HARD ENCODING OF PHYSICS FOR LEARNING SPATIOTEMPORAL DYNAMICS

Chengping Rao, Hao Sun & Yang Liu*
Northeastern University, Boston, MA, USA
{rao.che, h.sun, yang1.liu}@northeastern.edu

ABSTRACT

Modeling nonlinear spatiotemporal dynamical systems has primarily relied on partial differential equations (PDEs). However, the explicit formulation of PDEs for many underexplored processes, such as climate systems, biochemical reaction and epidemiology, remains uncertain or partially unknown, where very limited measurement data is yet available. To tackle this challenge, we propose a novel deep learning architecture that forcibly encodes known physics knowledge to facilitate learning in a data-driven manner. The coercive encoding mechanism of physics, which is fundamentally different from the penalty-based physics-informed learning, ensures the network to rigorously obey given physics. Instead of using nonlinear activation functions, we propose a novel elementwise product operation to achieve the nonlinearity of the model. Numerical experiment demonstrates that the resulting physics-encoded learning paradigm possesses remarkable robustness against data noise/scarcity and generalizability compared with some state-of-the-art models for data-driven modeling.

1 BACKGROUND

Partial differential equations (PDEs) have played an indispensable role in modeling complex dynamical systems or processes. However, there still exist a considerable portion of dynamical systems, such as those in epidemiology and climate science, whose governing PDEs are unclear or only partially known. To give prediction on systems like these, there have been efforts seeking alternatives to the physics-based models. In recent years, increasing amount of attempts have been made on leveraging physics principles to inform deep neural networks (DNN) for the modeling of systems in a data-driven manner. Among those approaches, the physics-informed neural networks (PINN) is a representative one that can be employed in data-driven modeling (Raissi, 2018), as well as solving forward and inverse PDE problems (Raissi et al., 2019a;b; 2020; Rao et al., 2020). In the framework of PINN, the network is usually informed by physics through a weakly imposed penalty loss consisting of residuals of PDEs and initial/boundary conditions (I/BCs).

Although PINN has achieved success in modeling dynamical systems, one of its major limitations is that its accuracy relies largely on the these soft physical constraints (Wang et al., 2020b; Rao et al., 2021) which may not be satisfied well during training due to the ill-posedness of the optimization problem. Furthermore, the use of fully connected layers poses intrinsic limitations to low-dimensional parameterizations. Efforts have been placed to overcome these issues by employing discrete learning schemes via convolutional filters, such as HybridNet (Long et al., 2018a), dense convolutional encoder-decoder network (Zhu et al., 2019), auto-regressive encoder-decoder model (Geneva & Zabaras, 2020), TF-Net (Wang et al., 2020a), DiscretizationNet (Ranade et al., 2021) and PhyGeoNet (Gao et al., 2021). These methods generally show better computational efficiency and accuracy. However, the core learning component of these networks is still a black box and the resulting models lack the capability to "hard-encode" our prior physical knowledge. How to construct a data-driven model that both fits limited data and generalizes well the underlying physics remains a critical challenge.

To address this challenge, in this work, we propose a physics-encoded recurrent-convolutional neural network (PeRCNN), which forcibly encodes the physics structure to facilitate learning for data-

*Corresponding author.

driven modeling of nonlinear systems. The physics-encoding mechanism guarantees the model to rigorously obey the given physics based on our prior knowledge. Instead of using activation functions, which results in poor interpretability and generalizability, we achieve nonlinear approximation via elementwise product among the feature maps, leading to a recurrent Π-block that renders PeRCNN with good expressiveness and flexibility at representing complex nonlinear physics. The Π-block mimics governing terms in a PDE. The spatial dependency is learned by either convolutional or predefined finite-difference-based filters while the temporal evolution is modeled by a forward Euler time marching scheme. Numerical experiments demonstrate that PeRCNN outperforms the existing models (e.g., ConvLSTM, ResNet and DHPM) in the metrics of generalizability.

Our work is closely related to the deep residual network (ResNet) which addresses the notorious problem of gradient vanishing/exploding for very deep networks with residual learning (He et al., 2016). Our network architecture also features the residual connection that enables the residual learning of dynamical systems, which has been interpreted by others as the forward Euler time-stepping scheme (Chen et al., 2015; Chang et al., 2017; Chen et al., 2018; Ruthotto & Haber, 2019). People have employed the recurrent ResNet, a variant of ResNet whose parameters are shared across time, to solve spatiotemporal prediction problems (Liao & Poggio, 2016; Zhang et al., 2017). Although ResNet has shown success in lots of applications, residual blocks composed of traditional convolutional or fully connected layers still face the issue of poor interpretability, hence hindering its applications to spatiotemporal dynamical systems where governing PDEs are potentially available.

2 METHODOLOGY

In this part, we elaborate the principle for designing the network architecture of PeRCNN for data-driven modeling of spatiotemporal dynamical systems. The formulation to establish a data-driven model from limited and noisy measurements is introduced in A.2 with more details.

2.1 NETWORK ARCHITECTURE

Our proposed PeRCNN (see Fig. 1) consists of two major components: a fully convolutional (Conv) network as initial state generator (ISG) and an unconventional Conv block, namely Π-block (product), for recurrent computation, as depicted in Fig. 1(a). ISG enables a mapping from the noisy low-resolution measurement $\tilde{\mathbf{u}}_0$ to the full-resolution initial state $\hat{\boldsymbol{\mathcal{U}}}_0$ from which the recurrent computation can be initiated.

In the Π-block shown in Fig. 1(b), which is the core of PeRCNN, the state variable $\hat{\boldsymbol{\mathcal{U}}}_k$ from the previous time step first goes through multiple parallel Conv layers, whose feature maps will then be fused via an elementwise product layer. A 1×1 Conv layer (or network in network (Lin et al., 2013)) is appended after the product operation to aggregate multiple channels into the output of desired number of channels. Assuming the output of the 1×1 Conv layer approximates the nonlinear function $\mathcal{F}(\cdot)$, we multiply it by the time spacing δt to obtain the residual of the dynamical system at time t_k, i.e., $\delta\hat{\boldsymbol{\mathcal{U}}}_k$. The Π-block operation is expressed as:

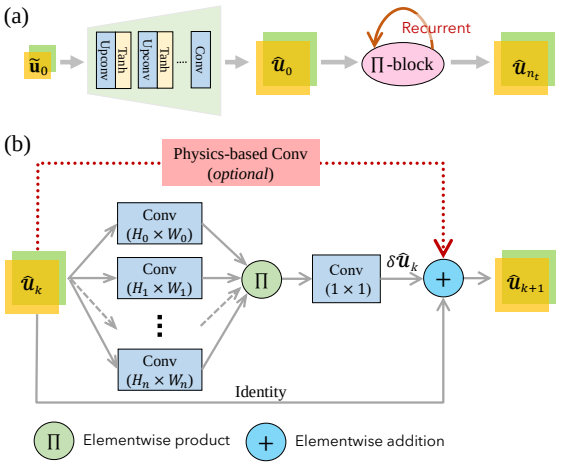


Figure 1: Schematic architecture of PeRCNN: (a) the network with a recurrent Π-block folded; and (b) Π-block for the recurrent computation. Here, $\tilde{\mathbf{u}}_0$ is the low-resolution noisy initial state measurement, while $\hat{\boldsymbol{\mathcal{U}}}_k$ is the predicted full-resolution solution at time t_k. The decoder (initial state generator) is used to downscale/upsample the low-resolution initial state.

$$\hat{\boldsymbol{\mathcal{U}}}_{k+1} = \hat{\boldsymbol{\mathcal{U}}}_k + \left\{ \left[\prod_{i=0}^{n} \left(\hat{\boldsymbol{\mathcal{U}}}_k * \mathbf{W}_i + \mathbf{b}_i \right) \right] * \mathbf{W}^{(1)} + \mathbf{b}^{(1)} \right\} \delta t \tag{1}$$

where $*$ denotes the Conv operation; $\mathbf{W}_i$ and $\mathbf{b}_i$ are the weight and bias for the filter in the i^{th} layer while $\mathbf{W}^{(1)}$ and $\mathbf{b}^{(1)}$ correspond to those of the 1×1 Conv filter; $n+1$ is the number of parallel Conv layers; Π denotes the elementwise product; δt is the selected time spacing. It should be noted that a highway physics-based Conv layer (see Fig 1(b)) could be created when some specific terms are known *a priori* in the PDEs. Such a highway connection is not necessary to the succeed of data-driven modeling but could accelerate the training speed and improve the extrapolation accuracy.

It is worth noting that we achieve the nonlinearity of our network via elementwise product of the feature maps instead of using nonlinear activation functions, mainly for three reasons: **(1)** Though the nonlinear activation function is crucial to the expressiveness of the DL model, it is also a source of poor interpretability. We consider it unfavorable to use these nonlinear functions to build a recurrent block that aims to generalize the unknown physics. **(2)** The nonlinear function $\mathcal{F}$ in the form of polynomial[1] covers a wide range of well-known dynamical systems, such as Navier-Stokes, reaction-diffusion (RD), Lorenz, Schrödinger equations, to name only a few. Since the spatial derivatives can be computed by Conv filters (Cai et al., 2012), a Π-block with n parallel Conv layers of appropriate filter size is able to represent a polynomial up to the n^{th} order. **(3)** Compared with the regression models that heavily rely on predefined basis functions (Brunton et al., 2016), the Π-block is flexible at generalizing the function $\mathcal{F}$. For example, a Π-block with 2 parallel layers of suitable filter size ensembles a family of polynomials up to 2^{nd} order (e.g., u, Δu, uv, $\mathbf{u} \cdot \nabla u$), with no need to explicitly define the basis.

Although we mainly consider the nonlinear function $\mathcal{F}$ in the form of polynomial, terms of other forms such as trigonometric and exponential functions can be incorporated by adding a particular symbolic activation (e.g., sin, cos, exp, etc.) layer following the Conv operation.

2.2 Hard encoding mechanism of physics

The encoding mechanism is employed to strictly impose the prior physical knowledge of the system to PeCRNN, which contributes to a well-posed optimization problem. In this work, two types of physics can be considered for hard-encoding, namely, the prior knowledge on I/BCs and active terms in the governing PDEs. The ICs (or initial states) can be naturally imposed when PeRCNN starts the recurrent computation from $\tilde{\mathbf{u}}_0$. For the BCs (Dirichlet or Neumann type), we borrow the idea from the finite difference (FD) method and apply the physics-based padding to the model's prediction at each time step (i.e., $\hat{\boldsymbol{\mathcal{U}}}_k$). More specifically, we pad the prediction with prescribed values defined by the Dirichlet BCs. The padding operations of the Neumann BCs will be computed based on the boundary values and their gradient information.

PeRCNN also has the capability to encode prior-known terms in the governing PDEs via a highway Conv layer (see Fig. 1(b)) with predefined FD-based filters. In Section 3, we consider a reaction-diffusion system and assume the diffusion term $\Delta \mathbf{u}$ is known. Therefore, a Conv layer with discrete Laplacian operator (see Eq. (4)) as its filter is created to approximate $\Delta \mathbf{u}$. The associated diffusion coefficient is unknown and placed as part of the trainable variables. It should be noted that, by using the residual connection in the recurrent Π-block, we also implicitly encode the existing term of $\mathbf{u}_t$.

3 Experiments

3.1 Datasets

In the experiments, we employ two different dynamical systems, 2D Burgers' equation and 3D Gray-Scott (GS) reaction-diffusion (RD) equation, to examine our approach. The 2D Burgers' equation, with a wide applications in applied mathematics, such as fluid/traffic flow modeling, is given by

$$\mathbf{u}_t + \mathbf{u} \cdot \nabla \mathbf{u} = \nu \Delta \mathbf{u} \tag{2}$$

where $\mathbf{u} = [u,\ v]^{\mathrm{T}}$ denotes the fluid velocities; ∇ is the Nabla operator; Δ is the Laplacian operator and ν is the viscosity coefficient selected to be 0.005 in this case. We consider a computational domain of $\Omega \times [0, T] = [-0.5, 0.5]^2 \times [0, 0.4]$ under periodic boundary conditions and generate the solution on a $101^2 \times 1601$ spatiotemporal grid.

[1] Polynomial herein encompasses linear derivative terms, e.g., $\mathbf{u} \cdot \nabla u + u^2 v$ has 2^{nd} and 3^{rd} order terms.

The second dataset considered is the 3D GS-RD equation, which can be described by

$$\mathbf{u}_t = \mathbf{D}\Delta\mathbf{u} + \mathbf{R}(\mathbf{u}) \tag{3}$$

Here, $\mathbf{u} = [u,\ v]^\intercal$ is the concentration vector; $\mathbf{D} = \text{diag}(\mu_u, \mu_v)$ is the diagonal diffusion coefficient matrix; $\mathbf{R}(\mathbf{u}) = [-uv^2 + f(1-u),\ uv^2 - (f+\kappa)v]^\intercal$ is the nonlinear reaction vector, where κ and f denote the kill and feed rate, respectively. This model has found wide applications in computational chemistry and biochemistry. In our dataset, the parameters of $\mu_u = 0.2,\ \mu_v = 0.1,\ \kappa = 0.055$ and $f = 0.025$ are employed. We consider the physical domain of $\Omega \times [0, T] = [-50, 50]^3 \times [0, 750]$ with periodic boundary condition and generate the solution on a Cartesian grid ($49^3 \times 1501$) using the FD method. For both of the two cases, we assume there is diffusion phenomenon observed in the system, i.e., $\Delta\mathbf{u}$ will appear in the governing PDE. In addition, 9-point stencil (see Eq. (4) in A.1) is used to compute the diffusion terms while the Runge-Kutta method is adopted for time stepping. The loss function employed in the training and the evaluation metrics are detailed in A.4 and A.5.

3.2 RESULTS

To evaluate the performance of the proposed PeRCNN, we make a comparison of the prediction between the PeRCNN and some widely used spatiotemporal predictive models, e.g., ConvLSTM (Shi et al., 2015), recurrent ResNet (Liao & Poggio, 2016; Zhang et al., 2017) and the deep hidden physics model (DHPM) (Raissi, 2018). A brief introduction to each method and the range of hyperparameters considered are given in A.3.

3.2.1 2D BURGERS' DATASET

In this case, the training data includes 11 low-resolution (51×51) snapshots uniformly selected from the time interval of $[0, 0.1]$ after 10% Gaussian noise is added to the original dataset. Also, 2 snapshots are used as the hold-out validation dataset for hyperparameters selection and early stopping. Each model is constructed to produce the prediction with full spatiotemporal resolution, i.e., $\hat{\boldsymbol{\mathcal{U}}} \in \mathbb{R}^{2\times 401\times 101\times 101}$. After the model is finalized[2], 1200 extra prediction steps are performed to evaluate how the learned model generalizes beyond the training regime. Fig. 2 shows the snapshots predicted by each model at $t = 0.095$ and 0.395. As shown in Fig. 2(a), all the models are able to fit the training data and produce satisfactory prediction in the supervised time period. However, when it comes to long-term extrapolation, the model predictions deviate from the ground truth significantly except for PeRCNN, which demonstrates that PeRCNN generalizes the unknown underlying physics well from the data. This conclusion is further confirmed by Fig. 4(a) which depicts the accumulative RMSE (see Eq. (7)). We may notice that the accumulative RMSE starts from an initial high value. This is due to the fact that the training data is corrupted by 10% Gaussian noise and the metrics is computed from one single snapshot at the beginning. The effect of the unrelated noise gradually fades out as more time steps are considered.

3.2.2 3D GRAY-SCOTT RD DATASET

In this example, the training data includes 21 noisy snapshots on a 25^3 grid sampled from $t = 0$ to 150. Each trained model produces 301 full-resolution snapshots during supervised learning stage while 700 extrapolation steps are predicted after each model is finalized.

The predicted isosurfaces of two levels are plotted in Fig. 3. It can be observed that most of the models are able to produce reasonable prediction during the supervised time period. However, for the time beyond the training regime, the PeRCNN outperforms remaining models significantly. The flat error propagation curve of PeRCNN, as shown in Fig. 4(b), also demonstrates the remarkable generalization capability of PeRCNN.

As explained in Section 2.1, the architecture of PeRCNN enables a better approximation to the $\mathcal{F}$ of nonlinear PDEs, resulting in the good generalization capability of our model. To verify this claim, we exploit the multiplicative form of the Π-block and extract the explict expression from the learned model. We find the equivalent expression of the learned model is quite close to the genuine PDE. For more details on interpreting the learned PeRCNN model, please refer to Section A.6.

[2]All the implementations are coded in PyTorch or TensorFlow on a NVIDIA Tesla v100 GPU (32G).

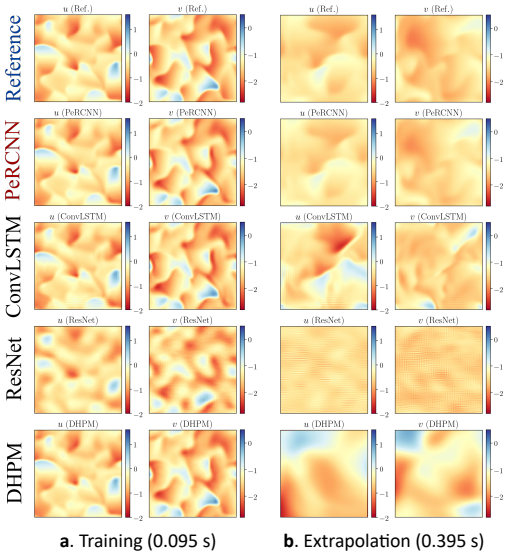


Figure 2: Contours for 2D Burgers' predictions.

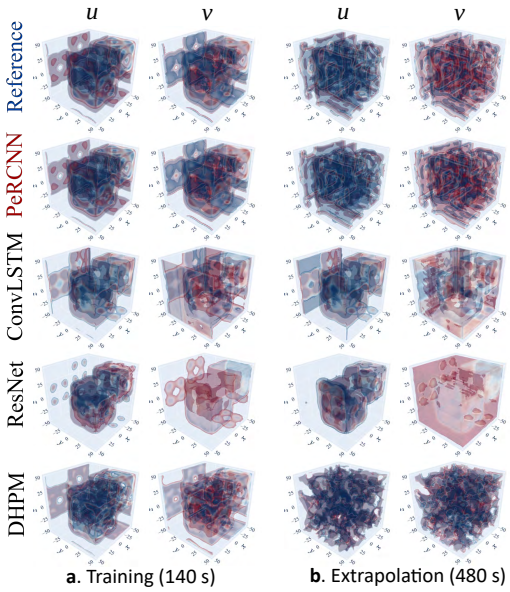
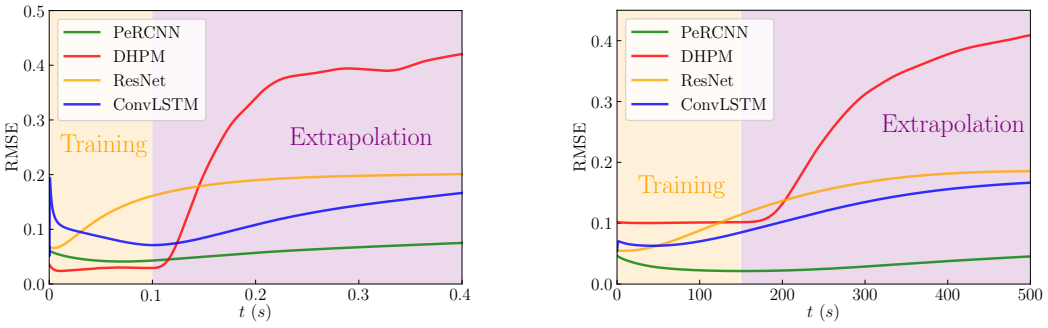


Figure 3: Isosurfaces for 3D GS-RD predictions. (u: blue = 0.5, red = 0.3; v: blue = 0.3, red = 0.1).



(a) 2D Burgers' dataset (b) 3D GS-RD dataset

Figure 4: Error propagation of the training and extrapolation prediction.

4 Conclusions

A novel DL architecture, called PeRCNN, is developed for modeling of nonlinear spatiotemporal dynamical systems based on sparse and noisy data. Our prior physics knowledge is forcibly encoded into PeRCNN which guarantees the resulting network strictly obeys given physics (e.g., I/BCs or known terms in PDEs). This brings distinct benefits for improving the convergence of training and accuracy of the model. To evaluate the generalizability of PeRCNN, the trained model is used for extrapolation along the temporal horizon. Comparisons with several state-of-the-art models demonstrate that physics-encoded learning paradigm uniquely possesses remarkable robustness against data noise/scarcity and generalizability. Equally important, PeRCNN shows good interpretability due to the multiplicative form of the recurrent block. As shown in Section A.6, an explicit expression can be extracted from the learned model via some symbolic computations.

Although PeRCNN shows promise in data-driven modeling 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 batching and multi-GPU training, to be investigated in our future studies.

REFERENCES

Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016.

Jian-Feng Cai, Bin Dong, Stanley Osher, and Zuowei Shen. Image restoration: total variation, wavelet frames, and beyond. *Journal of the American Mathematical Society*, 25(4):1033–1089, 2012.

Bo Chang, Lili Meng, Eldad Haber, Frederick Tung, and David Begert. Multi-level residual networks from dynamical systems view. *arXiv preprint arXiv:1710.10348*, 2017.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, volume 31, pp. 6572–6583, 2018.

Yunjin Chen, Wei Yu, and Thomas Pock. On learning optimized reaction diffusion processes for effective image restoration. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5261–5269, 2015.

Han Gao, Luning Sun, and Jian-Xun Wang. Phygeonet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state pdes on irregular domain. *Journal of Computational Physics*, 428:110079, 2021.

Nicholas Geneva and Nicholas Zabaras. Modeling the dynamics of pde systems with physics-constrained deep auto-regressive networks. *Journal of Computational Physics*, 403:109056, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Xiaodan Liang, Lisa Lee, Wei Dai, and Eric P Xing. Dual motion gan for future-flow embedded video prediction. In *proceedings of the IEEE international conference on computer vision*, pp. 1744–1752, 2017.

Qianli Liao and Tomaso Poggio. Bridging the gaps between residual learning, recurrent neural networks and visual cortex. *arXiv preprint arXiv:1604.03640*, 2016.

Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

Yun Long, Xueyuan She, and Saibal Mukhopadhyay. Hybridnet: integrating model-based and data-driven learning to predict evolution of dynamical systems. In *Conference on Robot Learning*, pp. 551–560. PMLR, 2018a.

Zichao Long, Yiping Lu, Xianzhong Ma, and Bin Dong. Pde-net: Learning pdes from data. In *International Conference on Machine Learning*, pp. 3208–3216. PMLR, 2018b.

Maziar Raissi. Deep hidden physics models: Deep learning of nonlinear partial differential equations. *The Journal of Machine Learning Research*, 19(1):932–955, 2018.

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019a.

Maziar Raissi, Zhicheng Wang, Michael S Triantafyllou, and George Em Karniadakis. Deep learning of vortex-induced vibrations. *Journal of Fluid Mechanics*, 861:119–137, 2019b.

Maziar Raissi, Alireza Yazdani, and George Em Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367(6481):1026–1030, 2020.

Rishikesh Ranade, Chris Hill, and Jay Pathak. DiscretizationNet: A machine-learning based solver for navier–stokes equations using finite volume discretization. *Computer Methods in Applied Mechanics and Engineering*, 378:113722, 2021.

Chengping Rao, Hao Sun, and Yang Liu. Physics-informed deep learning for incompressible laminar flows. *Theoretical and Applied Mechanics Letters*, 10(3):207–212, 2020.

Chengping Rao, Hao Sun, and Yang Liu. Physics informed deep learning for computational elastodynamics without labeled data. *Journal of Engineering Mechanics*, DOI: 10.1061/(ASCE)EM.1943-7889.0001947, 2021.

Lars Ruthotto and Eldad Haber. Deep neural networks motivated by partial differential equations. *Journal of Mathematical Imaging and Vision*, pp. 1–13, 2019.

Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *arXiv preprint arXiv:1506.04214*, 2015.

Xin Tao, Hongyun Gao, Renjie Liao, Jue Wang, and Jiaya Jia. Detail-revealing deep video super-resolution. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4472–4480, 2017.

Rui Wang, Karthik Kashinath, Mustafa Mustafa, Adrian Albert, and Rose Yu. Towards physics-informed deep learning for turbulent flow prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1457–1466, 2020a.

Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient pathologies in physics-informed neural networks. *arXiv preprint arXiv:2001.04536*, 2020b.

Zhuoning Yuan, Xun Zhou, and Tianbao Yang. Hetero-convlstm: A deep learning approach to traffic accident prediction on heterogeneous spatio-temporal data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 984–992, 2018.

Junbo Zhang, Yu Zheng, and Dekang Qi. Deep spatio-temporal residual networks for citywide crowd flows prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Yinhao Zhu, Nicholas Zabaras, Phaedon-Stelios Koutsourelakis, and Paris Perdikaris. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *Journal of Computational Physics*, 394:56–81, 2019.

A Appendices

A.1 Discrete Laplacian operator

The discrete Laplacian operator, defined by Eq. (4), is used in finite difference (FD) method to generate the synthetic dataset and in the PeRCNN to create physics-based convolutional layers.

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

A.2 Formulation of data-driven modeling

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

$$\mathbf{u}_t = \mathcal{F}\left(\mathbf{x}, t, \mathbf{u}, \mathbf{u}^2, \nabla_{\mathbf{x}}\mathbf{u}, \mathbf{u}\cdot\nabla_{\mathbf{x}}\mathbf{u}, \nabla^2\mathbf{u}, \cdots\right) \tag{5}$$

where the state variable/solution $\mathbf{u}(\mathbf{x}, t) \in \mathbb{R}^s$ (e.g., $\mathbf{u} = [u,\ v]^{\mathrm{T}}$ for $s = 2$) is defined over the spatiotemporal temporal domain $\{\mathbf{x}, t\} \in \Omega \times [0, T]$; $\nabla_{\mathbf{x}}$ is the Nabla operator with respect to $\mathbf{x}$; and $\mathcal{F}(\cdot)$ is a nonlinear function. The solution to this problem is subject to the initial condition (IC)

$\mathcal{I}(\mathbf{u}, \mathbf{u}_t; t = 0, \mathbf{x} \in \Omega) = 0$ and boundary condition (BC) $\mathcal{B}(\mathbf{u}, \nabla_{\mathbf{x}}\mathbf{u}, \cdots; \mathbf{x} \in \partial\Omega) = 0$, where $\partial\Omega$ denotes the boundary of the system. In this work, we mainly focus on regular physical domains, i.e., $\mathbf{u}$ can be discretized on a $H \times W$ Cartesian grid at time steps $\{t_1, ..., t_k, ..., t_{n_t}\}$, where n_t denotes the total number of time steps. Provided a scarce and potentially noisy set of measurements $\tilde{\mathbf{u}} \in \mathbb{R}^{2\times n_t' \times H' \times W'}$ over a coarser spatiotemporal grid, the goal of the data-driven modeling is to establish a reliable model that gives the most likely full-field solution $\hat{\boldsymbol{\mathcal{U}}} \in \mathbb{R}^{2\times n_t \times H \times W}$ and possesses satisfactory extrapolation ability over the temporal horizon (e.g., for $t > t_{n_t}$).

A.3 Baselines for comparisons

To make a comparison of the PeRCNN with other models used widely for data-driven modeling of spatiotemporal systems, we also implemented the recurrent ResNet (Liao & Poggio, 2016; Zhang et al., 2017), convolutional long-short term memory (ConvLSTM) (Shi et al., 2015) and deep hidden physics model (DHPM) (Raissi, 2018). A very brief introduction to each method is given below for readers to grasp the major characteristics of each model.

ConvLSTM (Shi et al., 2015) is a convolutional variant of LSTM which exploits multiple self-parameterized controlling gates, such as input, forget and output gates, to capture the spatiotemporal correlations among the data. It has been extensively used in applications such as video super-resolution (Tao et al., 2017; Liang et al., 2017), traffic prediction (Yuan et al., 2018) and climate forecasting (Shi et al., 2015), among many others.

Recurrent ResNet is another model adopted widely by researchers (Liao & Poggio, 2016; Zhang et al., 2017) for the spatiotemporal prediction of dynamical systems. One main characteristic distinguishes the recurrent ResNet with the conventional ResNet (He et al., 2016) is that the weights are shared across time.

DHPM (Raissi, 2018) differs from the previous two models as it utilizes fully connected neural networks (FCNNs) and exert hidden (unknown) physics prior on the solution. In DHPM, one deep FCNN is employed to fit the data, i.e., pairs of the spatiotemporal location and solution, while another shallow FCNN is used to impose a hidden physical constraint on the fitted solution.

Since it is impossible to keep the hyperparameters exactly the same among different models, we select the best configuration for each model from a range of hyperparameters, which are summarized in Table 1, 2, 3 and 4. In addition to the listed hyperparameters, all other hyperparameters are kept the same, e.g., training/validation/testing dataset split, the number of prediction steps, the optimizer (Adam), the max number of epochs, the Gaussian noise level (10%) and the random seed. In the network architecture design, we assume the solution within the domain is periodic while the dynamical system of interest is accompanied with the ubiquitous diffusion phenomenon. Hence, a diffusion Conv layer with fixed filters will be created in the following PeRCNN models.

Table 1: Range of hyperparameters for PeRCNN.

Dataset	Filter size	# layers (Π-block)	# channels (ISG)	# channels	Learning Rate	λ
2D BE	1∼5 (5)	2∼4 (4)	4∼16 (8)	4∼16 (8)	0.001∼0.01 (0.002)	0.001∼1 (1)
3D GS	1∼5 (1)	2∼4 (3)	2∼8 (4)	4∼8 (4)	0.001∼0.01 (0.005)	0.001∼1 (0.5)

Table 2: Range of hyperparameters for ConvLSTM.

Dataset	Filter size	# layers	# channels	Learning rate	Weight decay
2D BE	3∼5 (5)	1∼2 (2)	16∼32 (32)	0.0005∼0.01 (0.001)	e-5∼e-3 (e-5)
3D GS	3∼5 (5)	1	8∼16 (16)	0.0005∼0.01 (0.0005)	e-5∼e-3 (e-5)

Table 3: Range of hyperparameters for recurrent ResNet.

Dataset	Filter size	# layers	# channels	Learning rate	Weight decay
2D BE	3∼5 (3)	2∼4 (2)	16∼128 (64)	0.0005∼0.01 (0.002)	e-5∼e-2 (e-4)
3D GS	3∼5 (3)	2∼3 (2)	8∼32 (32)	0.0005∼0.01 (0.001)	e-5∼e-2 (e-4)

Table 4: Range of hyperparameters for DHPM.

Dataset	$\mathcal{N}_1$ width	$\mathcal{N}_1$ depth	$\mathcal{N}_2$ width	$\mathcal{N}_2$ depth	Input for $\mathcal{N}_2$	Learning rate
2D BE	80∼120 (120)	4∼5 (5)	10∼30 (20)	2∼3 (2)	$(\Delta u, \Delta v, uu_x, vu_y, uv_x, vv_y)$	0.001 ∼0.02 (0.005)
3D GS	60∼100 (80)	4∼5 (5)	10∼30 (10)	2∼3 (2)	$(\Delta u, \Delta v, u, v)$	0.001 ∼0.02 (0.01)

A.4 Loss function

Given the low resolution measurement[3] $\tilde{\mathbf{u}} \in \mathbb{R}^{2\times n_t' \times H' \times W'}$ where $n_t' < n_t$, $H' < H$ and $W' < W$, our goal of the data-driven modeling is to reconstruct the most likely full-field solution $\hat{\boldsymbol{\mathcal{U}}} \in \mathbb{R}^{2\times n_t \times H \times W}$. The loss function to train PeRCNN is defined as:

$$\mathcal{L}(\mathbf{W}, \mathbf{b}) = \text{MSE}\left(\hat{\boldsymbol{\mathcal{U}}}(\tilde{\mathbf{x}}) - \tilde{\mathbf{u}}\right) + \lambda \cdot \text{MSE}\left(\hat{\boldsymbol{\mathcal{U}}}_0 - \mathcal{P}(\tilde{\mathbf{u}}_0)\right) \tag{6}$$

where $\hat{\boldsymbol{\mathcal{U}}}(\tilde{\mathbf{x}})$ denotes the network's prediction at the coarse grid nodes $\tilde{\mathbf{x}}$; $\tilde{\mathbf{u}}$ denotes the low-resolution measurement; $\mathcal{P}(\cdot)$ is a spatial interpolation function (e.g., bicubic or bilinear); λ is the weighting coefficient for the regularizer. The regularization term denotes the IC discrepancy between the interpolated initial state and the network's prediction, which is found effective in preventing network overfitting.

A.5 Evaluation metrics

Accumulative rooted-mean-square error (RMSE), defined by Eq. (7), is employed to compute the error of all snapshots before a time step t_k. It is used throughout this paper to evaluate the error propagation of the model prediction.

$$\text{RMSE}(t_k) = \sqrt{\frac{1}{nk}\sum_{i=1}^{k}\left\|\hat{\boldsymbol{\mathcal{U}}}_i - \boldsymbol{\mathcal{U}}_i^{\text{ref}}\right\|_2^2} \tag{7}$$

where $\boldsymbol{\mathcal{U}}_i^{\text{ref}}$ is the reference solution and $k \in \{1, 2, \cdots n_t\}$.

A.6 Interpret the learned model

Since each channel of the input ($\hat{\boldsymbol{\mathcal{U}}}_k$) denotes a state variable component (i.e., $[u, v]$), the multiplicative form of Π-block makes it possible to extract (or interpret) an explicit expression for $\mathcal{F}$, in a symbolic way, from the learned weights and biases. We first interpret the learned model from 3D GS-RD case, whose parallel Conv layers have filter size of 1. In this case, each output channel from the Conv layer would be the linear combination of u, v and a constant. The identified diffusion coefficient matrix from the physics-based Conv layer is $\mathbf{D} = \text{diag}(0.18, 0.080)$. The extracted expression from the learned Π-block is

$$\mathbf{R}(\mathbf{u}) = \begin{bmatrix} -0.0074u^3 - 0.0051u^2v - 0.2uv^2 - 0.0386v^3 - 0.0018u^2 - 0.11uv - 0.055v^2 - 0.016u - 0.022v + 0.025 \\ 0.0005u^3 - 0.013u^2v + 0.54uv^2 - 0.087v^3 - 0.0076u^2 + 0.023uv + 0.046v^2 + 0.017u - 0.036v - 0.0097 \end{bmatrix} \tag{8}$$

which includes some distracting terms with small coefficients, due to the 10% noise and scarcity of the training data.

To interpret terms involving partial derivatives (e.g., $u\nabla^2 u$, uu_x), it would require us to completely freeze or impose moment matrix constraints on part of the convolutional filters (Long et al., 2018b).

[3] It can be readily generalized to 3D cases.

Here, a simple experiment on 2D Burgers' equation is conducted. The network employed has two Conv layers with two channels. The first Conv layer are associated with ∂x and ∂y respectively, by fixing the filters with corresponding FD stencils. The remaining settings are kept the same as in Section 3.2.1 except that noise-free training data is used. The interpreted expression from the whole PeRCNN model is

$$\mathbf{u}_t = \begin{bmatrix} 0.0051\Delta u - 0.95u_x(1.07u - 0.0065v - 0.17) + 0.98u_y(0.0045u - 1.01v + 0.17) + 0.053 \\ 0.0051\Delta v - 0.82v_x(1.22u + 0.0078v - 0.18) - 0.91v_y(0.0063u + 1.08v - 0.17) + 0.058 \end{bmatrix} \tag{9}$$

which is very close to the ground truth of the governing PDE. Although the selection of candidate differential operators is crucial for identifying the genuine PDE, the Π-block shows better interpretability compared with the prolonged nested function formed by the FCNN or CNN.