ST-PCNN: Spatio-Temporal Physics-Coupled Neural Networks for Dynamics Forecasting

Yu Huang

Florida Atlantic University

James Li

Georgia Institute of Technology

Min Shi

Washington University School of Medicine

Hanqi Zhuang (✉ zhuang@fau.edu)

Florida Atlantic University

Yufei Tang

Florida Atlantic University

Xingquan Zhu

Florida Atlantic University

James VanZwieten

Florida Atlantic University

Laurent Chérubin

Florida Atlantic University

Research Article

Keywords: Spatio-Temporal Physics-Coupled Neural Networks, Dynamics Forecasting, Ocean current, fluid mechanics

Posted Date: October 26th, 2021

DOI: <https://doi.org/10.21203/rs.3.rs-966026/v1>

License: This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

ST-PCNN: Spatio-Temporal Physics-Coupled Neural Networks for Dynamics Forecasting

Yu Huang¹, James Li², Min Shi³, Hanqi Zhuang^{1,*}, Yufei Tang¹, Xingquan Zhu¹, James VanZwieten¹, and Laurent Chérubin⁴

¹Florida Atlantic University, Department of Electrical Engineering and Computer Science, Boca Raton, FL 33431, USA

²Georgia Institute of Technology, Daniel Guggenheim School of Aerospace Engineering, Atlanta, GA 30332, USA

³Washington University School of Medicine, Department of Genetics, St. Louis, MO 63110, USA.

⁴Florida Atlantic University, Harbor Branch Oceanographic Institute, Fort Pierce, FL 34946, USA.

*zhuang@fau.edu

ABSTRACT

Ocean current, fluid mechanics, and many other physical systems with spatio-temporal dynamics are essential components of the universe. One key characteristic of such systems is that they can be represented by certain physics laws, such as ordinary/partial differential equations (ODEs/PDEs), irrespective of time or location. Physics-informed machine learning has recently emerged to learn physics from data for accurate prediction, but they often lack a mechanism to leverage localized spatial and temporal correlation or rely on hard-coded physics parameters. In this paper, we advocate a physics-coupled neural network model to learn parameters governing the physics of the system, and further couple the learned physics to assist the learning of recurring dynamics. Here a spatio-temporal physics-coupled neural network (ST-PCNN) model is proposed to achieve three goals: (1) learning the underlying physics parameters, (2) transition of local information between spatio-temporal regions, and (3) forecasting future values for the dynamical system. The physics-coupled learning ensures that the proposed model can be tremendously improved by using learned physics parameters, and can achieve useful long-range forecasting (e.g., more than two weeks). Experiments using simulated wave propagation and field-collected ocean current data validate that ST-PCNN outperforms typical deep learning models and existing physics-informed models.

Introduction

Spatio-temporal modeling is essential in many scientific fields ranging from studies in biology^{1,2}, information flow in social networks³, sensor network communications⁴, traffic predictions^{5,6}, climate and environment forecasting^{7,8}, to recent COVID-19 spread modeling⁹. These applications rely on accurate predictions of spatio-temporal structured data reflecting the real-world phenomena. In all mentioned cases, the major challenge is to infer, model, and predict the underlying causes, which generate the perceived data stream and propagate the involved causal dynamics through graphs and distributed sensor meshes. A stunning characteristic of such dynamical systems is that the widely distributed members (or sensors) share striking *homogeneity* and *heterogeneity*. The former is driven by the physics laws governing the systems, whereas the latter is impacted by the localized factors in spatial and temporal regions.

Take information propagation mechanisms of ocean current in Figure 1 as an example, where each node denotes a geographic location of an ocean current measurement. Three types of dependencies exist in spatio-temporal modeling: 1) Spatial dependence: a node in the mesh concurrently affects, and be affected by, its neighbors; 2) Temporal dependence: a node status depends on its previous status and affects its future status; and 3) Spatio-temporal dependence: a node directly influences its neighbor nodes across the time.

Deep learning methods, such as graph neural networks (GNNs), have been applied to spatio-temporal modeling. Existing methods take temporal information into account – e.g. ARIMA¹⁰, or integrate complex spatial dependencies into temporal models – e.g. ConvLSTM¹¹ and ST-3DNet¹². Most recently, researchers utilize graph convolution methods, such as DCRNN¹³ and STGCN⁶, to model spatial correlations in spatio-temporal structured data. Instead of modeling the spatio and temporal correlations separately, STSGCM¹⁴ and STG2Seq¹⁵ try to simultaneously capture the localized spatio-temporal correlations. It is worth mentioning that STSGCM deploys multiple modules at each time period to capture the heterogeneity, which is computationally intensive.

One major issue of existing deep learning methods is that they seldom include prior knowledge of the underlying physics, i.e. taking the “homogeneity” into consideration. Physics is one of the fundamental pillars describing how the real-world

Figure 1. Three types of dependencies in spatio-temporal modeling. A node is mixed with heterogeneity (orange) and homogeneity (blue) information, propagating between neighbors. At any time point t , the status of a center node is influenced by its previous time point ($t - 1$) and its neighbors (red-arrows with various levels of strengths/weights).

behaves. A key property that all spatio-temporal processes have in common, is that some generally underlying principles will apply irrespective of time or location when observing natural processes. As a result, the same predictable patterns individually modified by local spatial and temporal influences are repeatedly observable at different spatial locations in time. Recently, physics informed learning has emerged to incorporate physics into the deep learning^{16–18}. Physics-informed learning can directly solve differential equations with neural nets given space x and time t as input variables^{19–21}. PDE-Net²² and ODE-Net²³ have been proposed to train neural networks that simultaneously approximate the simulations and conform to the PDEs representing the physical knowledge of systems. However, these methods suffer two major shortcomings: (1) most, if not all of them¹⁷, are often restricted to 1D temporal sequence or to a regular grid where constraints on the learnable filters can be easily defined; and (2) there is no good solution to combine homogeneity and heterogeneity for effective prediction. In summary, three research challenges are identified as follows:

- **Learning physics:** While physics informed learning has been proposed^{16–18}, they are only applicable when physics equations are explicitly given and none of them consider the spatio-temporal cases. In reality, it is not always possible to describe all rules governing real-world data. Physics-aware difference graph networks have been proposed recently⁷, however, additional edge features are required for graph structured data. Instead, we need to have a learning mechanism to automatically discover the underlying physics of the spatio-temporal data observations.
- **Coupling physics and spatio-temporal information:** Homogeneity and heterogeneity are two key characteristics of dynamical systems but are governed by different modules. Simultaneously learning the spatial heterogeneity and temporal homogeneity is not trivial. We need to have a new way to enable the learning of physics and use the inferred physics to further guide the spatio-temporal learning with robust prediction.
- **Long-term forecasting:** For dynamical systems, long-term forecasting allows proactive controls and early planning. Existing models that focused on one-step-ahead short-term forecasting have been proved to be successful^{24,25}, and an intuitive way to achieve long-term prediction is to recursively reuse previous-step predictions as input for the next-step prediction. Inevitably, such a mechanism leads to prediction errors that can accumulate over time, which will decrease the prediction skill.

In this paper, we propose a spatio-temporal physics-coupled neural networks (ST-PCNN) model to capture spatio-temporal correlations, include heterogeneity and homogeneity in a spatially distributed manner. ST-PCNN is a three-network architecture, consisting of a forecasting net (FN), a transition net (TN), and a physics net (PN). ST-PCNN learns predictive neural network (FNs) that are distributively executed at different locations of a grid. Additional information routing through transition neural network (TNs) laterally connect the FNs. Both FNs and TNs share their weights respectively, allowing efficient parallel computation and capturing heterogeneity from all spatial locations. To incorporate physics laws, which will yield an effective model that uses less samples in the training stage and is robust to unseen data, a third network PN is developed to reveal unknown governing physics from pre-given spatio-temporal data and vice versa facilitates the overall model to capture the homogeneity.

The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 defines the spatio-temporal forecasting problem for the dynamical systems from a machine-learning perspective. The proposed model ST-PCNN is introduced in Section 4, followed by experiments in Section 5 that include comparative and ablation studies. Conclusions are given in Section 6.

Related Work

Physical process modeling is close to the field of spatio-temporal statistical modeling that is increasingly being used across a wide variety of scientific disciplines (e.g., ocean dynamics²⁶, Covid-19²⁷, etc.) to describe and predict spatially explicit processes that evolve over time²⁸. Cressie et al.²⁹ advocate the use of physical prior knowledge to develop statistical models, e.g., PDEs related to the observed real-world phenomenon. They mainly consider auto regressive models within a hierarchical Bayesian framework. Another research direction is the use of neural networks (NNs) for enhancing the performance and reducing the complexity of numerical physical process simulation. There are three major approaches: 1) NNs are used in place of a computationally demanding component of the simulation process. For examples, Ladicky et al.³⁰ used a random forest to compute particle location and Tompson et al.³¹ adopted a CNN to approximate part of a numerical PDE scheme. 2) NNs are combined with related physics equations to model the whole physical process. For example, Raissi et al.¹⁹ uses physics-informed neural networks (PINNs) to learn nonlinear relations between spatial- and temporal-coordinates with a given PDE. The learned physics can be applied for various processes, such as informed advection-diffusion equation³², fluid dynamics⁷, Lagrangian mechanics³³, and Hamiltonian SymODEN³⁴, etc. These methods are only applicable when the specific equations are explicitly given and are hard to generalized to incorporate other types of physics equations. 3) NNs are used to uncover the underlying hidden physics and model the dynamics of complex systems. Early work can be found in Gonzalez et al.³⁵ which used a Runge-Kutta integrator for the identification of 1-D PDEs. State-of-the-art work includes discovering the PDEs^{16,22,36} or ODEs²³ from given observations of the systems. Christopher et al.³⁷ proposed universal differential equations that could be utilized to discover previously unknown governing equations, accurately extrapolate beyond the original data, and accelerate model simulation, all in a time and data-efficient manner.

Problem Definition

A dynamical system is observed from a grid of nodes (e.g., a sensor network), distributed/located at different locations. $s^{(t,i,j)} \in \mathbb{R}$ denotes observed value of a node located at (i, j) at time t . For ease of representation we use $s^{(t,:)}$ to denote value of any node at time t , and $s^{(:,i,j)}$ denotes values observed at (i, j) . Accordingly, each time slice of the observational grid is denoted by $S^{(t,:)} = \{s^{(t,i,j)}\}_{i,j \in \Omega} \in \mathbb{R}^\Omega$ and the time series of each node is defined by $S^{(:,i,j)} = \{s^{(t,i,j)}\}_{t \in T} \in \mathbb{R}^T$ (Ω is the total number of nodes and T is the total number of recorded time steps). $S \in \mathbb{R}^{\Omega \times T}$ denotes the whole observations and $\hat{S}$ represents the variable flow in the future.

Assume a variable observed from a node is governed by an unknown physics rule, i.e., PDE/ODE (ordinary differential equations form a subclass of partial differential equations, corresponding to functions of a single variable), which relates a function $u(\mathbf{x}, t)$ with its derivatives, i.e., $D^a u(\mathbf{x}, t) := \frac{\partial^{a_1}}{\partial x_1^{a_1}} \cdots \frac{\partial^{a_k}}{\partial x_k^{a_k}} u(\mathbf{x}, t)$, where $a = (a_1, \dots, a_k)$ are non-negative integers. Consequently, any PDE can be defined by:

$$\mathcal{G}(\mathbf{x}, t, u(\mathbf{x}, t), D^a u(\mathbf{x}, t)) = 0 \quad (1)$$

where $\mathcal{G}$ is a function that relates position $\mathbf{x} = (i, j) \in \Omega$ and time $t \in \mathbb{R}$ with u and its partial derivatives at $\mathbf{x}$ and t . We say u is a solution to the PDE if Eq. (1) holds for every point $\mathbf{x} \in \Omega$ and time step $t \in \mathbb{R}$. In this paper, we observe data points $(\mathbf{t}, \mathbf{x}; S) = \{t, i, j; s^{(t,i,j)}\}_{i,j \in \Omega}$, where S are the fusion values at $\mathbf{x}$, that is, $s^{(t,i,j)} = u(t, i, j) + \xi^{(t,i,j)}$. $u(t, i, j)$ refers to homogeneity while $\xi^{(t,i,j)}$ is the cause of heterogeneity in dynamical systems. $\xi^{(t,i,j)}$ is impacted by the temporal localized factors. The temporal localized factors can be described by the unknown temporary processes that happen in a specific location in a real-world case, or spatially/temporally dependent forcing or boundaries in a simulation case, or by non-stationary process noise.

Spatio-Temporal Forecasting Problem: From a machine-learning perspective, the spatio-temporal forecasting problem is to learn a non-linear mapping function f that maps the historical spatio-temporal observations $\{S^{t_p-\tau}, S^{t_p-\tau+1}, \dots, S^{t_p}\}$ into the future predictions $\{\hat{S}^{t_p+1}, \hat{S}^{t_p+2}, \dots, \hat{S}^{t_p+\tau'}\}$, where τ denotes the length of observation conditioned on and τ' denotes the prediction horizon. The learning is formulated as a deterministic optimization problem that constitutes both minimizing the data mismatches and estimating the hidden underlying PDE of a physical model by equating derivatives of the neural network approximation.

The Proposed Framework: ST-PCNN

To better model dynamical systems in terms that consider both homogeneity and heterogeneity components, a spatio-temporal physics-coupled neural networks (ST-PCNN) model is proposed. ST-PCNN is a tri-network architecture, as shown in Figure 2, consisting of a physics network (PN), a forecasting network (FN), and a transition network (TN). FN receives 1) dynamic data, which evolution will be predicted and change over time, 2) static information, which stays constant and characterizes the location of each FN, and 3) lateral information from neighbors. The output of each FN includes predicted dynamics and

Figure 2. (a): The lateral connection schema of FN, TN and PN. Green node denote a center node located at (i, j) , and faded green nodes are its neighbors. Three types of information are used to characterize each node: (1) Dynamics $S^{(t,i,j)}$: an embedding vector that represents status of node at time t , (2) Static Info $\mathbf{p}^{(i,j)}$: an embedding vector that represents node location; and (3) Lateral Info $\mathbf{L}^{(t,i,j)}$: an embedding vector (dashed dot-square set) capturing interaction (lateral info) between each node and its neighbors; (b): The coupling mechanism of FN and PN in sequential perspective.

additional lateral information that will be interacted with its neighbors. Such interaction, that distinguishes our architecture from others, is conducted through a TN with two-stacked linear layers. TN aims at modeling the location-sensitive transitions between adjacent FNs and thus enabling local context-dependent spatial information propagation.

Spatio-Temporal Heterogeneity

In many natural phenomena, data are collected in a distributed manner and exhibit heterogeneous properties: each of these distributed locations present a different view of the natural process at the same time, where each view has its own individual representation of the space and dynamics³⁸. Theoretically, each location may contain information that other location do not have access to. Therefore, all local views must be interacted in some way in order to describe the global activity comprehensively and accurately.

To enable the model to leverage localization information, we must inject some information about the relative or absolute position of each data point. How to explicitly encode location information into neural networks is critical in this location-wise forecasting. Inspired by the Transformer model³⁹ that encodes word positions in sentences, we extend the absolute positional encoding to represent grid positions. In particular, let i, j be the desired position in a regular grid, $\mathbf{p}^{(i,j)} \in \mathbb{R}^{2D}$ be its corresponding encoding, D be the encoding dimension, and $d = [1, \dots, D]$ be the element index in the encoded vector. Then, the encoding scheme $\mathcal{E}$ is defined as:

$$\mathbf{p}^{(i,j)} = [\mathcal{E}(i), \mathcal{E}(j)] \in \mathbb{R}^{2D}; \text{ where } \mathcal{E}(*)^d = \begin{cases} \sin(* \cdot \omega_k) & \text{if } d = 2k \\ \cos(* \cdot \omega_k) & \text{if } d = 2k + 1 \end{cases} \quad (2)$$

where $\omega_k = \frac{1}{10,000^{2k/D}}$, $k \in \mathbb{N}_{\leq \lceil \frac{D}{2} \rceil}$. The wavelengths form a geometric progression from 2π to $10000 \cdot 2\pi$. Because the positional embedding is a vector that contains pairs of *sines* and *cosines* for each decreasing frequency along the vector dimension, it allows the model to easily learn the relative positions of the grid nodes³⁹.

As illustrated in Figure 2, the FN and TN are executed in space simultaneously. At each time t , the TN first encodes the current operation node's lateral info $\mathbf{L}$ and static info $\mathbf{p}$ as follows:

$$\mathbf{L}_{enc}^{(t,i,j)} = \text{Relu}([\mathbf{p}^{(i,j)}, \mathbf{L}^{(t,i,j)}] \mathbf{W}_T^T + b_T) \quad (3)$$

where $\theta_T = [\mathbf{W}_T, b_T]$ denote the weights and bias of TN. $\mathbf{L}^{(t,i,j)}$ is a vector used to characterize interaction between a node at i, j and its neighbors. $\mathbf{L}$ is only aggregated over the nearest neighbours. It is initialized as zero and continuously updated by Eq. (8) when $t > 0$.

Then, FN encodes each view (i.e., static $\mathbf{p}$, dynamics S , and encoded $\mathbf{L}_{enc}$ of each node) using a fusion layer:

$$f^{(t,i,j)} = [\mathbf{p}^{(i,j)}, S^{(t,i,j)}, \mathbf{L}_{enc}^{(t,i,j)}] \mathbf{W}_{fusion}^T + b_{fusion} \quad (4)$$

These features, $f^{(t,i,j)} \in \mathbb{R}^{d_{f_{i,j}}}$, are then fed into an LSTM to model the node-specific interactions over time. The update mechanism of the LSTM cell is defined as:

$$[I^{(t)}; F^{(t)}; \tilde{C}^{(t)}; O^{(t)}] = \sigma(W \cdot f^{(t,i,j)} + T \cdot h^{(t-1)}) \quad (5)$$

$$C^{(t)} = \tilde{C}^{(t)} \circ I^{(t)} \quad (6)$$

$$h^{(t)} = O^{(t)} \circ C^{(t)} \quad (7)$$

where $\sigma(\cdot)$ applies sigmoid on the input gate $I^{(t)}$, forget gate $F^{(t)}$, and output gate $O^{(t)}$, and $\tanh(\cdot)$ on memory cell $\tilde{C}^{(t)}$. The parameters are characterized by $W \in \mathbb{R}^{d_{fi,j} \times d_{hi,j}}$ and $T \in \mathbb{R}^{d_{hi,j} \times d_{hi,j}}$, where $d_{hi,j}$ is the output dimension. A cell updates its hidden states $h^{(t)}$ based on the previous step $h^{(t-1)}$ and the current input $f^{(t,i,j)}$.

An output layer is stacked at the end of FN to transform the LSTM output into the expected dynamic prediction and additional lateral information as:

$$\left[\hat{S}^{(t,i,j)}; \hat{\mathbf{L}}^{(t,i,j)} \right] = \text{Relu}(W_{out} \cdot f^{(t,i,j)} + b_{out}) \quad (8)$$

where $\hat{S}^{(t,i,j)}$ denotes the prediction of the node dynamics at time step t . The learnable parameters are characterized by $W^{(t)} \in \mathbb{R}^{d_{fi,j} \times d_{yi,j}}$ and $b_{out} \in \mathbb{R}^{d_{yi,j}}$, where $d_{yi,j}$ denotes the total dimension of the dynamic and the lateral outputs.

Homogeneity by Underlying Physics

Physicists attempt to model natural phenomena in a principled way through analytic descriptions. Conservation laws, physical principles, or phenomenological behaviors are generally formalized using differential equations, which can best represent the homogeneity of observations. Knowledge accumulated for modeling physical processes in well developed fields such as maths or physics could provide a useful guideline for dynamics learning³². Our proposed ST-PCNN includes a physics-aware module, the physics network PN, to learn underlying hidden physics. Two main approaches are considered:

- **PDE-learning-net**³⁶: explicitly estimating the underlying partial differential equation (PDE) from time series assisted by a neural network model fitting to PDE solution.
- **ODE-informed-net**²³: implicitly approximating time-dependence with a neural network and solving via an ordinary differential equation (ODE) solver.

PDE-learning-net

A PDE is an equation which imposes relations between the various partial derivatives of a multivariable function. PDEs are ubiquitous in mathematically-oriented scientific fields, such as physics and engineering. For instance, they are the foundation in the modern scientific understanding of sound, heat, diffusion, fluid dynamics, general relativity, quantum mechanics, etc. PDEs for a single variable $\mathbf{v}$ can be generally described as a linear combination of functions of u , its derivatives, and the dimensions. More formally, letting $D = \{D_1, \dots, D_K\}$ be a dictionary of such terms, we can then generally define PDEs as:

$$\sum_{k=1}^K c_k D_k(\mathbf{v}, u(\mathbf{v}), D^a u(\mathbf{v})) = 0, \forall \mathbf{v} \in \Omega_v \quad (9)$$

where $\mathbf{c} = \{c_k\}_{k=1}^K$ is a set of coefficients to be determined. These terms are determined by the best estimate of what would be relevant for each use case. For most physical systems, this is typically limited to second order derivatives. Suppose the 1D wave equation is $au_{tt} - bu_{xx} = \mathcal{G}(u_{tt}, u_{xx}) = 0, \forall \mathbf{x} \in \Omega \in \mathbb{R}^2, \forall t \in \mathbb{R}$. If we define our dictionary as $D_1 = u_{tt}, D_2 = u_{xx}$, where $aD_1(u, x, t) - bD_2(u, x, t) = 0$, the PDE can then be represented by the coefficients $\mathbf{c} = [a, -b]$.

Suppose we observe $u(\mathbf{x}, t)$ such that u is a solution to Eq. (9). It is then possible to approximate u by a neural network $\hat{u} : \Omega \in \mathbb{R}$, where observations at $(\mathbf{x}, t)$ becomes the training inputs. To approximate u , the PDE-learning-net³⁶, as illustrated in the dotted box of Figure 3, consists of a four-linear-layer stacked, fully-connected network with linear input and output layers as denoted below:

$$\hat{u}(\mathbf{x}, t) = \tanh([\mathbf{x}, t] W_1^T + b_1) \cdots W_n^T + b_n \quad (10)$$

where the inputs are position $\mathbf{x}$ and time t , and yields the prediction value $\hat{u}$ at that position. The $\tanh(\cdot)$ activation function is employed after each layer except for the output layer.

This approximation is optimized by a combination of multiple loss terms to place emphasis on different parts of the model, defined as:

$$\mathcal{L}(\mathbf{x}, \mathbf{x}'; \mathbf{c}, \theta) = \mathcal{L}_u^{1/2} (1 + \lambda_d \mathcal{L}_d + \lambda_{sparse} \mathcal{L}_{sparse}) \quad (11)$$

Figure 3. Diagram of the PDE-learning-net structure and loss calculations.

where the use of $\mathcal{L}_u$ as a scaling factor for the other losses acts as a regularizing term to maintain a more consistent ratio of losses even when the mean square error (MSE) becomes very large or small. The λ_d and λ_{sparse} coefficients are pre-set such that $\mathcal{L}_d$ and $\mathcal{L}_{sparse}$ will have similar magnitude. The definition and contribution of loss terms are explained below³⁶.

$\mathcal{L}_u$ is the loss between observed data points u and values calculated by the neural network at these points $\hat{u}$:

$$\mathcal{L}_u(\mathbf{x}; \theta) = \frac{1}{N} \sum_{i=1}^N (u_i - \hat{u}(\mathbf{x}_i; \theta))^2 \quad (12)$$

where θ is the neural network parameters. This is the primary regression loss term for the neural network. A large value in $\mathcal{L}_u$ indicates a large noise variance. In response, the contribution of the other terms increases in order to smooth $\hat{u}$ and prevent it from overfitting to noise. When $\mathcal{L}_u$ is small, it indicates a small noise variance.

$\mathcal{L}_d$ is a differentiation loss used to measure the error of the estimated PDE coefficients. Here, a dictionary of L differential terms $D = \{D_1, D_2, \dots, D_L\}$ is used. These functions are evaluated at K points $\mathbf{x}' = \{\mathbf{x}_i\}_{i=1}^K$, sampled from Ω , resulting in a $\mathbb{R}^{K \times L}$ matrix with entries:

$$D(\hat{u}, \mathbf{x}', \theta)_{k,l} := D_l(\mathbf{x}'_k, \hat{u}(\mathbf{x}'_k; \theta), D^\alpha \hat{u}(\mathbf{x}'_k)) \quad (13)$$

By definition of the PDE, we require $D(u, \mathbf{x}')\mathbf{c} = 0$, thus $\mathbf{c}$ must lie within the null space of $D(u, \mathbf{x}')$. Equivalently, $\mathbf{c}$ is a singular vector of $D(u, \mathbf{x}')$, with associated singular value 0. If $\|\hat{u} - u\|_{m,2} < \varepsilon$ and assuming some regularity conditions on D , we can have $\|D(u, \mathbf{x}') - D(\hat{u}, \mathbf{x}')\|_{m,2} < \varepsilon C$ for some constant C that depends on D . Therefore, the singular vector of $D(\hat{u}, \mathbf{x}')$, associated with its smallest singular value, is an approximation of $\mathbf{c}$.

The loss term $\mathcal{L}_d$ is then defined along with the constraint $\|\mathbf{c}\| = 1$ to avoid $\mathbf{c}$ being minimized to zero:

$$\mathcal{L}_d(\mathbf{x}'; \theta, \mathbf{c}) = \|D(\hat{u}(\mathbf{x}'; \theta), \mathbf{x}')\mathbf{c}\|_2^2 \quad (14)$$

The contribution of $\mathcal{L}_d$ is twofold: minimizing $\mathcal{L}_d(\mathbf{x}', \theta, \mathbf{c})$ over $\mathbf{c}$ recovers the PDE and minimizing $\mathcal{L}_d(\mathbf{x}', \theta, \mathbf{c})$ over $\hat{u}$ further enforces fitting to a solution of the PDE, thus preventing over-fitting to noise.

The loss term $\mathcal{L}_{sparse}$ is further introduced:

$$\mathcal{L}_{sparse}(\mathbf{c}) = \|\mathbf{c}\|_1 \quad (15)$$

to impose the assumption that natural systems are inherently simple and are thus dependent on a few terms. The PDE-learning-net training is presented in Algorithm 1.

ODE-informed-net

An ODE is a differential equation containing one or more functions of one independent variable and the derivatives of those functions. The term ordinary is used in contrast with the term partial differential equation which may be with respect to more than one independent variables. ODE-informed-net, in this paper, is proposed to implicitly approximate time-dependence with a neural network and solved via an ODE solver. The model structure is shown in Figure 4.

ODE-informed-net training is performed with the DOPRI5 method⁴⁰ in the ODE solver and with an absolute and relative tolerance of $1e^{-3}$. The DOPRI5 method is well established as an ODE solver and allows for adaptive steps, thus enabling the model to perform network evaluations with a dynamic and arbitrary number of times. This can be tuned in and outside of training by changing tolerances before network evaluation.

Algorithm 1: PDE-learning-net

Input : Spatio-temporal point $(\mathbf{x}, t)$; Dictionary of differential terms $\{D(\hat{u}, \mathbf{x}, t)\}$

Output : $\mathbf{c}$ (the estimated coefficients of PDE)

Initialize

 Neural network parameters: θ of $\hat{u}$, $\theta_{\mathbf{c}} / \|\theta_{\mathbf{c}}\|$;

for number of epochs **do**

$\mathcal{L}_u(\mathbf{x}; \theta) \leftarrow \frac{1}{N} \sum_{i=1}^N (u_i - \hat{u}(\mathbf{x}_i; \theta))^2$;

$\mathcal{L}_d \leftarrow \|D(\hat{u}, \mathbf{x})\mathbf{c}\|_2^2$;

$\mathcal{L}_{\text{sparse}} \leftarrow \|\mathbf{c}\|_1$;

$\mathcal{L} \leftarrow \mathcal{L}_u^{1/2} (1 + \lambda_d \mathcal{L}_d + \lambda_{\text{sparse}} \mathcal{L}_{\text{sparse}})$;

 Update θ and $\theta_{\mathbf{c}} \leftarrow \mathcal{L}.\text{backward}()$;

$\mathbf{c} \leftarrow \theta_{\mathbf{c}} / \|\theta_{\mathbf{c}}\|$;

end

Figure 4. Diagram of the ODE-informed-net model structure with an integrated ODE-solver. The neural network consists of an Encoder u_E , an ODE-function u_O , and a Decoder u_D .

Model inputs are a set of derivatives of the target function, the current state, and the previous state. This is then encoded into the latent space, h_1 , as:

$$h_1 = \text{Relu}([\{D^\alpha u(\mathbf{x}, t)\}] W_{\text{encoder}}^T + b_{\text{encoder}}) \quad (16)$$

The model interprets this as the initial state for the ODE-solver DOPRI5⁴⁰ $g(u, \frac{du}{dt}, \Delta t)$, where Δt is the ODE solver time step and $\frac{du}{dt}$ is approximated by a neural network, in this case, a two-linear-layer stacked fully-connected network. The following process is looped within the ODE solver:

$$h'_1 = \text{Relu}(h_1 W_{\text{ode}}^T + b_{\text{ode}}) \quad (17)$$

$$h_2 = g(h_1, h'_1, \Delta t) \quad (18)$$

where h'_1 is the time derivative of the latent space state, h_2 is the calculated latent space state at $t + \Delta t$, and t is the current time of the solver. If this time is not yet $t + 1$, t and h_1 are updated for the next iteration. Once this converges to expected tolerance, the final h_2 is decoded as the model's estimate for the true state at $t + 1$ as:

$$\hat{u}(\mathbf{x}, t + 1) = W_{\text{decoder}} \cdot h_2 + b_{\text{decoder}} \quad (19)$$

Here, the ODE-informed-net unites a sub-network acting as an ODE function $\hat{u}_O$. By leveraging the multiple evaluations of this sub-network, the ODE-informed-net can effectively act as a deeper network while using fewer parameters and being more stable in training. When used to predict time-series, the ODE function solves for future states from some initial state at $u(t_0)$ as:

$$\frac{du}{dt} = f(u; \theta); \quad u(t_0) = (u_1(t_0), \dots, u_d(t_0))^T \quad (20)$$

where the united ODE function $\hat{u}_O$ fits the time derivative of u , mimicking the behavior of the physical systems. In more complex systems, if it is assumed that the system can be described with PDEs, then time gradients can be estimated as a function

Algorithm 2: ODE-informed-net

Input : Observations: $u(\mathbf{x}, t), u(\mathbf{x}, t - 1)$
Output : Prediction: $\hat{u}(\mathbf{x}, t + 1)$
Initialize
 Neural network parameters θ : $[\theta_E, \theta_O, \theta_D]$ of Encoder $\hat{u}_E$, ODE Function $\hat{u}_O$ and Decoder $\hat{u}_D$;
for number of epochs **do**
 for $\mathbf{x}$ in Ω **do**
 Perform finite difference for $\{D^\alpha u(\mathbf{x}, t)\}$;
 $u_E \leftarrow \hat{u}_E(u(\mathbf{x}, t), u(\mathbf{x}, t - 1), \{D^\alpha u(\mathbf{x}, t)\}; \theta_E)$;
 $u_O \leftarrow \int_0^1 \hat{u}_O(u(\mathbf{x}, \tau); \theta_O) d\tau + u_E$;
 $\hat{u}(\mathbf{x}, t + 1) \leftarrow \hat{u}_D(u_O; \theta_D)$;
 end
 $\mathcal{L} \leftarrow \|u(\mathbf{x}, t + 1) - \hat{u}(\mathbf{x}, t + 1)\|_2 + \alpha \|[\theta_E, \theta_O, \theta_D]\|_2$;
 Update $\theta_E, \theta_O, \theta_D \leftarrow \mathcal{L}.\text{backward}()$;
end

of the local state, previous state, and a set of spatial derivatives. These derivatives can be estimated via finite difference and be more formally defined as the set $\{D^\alpha u\}; \alpha = 0, 1, \dots, m$, where m is the highest order of derivative with a typical value of 2.

Our implementation performs this calculation at individual points in space and predicts the state of that point in the next time step. By leveraging ODE integration tools, any future values can be solved with arbitrary accuracy. When approximating time series data, the ODE-informed-net is trained using pairs of data at sequential time steps to predict values for the next time step. The data is sampled at successive time points $t - 1$ and t at some spatial point $\mathbf{x}$ and their corresponding values $u(\mathbf{x}, t - 1), u(\mathbf{x}, t)$, where $u = (u_1, u_2, \dots, u_d)^T$ is a d -dimension vector representing data values at time t and point $\mathbf{x}$. Finally, an estimate of $u(\mathbf{x}, t + 1)$ is calculated at each point using the input states as the initial state for the ODE integration.

Regression loss $\mathcal{L}$ is defined for ODE-informed-net as a function of the MSE and the network parameters θ :

$$\mathcal{L}(\theta, \alpha) = \|u - \hat{u}\|_2 + \alpha \|\theta\|_2 \quad (21)$$

where $\hat{u}$ denotes the prediction. The ODE-informed-net training is presented in Algorithm 2.

Coupling Hetero- and Homogeneity

Based on above analysis, we describe how the ST-PCNN represents the heterogeneous properties of spatio-temporal data and captures the homogeneous physics of the raw data. Here, a stacking coupling mechanism is proposed to integrate the obtained physics into the spatio-temporal learning.

As shown in Figure 2 (a), at each time step t and location i, j , the FN produces the initial prediction $\hat{S}_{he}^{(t+1, i, j)}$ based on the current observation $S^{(t, i, j)}$, the hidden states $h^{(t-1, i, j)}$ from previous-step (within LSTM), and the lateral information from its neighbors produced by TN. Here the ‘heterogeneous’ initial prediction leverages its own specific local attributes only. Then, regarding the integration of differential equations, the previous-step initial prediction $\hat{S}^{(t-1, i, j)}$ and the current observation $S^{(t, i, j)}$ are fed into the learnt physics PDEs/ODEs from PN to derive the numerical solution $\hat{S}_{ho}^{(t+1, i, j)}$, which is the ‘homogeneous’ part of the dynamics governed by physical laws. Finally, a coupling layer with parameters $\theta_C = [W_C, b_C]$ in Eq. (22), is used to produce final prediction $\hat{S}^{(t+1, i, j)}$ by synthesizing $\hat{S}_{he}^{(t+1, i, j)}$ and $\hat{S}_{ho}^{(t+1, i, j)}$.

$$\hat{S}^{(t+1, i, j)} = \text{Relu}([\hat{S}_{he}^{(t+1, i, j)}, \hat{S}_{ho}^{(t+1, i, j)}]W_C^T + b_C) \quad (22)$$

ST-PCNN training is presented in Algorithm 3 with supervised loss including sum of l_1 -norm and l_2 -norm loss. The l_2 -norm is useful for reducing outliers and the l_1 -norm is helpful for achieving sparseness, which has been shown to improve accuracy⁴¹. Since the two norms based representation methods have their own advantages^{41, 42}, i.e., the robustness of l_1 -norm and stability of l_2 -norm, we use a combination of the two for optimization.

Experiments and Comparative Study

Datasets

In this section, we evaluate ST-PCNN on a synthetic dataset and a real-world ocean current dataset, with the data statistics summarized in Table 1 and details introduced hereafter.

Algorithm 3: ST-PCNN Training

Input : Dynamics: $S \in \mathbb{R}^{T \times H \times W}$; Grid: $\Omega \in \mathbb{R}^{H \times W}$;
 $PN \leftarrow \text{ODE-informed-net/PDE-learning-net}$;
Initialize
 Neural network parameters: $\theta_T, \theta_F, \theta_C$;
 Static info: $\tilde{\mathbf{p}} \leftarrow \text{Positional-Encoding}(d, H, W)$;
 Lateral info: $\mathbf{L} \leftarrow \mathbf{0}$;
for number of epochs **do**
 for t in T **do**
 for i, j in Ω **do**
 $\hat{\mathbf{L}}_{enc}^{(t,i,j)} \leftarrow TN(\tilde{\mathbf{p}}^{(i,j)}, \mathbf{L}^{(t,i,j)}, \theta_T)$;
 $[\hat{S}_{he}^{(t+1,i,j)}, \hat{\mathbf{L}}^{(t+1,i,j)}] \leftarrow FN(\tilde{\mathbf{p}}^{(i,j)}, S^{(t,i,j)}, \hat{\mathbf{L}}_{enc}^{(t,i,j)}, \theta_F)$;
 $\hat{S}_{ho}^{(t+1,i,j)} \leftarrow PN(\hat{S}^{(t-1,i,j)}, S^{(t,i,j)})$;
 $\hat{S}^{(t+1,i,j)} = \text{Coupling}(\hat{S}_{he}^{(t+1,i,j)}, \hat{S}_{ho}^{(t+1,i,j)}, \theta_C)$;
 end
 Update $\mathbf{L}^{(t+1,:)} \leftarrow \hat{\mathbf{L}}^{(t+1,:)}$;
 end
 $\mathcal{L} \leftarrow \frac{1}{n} \|S^{(t,:)} - \hat{S}^{(t,:)}\|_1 + \frac{1}{n} \|S^{(t,:)} - \hat{S}^{(t,:)}\|_2$;
 Update $\theta_T, \theta_F, \theta_C \leftarrow \mathcal{L}.\text{backward}()$
end

Table 1. The synthetic and real-world data statistics.

Data Sets	# of Time Points	Grid Size	Sampling Rate
Reflected Wave	8,000	16×16	0.1s
LC of GoM	1,810	29×36	12h

Reflected Wave Simulation Data As illustrated in Figure 5, the problem consist of a single-wave propagation from a source that reflects at the domain boundaries. The following 2D wave equation was used for the reflected wave data generation:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) \quad (23)$$

PDE solutions were solved numerically using an explicit central difference approach:

$$\frac{\partial^2 u}{\partial b^2} = \frac{u(b+h) - 2u(b) + u(b-h)}{h^2} = u_{bb} \quad (24)$$

where b stands for a variable of function u , and h is the approximation step size. Equation (24) then becomes:

$$c^2(u_{xx} + u_{yy}) = \frac{u(x, y, t + \Delta t) - 2u(x, y, t) + u(x, y, t - \Delta t)}{\Delta_t^2} \quad (25)$$

which can be solved for $u(x, y, t + \Delta t)$ to obtain an equation for determining the state of the system at the next time step $t + \Delta t$ at each point. Both the boundary conditions ($x < 0$ or $x > \text{domainsize}$, similarly for y) and initial condition (at $t = 0$) were set at zero and the system's parameters were set to: $\Delta_t = 0.1$, $\Delta_x = \Delta_y = 1$ and $c = 3.0$. The velocity field was initialized with the Gaussian distribution

$$u(x, y, 0) = a \exp \left(- \left(\frac{(x - s_x)^2}{2\sigma_x^2} + \frac{(y - s_y)^2}{2\sigma_y^2} \right) \right), \quad (26)$$

with an amplitude factor $a = 0.34$, a wave width in the x and y directions $\sigma_x^2 = \sigma_y^2 = 0.5$, and where s_x, s_y are the initial position and center of the circular wave.

Figure 5. Visualization of circular wave amplitude (m) with boundary reflection.

(a)

(b)

Figure 6. (a): The Loop Current of Gulf of Mexico. Image credit: National Academies of Sciences, Engineering and Medicine. (b): Locations of moorings and Pressure-Recording Inverted Echo Sounders (PIES) deployed in the U.S. and Mexican sectors in the eastern Gulf of Mexico⁴³.

Gulf of Mexico (GoM) Loop Current Data As illustrated in Figure 6, the sensor array was placed in the GoM region, encompassing the region 89°W to 85°W , and 25°N to 27°N with a 30–50 km horizontal resolution, where the Loop Current (LC) extends northward and, more importantly, where eddy shedding events occurred most often⁴³. This sensor array consisted of 25 pressure-recording inverted echo sounders (PIES), 9 full-depth tall moorings with temperature, conductivity and velocity measurements, and 7 near bottom current meter moorings deployed under the LC region. The dataset contains velocity data gathered from June 2009 to June 2011. Since the sampling frequency from multiple sensors varied from minutes to hours, we processed the dataset with a fourth order Butterworth filter and sub-sampled at 12-hour intervals, leading to a total of 1,810 records (905 days).

Baseline Models for Comparative Study

- **FC-LSTM**⁴⁴ has been proven powerful for handling temporal correlation. The LSTM network here consists of 256 hidden units.
- **ConvLSTM**¹¹ has convolutional structures in LSTM cell to capture spatiotemporal correlations. The model consists of a 3-layer ConvLSTM network with 128-64-64 hidden states and 3×3 kernels.
- **PredRNN**⁴⁵ memorizes both spatial appearances and temporal variations in a unified memory pool, which consists of two ST-LSTM layers with 128 hidden states each and 3×3 convolution filters.
- **CDNN**³² is a recently developed physics-informed network, which has a convolutional-deconvolutional module with a warping mechanism to produce an interpretable latent state – the motion field. The motion field equation $\frac{\partial I}{\partial t} + (w \nabla) I = D \nabla^2 I$ describes the transport of quantity I through advection and diffusion. This equation describes a large family of physical processes (e.g., fluid dynamics, heat conduction, etc.).

Model Details and Implementation

Forecasting Network: The FN consists of a fully-connected layer, followed by an LSTM layer with 256 hidden units (chosen from a search space of [64, 128, 256, 512]), and another fully-connected layer.

Transition Network: The lateral output dimension is set to 8, similar to the dimension of the lateral input.

Physics Network: Two types of PN are adopted: 1) *PDE-learning-net*, which is a fully-connected network of width 50 with four hidden layers with ReLU activation. 2) *ODE-informed-net* of width 50 with integrated portion consisting of two fully-connected layers with ReLU activation.

All models were trained using an ADAM optimizer with the sum of l_1 -norm and l_2 -norm loss and 0.01 learning rate. The batch size is set to 16. Learning rate decay and scheduled sampling are activated once the model does not improve after 20

Table 2. MSE (m) of recovered reflected wave amplitude by learnt physics.

Models	Single-step Modeling	
	Wave Simulation (m)	GoM Loop Current ($m \cdot s^{-1}$)
PDE-learning-net	$3.60(\pm 0.48) \times 10^{-3}$	$1.36(\pm 1.51) \times 10^{-4}$
ODE-informed-net	$6.39(\pm 1.35) \times 10^{-8}$	$2.86(\pm 1.66) \times 10^{-4}$

Figure 7. Numerical solutions of the estimated PDE by PDE-learning-net and approximations by ODE-informed-net. Each column is the visualization at different time steps.

epochs (in term of validation loss). The implementation was based on the Pytorch equipped with NVIDIA Geforce GTX 1080Ti and Titan Xp GPU with 32GB memory.

Physics Learning Analysis

In this section, we validated the hypothesis that the PDE-learning-net and ODE-informed-net are able to uncover the underlying hidden physics from raw data, and thus were able to assist the spatio-temporal networks to capture the dynamics of the natural phenomenon. Both models assessed in this section are trained against values at individual points, i.e., trained over a single sequence of synthetic reflected wave data.

The ODE-informed-net model efficacy is demonstrated via a time-series derived from a closed-loop evaluation initialized with a pair of time steps. Qualitative assessments show that the model is capable of accurately recreating the physical behavior of the ground truth data over multi-steps, as shown in Table 2 and in Figure 7. This suggests that for a simple physics example, an ODE approximation can accurately recreate the physical process with a relatively shallow network. However, the computational overhead from finite difference operations required for the implementation of the network offsets some of this benefit.

Both the true and predicted PDEs were represented as a normalized vector of coefficients $\mathbf{c}$ for the following dictionary of differential terms: $\{u_{tt}, u_{xx}, u_{yy}, u_t, u_x, u_y, u\}$. Here, the wave propagating at velocity coefficient $c = 3.0(m \cdot s^{-1})$ ⁴⁶ with normalized coefficients $\mathbf{c} = [0.0783, -0.7049, -0.7049, 0., 0., 0., 0.]$ is set as a ground truth. PDE-learning-net predicts a normalized vector of $\mathbf{c}^* = [-0.086, 0.7274, 0.677, -0.0309, 0.0637, -0.0098, 0.0119]$. The PDE coefficient estimation error was calculated by $err(\mathbf{c}, \mathbf{c}^*) = (1 - |(\mathbf{c} \cdot \mathbf{c}^*)| / (\|\mathbf{c}\| \|\mathbf{c}^*\|))^{1/2}$ proposed by Hasan et al³⁶, which is always non-negative, and is 0 if $\mathbf{c}$ and $\mathbf{c}^*$ are co-linear. An error calculation based on this metric was approximately 5.74×10^{-2} , which indicated that the inferred PDE was different from the ground truth. To evaluate its robustness, we added noise to the coefficients. The comparison between the recovered data from estimated PDE and the noised PDEs (Table 3) revealed that the PDE-learning-net is able to capture the optimal PDE expression from the data. However, a direct comparison between the estimated PDE and the numerical solution of the PDE revealed that the former was less accurate than the ODE learned solution (Table 2). It appeared that a damping term was incorrectly introduced, resulting in the fading of the wave over multiple time steps as shown in Figure 7. Notwithstanding, this PDE estimation is valuable over a traditional neural network as this method can scale over the space-time dimensions and has flexibility through the choice of the numerical solver.

Analytical Wave Reflection Prediction

In order to evaluate the performance of our models, we first compared the outputs of the baseline models (no physics informed) to our ST-PCNN model (no physics informed) for the synthetic reflected-wave prediction problem. This comparison was conducted by calculating the MSE (Table 4, top-half) of each solution shown in Figure 8a. Pred-RNN exhibited the smallest MSE for 10-steps teacher forcing (number of initial steps that use the ground truth from a prior time step as input), whereas ST-PCNN consistently outperformed all the baseline models for >10 -steps teacher forcing. Although ConvLSTM and FC-LSTM yielded the best single-step prediction, they tend to quickly over-fit, which leads to a fading prediction compared to Pred-RNN and ST-PCNN (Fig. 8a). Considering that the multistep closed-loop (the prediction from previous step to predict the next time

Table 3. MSE (m) of recovered reflected wave amplitude data (Magnitude= 1×10^{-3}).

Estimated PDE+noise%	$\mathbf{c}^*$	$\mathbf{c}^*+5\%$	$\mathbf{c}^*+10\%$
Single-step Modeling	3.60(± 0.48)	3.65(± 0.48)	3.81(± 0.50)

Table 4. The MSE $\pm$ Std (m) of closed-loop prediction of *Analytical Reflected Wave Amplitude*.

Models	Physics	Magnitude	Multi-step Prediction with #-steps of teacher-forcing				Single-step Prediction
			10-steps tf	20-steps tf	30-steps tf	40-steps tf	
FC-LSTM	—	$\times 10^{-3}$	21.89 (± 8.83)	21.24 (± 8.09)	20.14 (± 5.39)	16.62 (± 6.57)	$5.01 (\pm 1.39) \times 10^{-5}$
ConvLSTM	—	$\times 10^{-3}$	10.07 (± 5.00)	10.02 (± 5.53)	8.42 (± 4.24)	7.33 (± 3.47)	$\mathbf{3.80 (\pm 2.16)} \times 10^{-5}$
PredRNN	—	$\times 10^{-3}$	8.81 (± 0.78)	9.08 (± 0.92)	8.88 (± 0.58)	8.04 (± 1.08)	$3.09 (\pm 1.98) \times 10^{-4}$
ST-PCNN	—	$\times 10^{-3}$	10.74 (± 0.48)	8.04 (± 0.37)	6.27 (± 0.21)	5.38 (± 0.15)	$2.25 (\pm 0.13) \times 10^{-4}$
ST-PCNN [▲]	Wave Equ	$\times 10^{-6}$	6.41(± 0.11)	5.91(± 0.06)	5.09(± 0.18)	3.40(± 0.21)	$1.76(\pm 0.003) \times 10^{-10}$
CDNN	Motion field	$\times 10^{-3}$	7.21(± 0.64)	7.19(± 1.06)	7.35(± 1.07)	6.39(± 0.06)	$5.59(\pm 0.74) \times 10^{-4}$
ST-PCNN	Predicted ODE	$\times 10^{-4}$	3.43(± 1.11)	4.00(± 1.22)	4.56(± 1.35)	5.58(± 1.66)	$2.27(\pm 0.88) \times 10^{-6}$
ST-PCNN	Predicted PDE	$\times 10^{-4}$	3.01 (± 0.37)	3.25 (± 0.34)	3.46 (± 0.39)	3.88 (± 0.33)	9.11 (± 0.27) $\times 10^{-8}$

* All the models are trained (validated) on 57 (7) sequences of 40 steps and tested on 16 new sequences of 80 steps.

step is used instead as input) prediction performance, which requires both intrinsic model stability and the maintenance of plausible ongoing dynamics, is more challenging than just single-step prediction, our ST-PCNN that distributively executes prediction in space showed the best performance overall.

Second we compared the outputs of the physics-informed baseline models with ours as shown in Table 4 (bottom-half) for the reflected-wave data problem as well. ST-PCNN[▲] with wave equation (Eq. 23) informed is regarded as a the reference here. It shows that if the governing physics is known, the model can perfectly capture the spatio-temporal dynamics even in the long-term. The CDNN, which discretizes the solution of the motion field (advection-diffusion equation) with a warping scheme to inform prediction, was not able to capture the complex long-term dynamics and strongly deviated from the truth after 40 steps (1 step equals to 0.1s in simulation) (Fig. 8a). It indicates that pre-given knowledge (e.g., PDE of ‘general form’) may not always be beneficial in physical process modeling, since it neglects the ‘heterogeneity’ of the underlying dynamics. On the contrary, the MSE score of ST-PCNN, either informed by predicted ODE or PDE, was better than any of the baselines with/without physics information. It indicates that ST-PCNN can capture both ‘homogeneity’ (underlying physics) and ‘heterogeneity’ (localized information) in modeling the evolution and dynamics of natural phenomena.

GoM Loop Current Prediction

Understanding the dynamics of the LC is fundamental to understanding the Gulf of Mexico’s full oceanographic system. Hurricane intensity, offshore safety, oil spill response, fisheries, and the Gulf Coast economy are all affected by the position, strength, and structure of the LC and associated eddies. The LC’s position varies greatly from its retracted state in the Yucatan Channel, directly east of the Florida Straits, to its extended state into the far northern and western Gulf. Of particular interest to the modeling community is the prediction of the velocity field and the duration of the LC circulation at a given location. However, useful forecasts of the flow field by current modeling methods do not exceed two days⁴⁷. For the previous synthetic wave problem, the data were represented on a regular Cartesian grid with constant spatial resolution and generated by a unique physical process through the analytical model, where such situation may not exist in real-world applications such as the GoM LC data. This challenging practical problem is used to evaluate the ST-PCNN’s ability to handle irregular gridded data with complex physical dynamics. In this evaluation, only the surface field is predicted.

The same type of comparison as for the wave reflection problem was conducted to evaluate the prediction of the LC dynamics by both no and physics-informed models. The MSE results are shown in Table 5 and the LC predicted velocity magnitude are shown in Figure 8b. Among the no-physics informed models, ST-PCNN outperformed the other models for all prediction ranges. In addition it exhibits the lowest single-step forecast error. Among the physics-informed models, the CDNN achieved the lowest short-term forecast MSE while the predicted ODE-informed ST-PCNN reached the lowest MSE for both the single and multi-step predictions. ST-PCNN MSE was less than $3.5 \times 10^{-2} m.s^{-1}$ after a 15-steps (180 hours) prediction, which is of the same order of magnitude as the best forecaster error of $1.0 \times 10^{-2} m.s^{-1}$ after only 48 hours in the LC forecasting evaluation by Cooper et al.⁴⁷.

(a)

(b)

Figure 8. (a) Wave amplitude (m) closed-loop prediction after 10-steps teacher-forcing. (b) Gulf of Mexico Loop Current surface velocity magnitude ($m \cdot s^{-1}$) multi-step closed-loop forecast with 10-steps teacher-forcing. Each column is the prediction at a different time step t ((a) 0.1s; (b) 12 hours) by baselines and proposed ST-PCNN models;

Table 5. The MSE $\pm$ Std ($m \cdot s^{-1}$) of closed-loop prediction of *Gulf of Mexico Loop Current Velocity*.

Models	# params	Physics	Multi-step Prediction Horizon ($\times 10^{-2}$)			Single-step Forecasting
			5-steps	10-steps	15-steps	
FC-LSTM	# 3.81m	—	4.46(± 2.38)	5.39(± 3.16)	6.49(± 3.63)	4.81 (± 2.73) $\times 10^{-2}$
ConvLSTM	# 1.33m	—	1.15(± 0.60)	3.11(± 1.67)	6.65(± 2.34)	4.49(± 1.40) $\times 10^{-4}$
PredRNN	# 6.41m	—	1.76(± 0.76)	4.68(± 1.97)	6.77(± 2.71)	8.39(± 6.21) $\times 10^{-4}$
ST-PCNN	# 0.53m	—	0.61 (± 0.20)	1.84 (± 0.52)	4.68 (± 1.02)	3.35 (± 0.03) $\times 10^{-4}$
CDNN	# 6.46m	Motion field	0.14 (± 0.02)	1.38(± 0.39)	4.32(± 1.34)	1.21(± 0.32) $\times 10^{-3}$
ST-PCNN	# 0.53m	Predicted ODE	0.38(± 0.05)	1.33 (± 0.43)	3.32 (± 1.81)	1.91 (± 0.81) $\times 10^{-4}$
ST-PCNN	# 0.53m	Predicted PDE	0.40(± 0.08)	1.49(± 0.49)	3.35(± 0.81)	2.29(± 0.73) $\times 10^{-4}$

* All the models are trained (validated) on 1,466 (163) steps and tested on the final 181 steps.

Conclusions

In this paper, we proposed a ST-PCNN model for accurate spatio-temporal forecasting. We argued that real-world dynamical systems are characterized by both heterogeneity and homogeneity. The former is driven by the general physics laws governing the system, whereas the latter is the product of the localized processes in spatial and temporal regions. Therefore, by coupling three networks to learn the underlying physics, enable transition of node interactions to capture the localized processes, and then predict the future evolution of the full dynamical system, ST-PCNN showed superior performance over baseline models that do not include a physics-informed component. The key contribution of this paper is three-fold: 1) we have created a novel hybrid learning framework for capturing complex localized spatial-temporal correlations; 2) our proposed framework includes a physics network that can efficiently learn hidden physics (e.g., scientific laws that governing the dynamical systems) using a 2D PDE-learning-net or ODE-informed-net; and 3) informed by the learned physics, our proposed physics-coupled neural network is able to conduct useful long-term predictions by using only limited observations. Finally, the proposed ST-PCNN framework can be adapted for many other complex, dynamic physical processes modeling.

References

1. Li, Y., Zheng, W., Wang, L., Zong, Y. & Cui, Z. From regional to global brain: A novel hierarchical spatial-temporal neural network model for eeg emotion recognition. *IEEE Transactions on Affect. Comput.* (2019).
2. Seebacher, D. *et al.* Visual analysis of spatio-temporal event predictions: Investigating the spread dynamics of invasive species. *IEEE Transactions on Big Data* (2018).
3. Li, Y., Zhu, Z., Kong, D., Xu, M. & Zhao, Y. Learning heterogeneous spatial-temporal representation for bike-sharing demand prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 1004–1011 (2019).
4. Bian, J., Xiong, H., Fu, Y. & Das, S. K. Cswa: Aggregation-free spatial-temporal community sensing. In *Thirty-Second AAAI Conference on Artificial Intelligence* (2018).
5. Nguyen, H., Liu, W. & Chen, F. Discovering congestion propagation patterns in spatio-temporal traffic data. *IEEE Transactions on Big Data* **3**, 169–180 (2016).
6. Yu, B., Yin, H. & Zhu, Z. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875* (2017).
7. Seo, S. & Liu, Y. Differentiable physics-informed graph networks. *arXiv preprint arXiv:1902.02950* (2019).
8. Zhu, J. Y., Sun, C. & Li, V. O. An extended spatio-temporal granger causality model for air quality estimation with heterogeneous urban big data. *IEEE Transactions on Big Data* **3**, 307–319 (2017).
9. La Gatta, V., Moscato, V., Postiglione, M. & Sperli, G. An epidemiological neural network exploiting dynamic graph structured data applied to the covid-19 outbreak. *IEEE Transactions on Big Data* (2020).
10. Williams, B. M. & Hoel, L. A. Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results. *J. transportation engineering* **129**, 664–672 (2003).
11. Xingjian, S. *et al.* Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems*, 802–810 (2015).

12. Guo, S., Lin, Y., Li, S., Chen, Z. & Wan, H. Deep spatial-temporal 3d convolutional neural networks for traffic data forecasting. *IEEE Transactions on Intell. Transp. Syst.* **20**, 3913–3926 (2019).
13. Li, Y., Yu, R., Shahabi, C. & Liu, Y. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926* (2017).
14. Song, C., Lin, Y., Guo, S. & Wan, H. Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 914–921 (2020).
15. Bai, L. *et al.* Stg2seq: Spatial-temporal graph to sequence model for multi-step passenger demand forecasting. *arXiv preprint arXiv:1905.10069* (2019).
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. Berg, J. & Nyström, K. Data-driven discovery of pdes in complex datasets. *J. Comput. Phys.* **384**, 239–252 (2019).
18. Xu, H., Chang, H. & Zhang, D. DI-pde: Deep-learning based data-driven discovery of partial differential equations from discrete and noisy data. *arXiv preprint arXiv:1908.04463* (2019).
19. Raissi, M., Perdikaris, P. & Karniadakis, G. E. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561* (2017).
20. Al-Aradi, A., Correia, A., Naiff, D., Jardim, G. & Saporito, Y. Solving nonlinear and high-dimensional partial differential equations via deep learning. *arXiv preprint arXiv:1811.08782* (2018).
21. Sirignano, J. & Spiliopoulos, K. Dgm: A deep learning algorithm for solving partial differential equations. *J. computational physics* **375**, 1339–1364 (2018).
22. Long, Z., Lu, Y., Ma, X. & Dong, B. Pde-net: Learning pdes from data. In *International Conference on Machine Learning*, 3208–3216 (2018).
23. Hu, P., Yang, W., Zhu, Y. & Hong, L. Revealing hidden dynamics from time-series data by odenet. *arXiv preprint arXiv:2005.04849* (2020).
24. Zhang, C. & Patras, P. Long-term mobile traffic forecasting using deep spatio-temporal neural networks. In *Proceedings of the Eighteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 231–240 (2018).
25. Wang, K., Xu, C., Zhang, Y., Guo, S. & Zomaya, A. Y. Robust big data analytics for electricity price forecasting in the smart grid. *IEEE Transactions on Big Data* **5**, 34–45 (2017).
26. Huang, Y., Tang, Y., Zhuang, H., VanZwieten, J. & Cherubin, L. Physics-informed tensor-train convlstm for volumetric velocity forecasting. *arXiv preprint arXiv:2008.01798* (2020).
27. Wang, R., Maddix, D., Faloutsos, C., Wang, Y. & Yu, R. Bridging physics-based and data-driven modeling for learning dynamical systems. In *Learning for Dynamics and Control*, 385–398 (PMLR, 2021).
28. Wikle, C. K. Modern perspectives on statistics for spatio-temporal data. *Wiley Interdiscip. Rev. Comput. Stat.* **7**, 86–98 (2015).
29. Cressie, N. & Wikle, C. K. *Statistics for spatio-temporal data* (John Wiley & Sons, 2015).
30. Ladický, L., Jeong, S., Solenthaler, B., Pollefeys, M. & Gross, M. Data-driven fluid simulations using regression forests. *ACM Transactions on Graph. (TOG)* **34**, 1–9 (2015).
31. Tompson, J., Schlachter, K., Sprechmann, P. & Perlin, K. Accelerating eulerian fluid simulation with convolutional networks. In *International Conference on Machine Learning*, 3424–3433 (PMLR, 2017).
32. de Bezenac, E., Pajot, A. & Gallinari, P. Deep learning for physical processes: Incorporating prior scientific knowledge. *J. Stat. Mech. Theory Exp.* **2019**, 124009 (2019).
33. Lutter, M., Ritter, C. & Peters, J. Deep lagrangian networks: Using physics as model prior for deep learning. *arXiv preprint arXiv:1907.04490* (2019).
34. Zhong, Y. D., Dey, B. & Chakraborty, A. Symplectic ode-net: Learning hamiltonian dynamics with control. *arXiv preprint arXiv:1909.12077* (2019).
35. González-García, R., Rico-Martínez, R. & Kevrekidis, I. G. Identification of distributed parameter systems: A neural net based approach. *Comput. & chemical engineering* **22**, S965–S968 (1998).

36. Hasan, A., Pereira, J. M., Ravier, R., Farsiu, S. & Tarokh, V. Learning partial differential equations from data using neural networks. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 3962–3966 (IEEE, 2020).
37. Rackauckas, C. *et al.* Universal differential equations for scientific machine learning. *arXiv preprint arXiv:2001.04385* (2020).
38. Zadeh, A. *et al.* Memory fusion network for multi-view sequential learning. In *Thirty-Second AAAI Conference on Artificial Intelligence* (2018).
39. Vaswani, A. *et al.* Attention is all you need. In *Advances in neural information processing systems*, 5998–6008 (2017).
40. Wanner, G. & Hairer, E. *Solving ordinary differential equations II* (Springer Berlin Heidelberg, 1996).
41. Su, J. *et al.* Convolutional tensor-train lstm for spatio-temporal learning. *arXiv preprint arXiv:2002.09131* (2020).
42. Zhu, Q., Zhang, D., Sun, H. & Li, Z. Combining l1-norm and l2-norm based sparse representations for face recognition. *Optik* **126**, 719–724 (2015).
43. Hamilton, P., Lugo-Fernández, A. & Sheinbaum, J. A loop current experiment: Field and remote measurements. *Dyn. Atmospheres Ocean*. **76**, 156–173 (2016).
44. Gers, F. A., Schmidhuber, J. & Cummins, F. Learning to forget: Continual prediction with lstm. *Neural computation* **12**, 2451–2471 (2000).
45. Wang, Y., Long, M., Wang, J., Gao, Z. & Philip, S. Y. Predrnn: Recurrent neural networks for predictive learning using spatiotemporal lstms. In *Advances in Neural Information Processing Systems*, 879–888 (2017).
46. Tarasov, V. Discrete wave equation with infinite differences. *Appl. Math. & Inf. Sci. Lett.* **5**, 41–44 (2017).
47. Cooper, C. *et al.* Real Time Observing and Forecasting of Loop Currents in 2015. vol. Day 2 Tue, May 03, 2016 of *OTC Offshore Technology Conference*, DOI: [10.4043/26920-MS](https://doi.org/10.4043/26920-MS) (2016).

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- [Appendix.pdf](#)