ a Π-block in the network as the approximator to $\mathcal{F}$ and compute the governing equation's residual from the spatiotemporal prediction using high-order FD stencils. The mean squared error (MSE) of the physics residual (following equation (7)) is employed as the loss function, which reads

$$\mathcal{J}(\mathbf{W},\mathbf{b}) = \mathrm{MSE}\left(\hat{\mathbf{U}}_t - \mathcal{F}\left(\hat{\mathbf{U}}\right)\right) \tag{17}$$

where $\hat{\mathbf{U}} \in \mathbb{R}^{n_t \times n \times H \times W}$ is the high-resolution prediction from the model, $\hat{\mathbf{U}}_t$ is the time derivative of $\hat{\mathbf{U}}$ computed through numerical discretization while $\mathcal{F}(\hat{\mathbf{U}})$ is the right-hand side of equation (7), and $(\mathbf{W}, \mathbf{b})$ denotes the trainable parameters of the network. With the gradient descent method for optimization, we can obtain a suitable set of Π-block parameters. This implicit way of establishing predictive model is more stable numerically regarding the selection of δt. We also need to note that the loss of ICs and BCs is not included in the loss function as they are already encoded through customized padding (see 'Physics-encoded spatiotemporal learning').

In the problem of data-driven modelling, the goal is to reconstruct the most likely full-field solution $\hat{\mathbf{U}}$ given some low-resolution snapshots $\tilde{\mathbf{u}} \in \mathbb{R}^{n'_t \times n \times H' \times W'}$ where $n'_t < n_t$, $H' < H$ and $W' < W$. Therefore, the loss function to train the network is defined as

$$\mathcal{J}(\mathbf{W},\mathbf{b}) = \mathrm{MSE}\left(\hat{\mathbf{U}}(\tilde{\mathbf{x}}) - \tilde{\mathbf{u}}\right) + \lambda \cdot \mathrm{MSE}\left(\hat{\mathbf{U}}^{(0)} - \mathcal{P}\left(\tilde{\mathbf{u}}^{(0)}\right)\right) \tag{18}$$

where $\hat{\mathbf{U}}(\tilde{\mathbf{x}})$ denotes the mapping of high-resolution prediction $\hat{\mathbf{U}} \in \mathbb{R}^{n_t \times n \times H \times W}$ on the coarse grid $\tilde{\mathbf{x}}$; $\tilde{\mathbf{u}}$ denotes the low-resolution measurement; $\mathcal{P}(\cdot)$ is a spatial interpolation function (for example, bicubic or bilinear); and λ is the regularizer weighting. The regularization term denotes the IC discrepancy between the interpolated high-resolution initial state $\mathcal{P}\left(\tilde{\mathbf{u}}^{(0)}\right)$ and the predicted high-resolution initial state $\hat{\mathbf{U}}^{(0)}$ from the ISG, which is found to be effective in preventing network overfitting. Compared with the existing work on physics-informed learning[24,38,42,67], one major distinction of the loss function employed here is the absence of the physics loss. This is because the prior physics knowledge is already encoded into the network architecture as shown in the 'Physics-encoded spatiotemporal learning' section. This facilitates the learning process of the spatiotemporal system significantly. Equation (18) is also utilized as the loss function in the problem of system coefficient identification where the noisy and scarce measurement is available. However, a different network design is employed as elaborated in Supplementary Fig. 10. In this case, multiple physics-based convolutional layers are created to represent existing terms in the PDE (for example, Δu, uv^2, u) while each layer (or term) is associated with a trainable variable to represent the corresponding coefficient. By minimizing the the loss function with IC discrepancy regularizer, the unknown scalar coefficients in the system could be obtained.

Equation discovery

The proposed PeRCNN-based PDE discovery model consists of three steps, including data reconstruction, sparse regression and fine-tuning of coefficients, discussed as follows.

Data reconstruction. As the available measurement data collected in the real world are usually sparse and accompanied by noise, it is common practice to pre-process the raw data to reconstruct the high-fidelity data (for example, de-noised or high-resolution). In our proposed framework, the data reconstruction step (Fig. 6a) is first performed with the help of the PeRCNN as a high-resolution data-driven predictive model. This step follows the same routine described in the 'Data-driven modelling of spatiotemporal dynamics' section. Specifically, we establish a data-driven model from some low-resolution snapshots and then infer the high-resolution prediction (or solution) from the trained model. The reconstructed high-resolution data are then employed in the subsequent sparse regression to ensure the accuracy of the constructed library. The derivative terms in the library are estimated via FD-based filtering on the reconstructed high-fidelity data.

Sparse regression. With the reconstructed high-fidelity (that is, high-resolution and de-noised) solution, we are able to reliably estimate the library and thus accurately perform sparse regression for the explicit form or analytical structure of PDEs. Note that sparse regression is an extensively used technique for data-driven PDE discovery. It is rooted on a critical observation that the right-hand side of equation (7) for the majority of natural systems consists of only a few terms. To demonstrate how the sparse regression works, let us consider the measurement data with one single component, that is, $u \in \mathbb{R}^{n_s \times n_t}$, which is defined on n_s spatial locations and at n_t time steps. After flattening the state variable into a column vector $\mathbf{U} \in \mathbb{R}^{n_s \cdot n_t \times 1}$, we are able to establish a library matrix $\mathbf{\Theta}(\mathbf{U}) \in \mathbb{R}^{n_s \cdot n_t \times s}$ such that each of s column vectors denotes a candidate function in $\mathcal{F}$ (for example, linear, nonlinear, trigonometric and so on). Accordingly, each row of $\mathbf{\Theta}(\mathbf{U})$ denotes a spatiotemporal location. If the column space of the library matrix is sufficiently rich, the governing PDE of the system can then be written as a linear system, namely

$$\mathbf{U}_t = \mathbf{\Theta}(\mathbf{U})\mathbf{\Xi} \tag{19}$$

where $\mathbf{U}_t$ is the vector of time derivative of $\mathbf{U}$; $\mathbf{\Theta}(\cdot)$ maps the original state variable space to a higher-dimensional nonlinear space, for example, $\mathbf{\Theta}(\mathbf{U}) = [1, \mathbf{U}, \mathbf{U}^2, \ldots, \mathbf{U}_x, \mathbf{U}_y, \ldots]$; and $\Xi \in \mathbb{R}^{s \times 1}$ is the sparse coefficient vector that represents the governing PDE. Sparse regression seeks to find a suitable $\mathbf{\Xi}$ such that the sparsity of the vector and the regression error are balanced. Specifically in our proposed framework, the STRidge algorithm[16] is adopted among other effective sparsity-promoting methods such as the iterative hard thresholding method[68,69], due to its superior performance compared with other sparsity-promoting algorithms, such as LASSO[70] and sequentially thresholded least squares[15]. For a given tolerance that filters the entries of $\mathbf{\Xi}$ with small value, we can obtain a sparse representation of $\mathcal{F}$ with the help of the STRidge algorithm, the technical details of which are provided in Supplementary Note F.2. Iterative search with STRidge can be performed to find the optimal tolerance according to the selection criteria given by:

$$\mathbf{\Xi}^* = \arg\min_{\mathbf{\Xi}} \left\{ ||\mathbf{U}_t - \mathbf{\Theta}(\mathbf{U})\mathbf{\Xi}||_2^2 + \gamma ||\mathbf{\Xi}||_0 \right\} \tag{20}$$

where $||\mathbf{\Xi}||_0$ is used to measure the sparsity of coefficient vector while regression error $||\mathbf{U}_t - \mathbf{\Theta}(\mathbf{U})\mathbf{\Xi}||_2^2$ is used to measure model accuracy. As the optimization objective has two components, we apply Pareto front analysis to select an appropriate weighting coefficient γ (Supplementary Note F.5.3). As the accurate computation of the library matrix is critical to obtaining an accurate coefficient vector via sparse regression, the reconstructed high-fidelity solution is subsequently used for computing the partial derivative terms involved in the library.

As shown in Fig. 6b, we collect the candidate set from the network for data reconstruction by performing symbolic computations on the Π-block and establish the library matrix $\mathbf{\Theta}(\mathbf{U})$. Note this is different from the traditional sparse regression in which the candidate set is predefined. A comparative study of these two ways of establishing the library matrix is performed in Supplementary Note F.5.1. With the established linear system, sparse regression is performed afterwards to find a suitable coefficient vector $\mathbf{\Xi}$ that balances model complexity and accuracy. This is realized by solving the optimization problem described by equation (20) with the help by the STRidge algorithm.

Fine-tuning of coefficients. Due to the high dimensionality of the reconstructed data, the sparse regression is performed on the subsampled linear system (for example, randomly sampled 10% rows, 8.2×10^6 rows in the 2D GS RD case) to avoid the very large number of rows, which retains computational efficiency without the loss of accuracy. To fully exploit the available measurement and further improve the accuracy of the discovered equations, we introduce the coefficient fine-tuning step to produce the final explicit PDE formula. The rest of training procedure is the same as that discussed in 'Inverse analysis of PDE systems': all the original measurements are used to train a PDE structure preserved network (Fig. 6c) while the coefficient of each term is treated as a trainable variable. In Supplementary Note F.5.4, we show that such a fine-tuning can considerably improve the accuracy of the discovered PDEs.

Data availability

All the used datasets in this study are available in the Zenodo repository[71], the Gitee repository at https://gitee.com/chengzrz/percnn and the GitHub repository at https://github.com/isds-neu/PeRCNN.

Code availability

All the source codes to reproduce the results in this study are available in the Zenodo repository[71], the Gitee repository at https://gitee.com/chengzrz/percnn and the GitHub repository at https://github.com/isds-neu/PeRCNN.

References

1. Raissi, M., Perdikaris, P. & Karniadakis, G. E. Machine learning of linear differential equations using Gaussian processes. *J. Comput. Phys.* **348**, 683–693 (2017).
2. Han, J., Jentzen, A. & Weinan, E. Solving high-dimensional partial differential equations using deep learning. *Proc. Natl Acad. Sci. USA* **115**, 8505–8510 (2018).
3. Bar-Sinai, Y., Hoyer, S., Hickey, J. & Brenner, M. P. Learning data-driven discretizations for partial differential equations. *Proc. Natl Acad. Sci. USA* **116**, 15344–15349 (2019).
4. Sanchez-Gonzalez, A. et al. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning* 8459–8468 (PMLR, 2020).
5. Long, Z., Lu, Y., Ma, X. & Dong, B. PDE-Net: learning PDEs from data. In *International Conference on Machine Learning* 3208–3216 (PMLR, 2018).
6. Wang, R., Kashinath, K., Mustafa, M., Albert, A. & Yu, R. Towards physics-informed deep learning for turbulent flow prediction. In *Proc. 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* 1457–1466 (ACM, 2020).
7. Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A. & Battaglia, P. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations* (OpenReview.net, 2021).
8. de Avila Belbute-Peres, F., 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).
9. Kochkov, D. et al. Machine learning-accelerated computational fluid dynamics. *Proc. Natl Acad. Sci. USA* **118**, e2101784118 (2021).
10. Erichson, N. B. Shallow neural networks for fluid flow reconstruction with limited sensors. *Proc. R. Soc. A* **476**, 20200097 (2020).
11. Stengel, K., Glaws, A., Hettinger, D. & King, R. N. Adversarial super-resolution of climatological wind and solar data. *Proc. Natl Acad. Sci. USA* **117**, 16805–16815 (2020).
12. Fukami, K., Fukagata, K. & Taira, K. Machine-learning-based spatio-temporal super resolution reconstruction of turbulent flows. *J. Fluid Mech.* **909**, A9 (2021).
13. Rao, C. & Liu, Y. Three-dimensional convolutional neural network (3D-CNN) for heterogeneous material homogenization. *Comput. Mater. Sci.* **184**, 109850 (2020).
14. Schmidt, M. & Lipson, H. Distilling free-form natural laws from experimental data. *Science* **324**, 81–85 (2009).
15. Brunton, S. L., Proctor, J. L. & Kutz, J. N. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proc. Natl Acad. Sci. USA* **113**, 3932–3937 (2016).
16. Rudy, S. H., Brunton, S. L., Proctor, J. L. & Kutz, J. N. Data-driven discovery of partial differential equations. *Sci. Adv.* **3**, e1602614 (2017).
17. Udrescu, S.-M. & Tegmark, M. AI Feynman: a physics-inspired method for symbolic regression. *Sci. Adv.* **6**, eaay2631 (2020).
18. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436–444 (2015).
19. Lusch, B., Kutz, J. N. & Brunton, S. L. Deep learning for universal linear embeddings of nonlinear dynamics. *Nat. Commun.* **9**, 4950 (2018).
20. 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).
21. Chen, Z., Liu, Y. & Sun, H. Physics-informed learning of governing equations from scarce data. *Nat. Commun.* **12**, 6136 (2021).
22. Cranmer, M. D. et al. Discovering symbolic models from deep learning with inductive biases. In *Advances in Neural Information Processing Systems* (Curran Associates, 2020).
23. Karpatne, A. et al. Theory-guided data science: a new paradigm for scientific discovery from data. *IEEE Trans. Knowl. Data Eng.* **29**, 2318–2331 (2017).
24. 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).
25. Raissi, M., Yazdani, A. & Karniadakis, G. E. Hidden fluid mechanics: learning velocity and pressure fields from flow visualizations. *Science* **367**, 1026–1030 (2020).
26. Karniadakis, G. E. et al. Physics-informed machine learning. *Nat. Rev. Phys.* **3**, 422–440 (2021).
27. Rao, C., Sun, H. & Liu, Y. Physics-informed deep learning for incompressible laminar flows. *Theor. Appl. Mech. Lett.* **10**, 207–212 (2020).
28. Sheng, H. & Yang, C. PFNN: a penalty-free neural network method for solving a class of second-order boundary-value problems on complex geometries. *J. Comput. Phys.* **428**, 110085 (2021).
29. Sun, L., Gao, H., Pan, S. & Wang, J.-X. Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Comput. Methods Appl. Mech. Eng.* **361**, 112732 (2020).
30. Kim, Y., Choi, Y., Widemann, D. & Zohdi, T. A fast and accurate physics-informed neural network reduced order model with shallow masked autoencoder. *J. Comput. Phys.* **451**, 110841 (2021).
31. Yang, Y. & Perdikaris, P. Adversarial uncertainty quantification in physics-informed neural networks. *J. Comput. Phys.* **394**, 136–152 (2019).
32. Zhu, Y., Zabaras, N., Koutsourelakis, P.-S. & Perdikaris, P. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *J. Comput. Phys.* **394**, 56–81 (2019).
33. Haghighat, E., Raissi, M., Moure, A., Gomez, H. & Juanes, R. A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics. *Comput. Methods Appl. Mech. Eng.* **379**, 113741 (2021).
34. Jin, X., Cai, S., Li, H. & Em Karniadakis, G. NSFnets (Navier–Stokes flow nets): physics-informed neural networks for the incompressible Navier–Stokes equations. *J. Comput. Phys.* **426**, 109951 (2021).
35. He, Q. Z., Barajas-Solano, D., Tartakovsky, G. & Tartakovsky, A. M. Physics-informed neural networks for multiphysics data assimilation with application to subsurface transport. *Adv. Water Resour.* **141**, 103610 (2020).
36. He, Q. Z. & Tartakovsky, A. M. Physics-informed neural network method for forward and backward advection-dispersion equations. *Water Resour. Res.* **57**, e2020WR029479 (2021).

37. Zhang, R., Liu, Y. & Sun, H. Physics-informed multi-lstm networks for metamodeling of nonlinear structures. *Comput. Methods Appl. Mech. Eng.* **369**, 113226 (2020).
38. Rao, C., Sun, H. & Liu, Y. Physics-informed deep learning for computational elastodynamics without labeled data. *J. Eng. Mech.* **147**, 04021043 (2021).
39. Niaki, S. A., Haghighat, E., Campbell, T., Poursartip, A. & Vaziri, R. Physics-informed neural network for modelling the thermochemical curing process of composite-tool systems during manufacture. *Comput. Methods Appl. Mech. Eng.* **384**, 113959 (2021).
40. Weinan, E. & Yu, B. The Deep Ritz Method: a deep learning-based numerical algorithm for solving variational problems. *Commun. Math. Stat.* **6**, 1–12 (2018).
41. Ren, P., Rao, C., Liu, Y., Wang, J.-X. & Sun, H. PhyCRNet: physics-informed convolutional-recurrent network for solving spatiotemporal PDEs. *Compu. Methods Appl. Mech. Eng.* **389**, 114399 (2022).
42. Gao, H., Sun, L. & Wang, J.-X. PhyGeoNet: physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain. *J. Comput. Phys.* **428**, 110079 (2021).
43. Gao, H., Zahr, M. J. & Wang, J.-X. Physics-informed graph neural galerkin networks: a unified framework for solving PDE-governed forward and inverse problems. *Comput. Methods Appl. Mech. Eng.* **390**, 114502 (2022).
44. Geneva, N. & Zabaras, N. Transformers for modeling physical systems. *Neural Netw.* **146**, 272–289 (2021).
45. Geneva, N. & Zabaras, N. Modeling the dynamics of PDE systems with physics-constrained deep auto-regressive networks. *J. Comput. Phys.* **403**, 109056 (2020).
46. Gao, H., Sun, L. & Wang, J.-X. Super-resolution and denoising of fluid flow using physics-informed convolutional neural networks without high-resolution labels. *Phys. Fluids* **33**, 073603 (2021).
47. Lu, L., Jin, P., Pang, G., Zhang, Z. & Karniadakis, G. E. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nat. Mach. Intell.* **3**, 218–229 (2021).
48. Li, Z. et al. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations* (OpenReview.net, 2021).
49. Wang, S., Wang, H. & Perdikaris, P. Learning the solution operator of parametric partial differential equations with physics-informed deeponets. *Sci. Adv.* **7**, eabi8605 (2021).
50. Halatek, J. & Frey, E. Rethinking pattern formation in reaction–diffusion systems. *Nat. Phys.* **14**, 507–514 (2018).
51. Holmes, E. E., Lewis, M. A., Banks, J. E. & Veit, R. R. Partial differential equations in ecology: spatial interactions and population dynamics. *Ecology* **75**, 17–29 (1994).
52. Vervloet, D., Kapteijn, F., Nijenhuis, J. & van Ommen, J. R. Fischer–Tropsch reaction–diffusion in a cobalt catalyst particle: aspects of activity and selectivity for a variable chain growth probability. *Catal. Sci. Technol.* **2**, 1221–1233 (2012).
53. Maini, P. K., McElwain, D. L. S. & Leavesley, D. I. Traveling wave model to interpret a wound-healing cell migration assay for human peritoneal mesothelial cells. *Tissue Eng.* **10**, 475–482 (2004).
54. Shi, X. et al. Convolutional LSTM network: a machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems* 802–810 (Curran Associates, 2015).
55. Liao, Q. & Poggio, T. Bridging the gaps between residual learning, recurrent neural networks and visual cortex. Preprint at *arXiv* https://arxiv.org/abs/1604.03640 (2016).
56. Zhang, J., Zheng, Y. & Qi, D. Deep spatio-temporal residual networks for citywide crowd flows prediction. In *Proc. AAAI Conference on Artificial Intelligence* Vol. 31, 1655–1661 (AAAI, 2017).
57. Raissi, M. Deep hidden physics models: deep learning of nonlinear partial differential equations. *J. Mach. Learn. Res.* **19**, 932–955 (2018).
58. Rao, C., Ren, P., Liu, Y. & Sun, H. Discovering nonlinear PDEs from scarce data with physics-encoded learning. In *International Conference on Learning Representations* (OpenReview.net, 2022).
59. Meurer, A. SymPy: symbolic computing in Python. *PeerJ Comput. Sci.* **3**, e103 (2017).
60. He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition* 770–778 (IEEE, 2016).
61. Chen, Y., Yu, W. & Pock, T. On learning optimized reaction diffusion processes for effective image restoration. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition* 5261–5269 (IEEE, 2015).
62. 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).
63. Ruthotto, L. & Haber, E. Deep neural networks motivated by partial differential equations. *J. Math. Imaging Vis.* **62**, 352–364 (2019).
64. Larsson, G., Maire, M. & Shakhnarovich, G. FractalNet: ultra-deep neural networks without residuals. In *International Conference on Learning Representations* (OpenReview.net, 2017).
65. Le Guen, V. & Thome, N. Disentangling physical dynamics from unknown factors for unsupervised video prediction. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition* 11474–11484 (IEEE, 2020).
66. Cai, J.-F., Dong, B., Osher, S. & Shen, Z. Image restoration: total variation, wavelet frames, and beyond. *J. Am. Math. Soc.* **25**, 1033–1089 (2012).
67. Raissi, M., Wang, Z., Triantafyllou, M. S. & Karniadakis, G. E. Deep learning of vortex-induced vibrations. *J. Fluid Mech.* **861**, 119–137 (2019).
68. Haupt, J. & Nowak, R. Signal reconstruction from noisy random projections. *IEEE Trans. Inf. Theor.* **52**, 4036–4048 (2006).
69. Blumensath, T. & Davies, M. E. Iterative hard thresholding for compressed sensing. *Appl. Comput. Harmon. Anal.* **27**, 265–274 (2009).
70. Tibshirani, R. Regression shrinkage and selection via the LASSO. *J. R. Stat. Soc. Ser. B* **58**, 267–288 (1996).
71. isds-neu & Ren, P. isds-neu/PeRCNN: encoding physics to learn reaction-diffusion processes. *Zenodo* https://doi.org/10.5281/zenodo.7955830 (2023).

Acknowledgements

The work is supported by the National Natural Science Foundation of China (no. 92270118 and no. 62276269), the Beijing Natural Science Foundation (no. 1232009), the National Key R&D Program of China (no. 2021ZD0110400) and the Beijing Outstanding Young Scientist Program (no. BJJWZYJH012019100020098). We also acknowledge the support by the Huawei MindSpore platform. Y.L. and H.S. acknowledge the support from the Fundamental Research Funds for the Central Universities. C.R. acknowledges the sponsorship of visiting research by H.S. at Renmin University of China.

Author contributions

C.R., H.S. and Y.L. contributed to the ideation and design of the research. C.R., P.R. and Q.W. performed the research. C.R., P.R., O.B., H.S. and Y.L. wrote the paper.

Competing interests

The authors declare no competing interests.

Additional information

Extended data is available for this paper at https://doi.org/10.1038/s42256-023-00685-7.

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s42256-023-00685-7.

Correspondence and requests for materials should be addressed to Hao Sun or Yang Liu.

Peer review information *Nature Machine Intelligence* thanks Ilias Bilionis and Lu Lu for their contribution to the peer review of this work.

https://doi.org/10.1038/s42256-023-00685-7

Extended Data Table 1 | Computational parameters for datasets generation

Dataset	Equation	Parameters	Dimensions	δt	δx
2D λ-Ω	Eq. (2)	μ_u=0.1, μ_v=0.1 β=1.0	$101^2 \times 801$	0.0125	0.2
2D FN	Eq. (3)	$\mu_u = 1.0,\ \mu_v = 10.0,\ \alpha = 0.01,\ \beta = 0.25$	$101^2 \times 6001$	0.002	1.0
3D FN	Eq. (3)	$\mu_u = 1.0,\ \mu_v = 10.0,\ \alpha = 0.01,\ \beta = 0.25$	$51^3 \times 1001$	0.004	1.0
2D GS	Eq. (4)	$\mu_u = 2.0e-5,\ \mu_v = 5.0e-6,\ F = 0.04,\ \kappa = 0.06$	$101^2 \times 2501$	0.5	0.01
3D GS	Eq. (4)	$\mu_u = 0.2,\ \mu_v = 0.1,\ F = 0.025,\ \kappa = 0.055$	$49^3 \times 1501$	0.5	$25/12$

δx denotes the spacing of the grid while δt denotes the time spacing. A detailed discussion on how to properly select δx and δt is given in Supplementary Note H.

https://doi.org/10.1038/s42256-023-00685-7

Extended Data Table 2 | Summary of the coefficient identification results for 2D Gray–Scott reaction–diffusion system

Scenario	Noise (%)	$\mu_u(10^{-5})$	$\mu_v(10^{-6})$	c_1	c_2	$c_F(10^{-2})$	$c_\kappa(10^{-2})$	MARE (%)
Ground truth	-	2.0	5.0	1.0	1.0	4.0	6.0	-
S1	0	1.987	4.989	0.9920	0.9941	3.970	5.965	0.60
	10	1.950	5.010	0.9724	0.9823	3.938	5.941	1.61
S2	0	1.981	5.124	0.9886	0.9993	4.014	6.046	0.96
	10	1.964	5.111	0.9864	0.9987	4.003	6.044	1.05

The training dataset (or measurement) includes 26 snapshots with resolution of 26 × 26 for S1 while 2 snapshots with resolution of 51 × 51 for S2.

https://doi.org/10.1038/s42256-023-00685-7

Extended Data Table 3 | Discovered PDEs from the measurement data under various noise levels compared with the ground truth

Example	Noise	Discovered PDE
2D λ–Ω RD	0%	$u_t = 0.096\Delta u + 1.013u - 1.019u^3 + 1.001u^2v - 1.021uv^2 + 0.9977v^3$ $v_t = 0.096\Delta v + 1.006v - 0.998u^3 - 1.0139u^2v - 1.002uv^2 - 1.012v^3$
	5%	$u_t = 0.096\Delta u + 1.038u - 1.048u^3 + 1.004u^2v - 1.050uv^2 + 0.998v^3$ $v_t = 0.100\Delta v + 1.014v - 0.998u^3 - 1.025u^2v - 0.999uv^2 - 1.015v^3$
	10%	$u_t = 0.101\Delta u + 1.079u - 1.090u^3 + 1.008u^2v - 1.090uv^2 + 0.9982v^3$ $v_t = 0.105\Delta v + 1.033v - 0.965u^3 - 1.046u^2v - 0.967uv^2 - 1.029v^3 + \underline{0.029u}$
	Truth	$u_t = 0.1\Delta u + (1 - u^2 - v^2)u + (u^2 + v^2)v$ $v_t = 0.1\Delta v - (u^2 + v^2)u + (1 - u^2 - v^2)v$
2D GS RD	0%	$u_t = 1.999 \times 10^{-5}\Delta u - 0.992uv^2 - 0.04003u + 0.03999$ $v_t = 5.008 \times 10^{-6}\Delta v + 1.021uv^2 - 0.1001v$
	5%	$u_t = 2.001 \times 10^{-5}\Delta u - 1.003uv^2 - 0.04008u + 0.04008$ $v_t = 5.042 \times 10^{-6}\Delta v + 1.009uv^2 - 0.1007v$
	10%	$u_t = 1.846 \times 10^{-5}\Delta u - 0.904uv^2 - \underline{0.0863u^3} + 0.04019$ $v_t = 5.438 \times 10^{-6}\Delta v + 1.051uv^2 - 0.1174v$
	Truth	$u_t = 2.0 \times 10^{-5}\Delta u - 1.0uv^2 + 0.04(1 - u)$ $v_t = 5.0 \times 10^{-6}\Delta v + 1.0uv^2 - 0.1v$