

Graph Networks with Physics-aware Knowledge Informed in Latent Space

Sungyong Seo and Yan Liu

{sungyons, yanliu.cs} at usc.edu

Department of Computer Science

University of Southern California

Abstract

While physics conveys knowledge of nature built from an interplay between observations and theory, it has been considered less important for modeling deep neural networks. Despite the usefulness of physical rules, it is particularly challenging to leverage the knowledge for sparse data since most physics equations are well defined on the continuous and dense space. In addition, it is even harder to inform the equations into a model if the observations are not fully governed by the given physical knowledge. In this work, we present a novel architecture to incorporate physics or domain knowledge given as a form of partial differential equations (PDEs) on sparse observations by utilizing graph structure. Moreover, we leverage the representation power of deep learning by informing the knowledge in latent space. We demonstrate that climate prediction tasks are significantly improved and validate the effectiveness and importance of the proposed model.

Introduction

Modeling natural phenomena in the real-world, such as climate, traffic, molecule, and so on, is extremely challenging but important. Deep learning has achieved significant successes in prediction performance by learning latent representations from *data-rich* applications such as speech recognition (Hinton et al. 2012), text understanding (Wu et al. 2016), and image recognition (Krizhevsky, Sutskever, and Hinton 2012). While the accuracy and efficiency of data-driven deep learning models can be improved with ad-hoc architectural changes for specific tasks, we are confronted with many challenging learning scenarios in modeling natural phenomenon, where a limited number of labeled examples are available, there is much noise in the data, and there could be constant changes in data distributions (e.g. dynamic systems). Furthermore, in many domains, data are only available on scattered collections of points (sensors or point clouds, see Figure 1) where the majority of existing methods are not applicable. These challenges are not easily addressed under the purely data-driven learning models and therefore, there is a pressing need to develop new generation robust learning models that can address these challenging learning scenarios.

Copyright © 2021 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

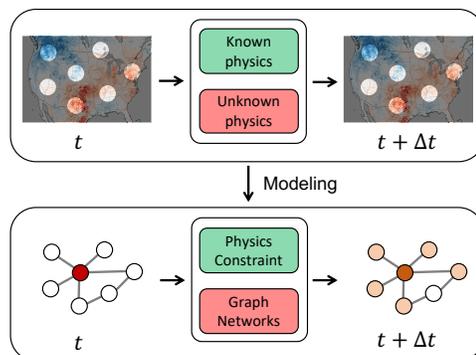


Figure 1: Concept of the proposed PaGN. Many sensor-based observations are only sparsely available (See circled regions) but there are continuous physical process (e.g., Diffusion) behind the sparse observations. Some of the known physics rules are injected into a model and the remained unknown dynamics will be extracted from data.

Physics is one of the fundamental pillars describing how the real-world behaves. It is imperative that physics-informed learning models are powerful solutions to modeling natural phenomena. Incorporating domain knowledge has several benefits: first, it helps an optimized solution to be more stable and to prevent overfitting; second, it provides theoretical guidance with which an optimized model is supposed to follow and thus, helps training with fewer data; lastly, since a model is driven by the desired inductive bias, it would be more robust to unseen data, and thus it is easier to enable accurate extrapolation.

In the meanwhile, there exist a series of challenges when we incorporate physics principles into machine learning models. First, a model needs to properly handle the spatial and temporal constraints. Many physics equations demonstrate how a set of physical quantities behaves on space and time. For example, the wave equation describes how a signal is propagated through a medium over time. Second, the model should capture relations between objects, such as image patches (Santoro et al. 2017) or rigid bodies (Battaglia et al. 2016; Chang et al. 2017). Third, the learning modules should be shared over all objects because physical laws are commonly applicable to all objects. Finally, the model should be flexible to extract unknown patterns instead of be-

ing strictly constrained to the physics knowledge. Since it is not always possible to describe all rules governing real-world data, data-driven learning is required to fill the gap between the known physics and real observations.

In this paper, we address the problem of modeling dynamical systems based on graph neural networks by incorporating useful knowledge described as differentiable physics equations. We propose a generic architecture, physics-aware graph networks (PaGN), which can leverage explicitly required physics and learn implicit patterns from data as illustrated in Figure 1. The proposed model properly handles spatially distributed objects and their relations as vertices and edges in a graph. Moreover, temporal dependencies are learned by recurrent computations. As Battaglia et al. (2018) suggest, the inductive bias of a graph-based model is its *invariance [to] node/edge permutations*, and thus, all trainable functions for the same input types are shared.

Our contributions of this work are summarized as follows:

- We develop a novel physics-aware learning architecture, PaGN, which incorporates differentiable physics equations with a graph network framework.
- We explore the performance of PaGN on graph signal prediction tasks to demonstrate that the physics knowledge is helpful to provide a significant improvement in prediction tasks and make a model more robust.
- We investigate the effectiveness and the importance of PaGN from climate prediction to provide how physics knowledge can be beneficial for prediction performance.

Related Work

Incorporating physics Among many attempts incorporating physical knowledge into data-driven models, Cressie and Wikle (2015) covered a number of statistical models (e.g., a hierarchical Bayesian framework) handling physical equations. Raissi, Perdikaris, and Karniadakis (2017a) introduced a concept of physics-informed neural networks, which utilize physics equations explicitly to train neural networks. By optimizing the model at initial/boundary and sampled collocation points, the data-driven solutions of nonlinear PDEs can be found. Based on this fundamental idea, a number of works for simulating and discovering PDEs have been published (Raissi and Karniadakis 2018; Raissi 2018; Raissi, Perdikaris, and Karniadakis 2017b). Although these works leveraged physical knowledge, they are limited because they require all physics behind given data to be explicitly known.

de Bezenac, Pajot, and Gallinari (2018) considered a similar problem as ours. They proposed how transport physics (advection and diffusion) could be incorporated for forecasting sea surface temperature (SST). In other words, they proposed how the motion flow that is helpful for the temperature flow prediction could be extracted in an unsupervised manner from a sequence of SST images.

This work is a major milestone since it captures not only the dominant transport physics but also unknown patterns inferred through the neural networks. Despite of its novel architecture, the model is specifically designed for transport physics and it is not straightforward to extend the model to

other physics equations. Furthermore, it is restricted in a regular grid to use conventional convolutional neural networks (CNNs) for images.

Discovering physical dynamics A class of models (Grzeszczuk, Terzopoulos, and Hinton 1998; Battaglia et al. 2016; Chang et al. 2017; Watters et al. 2017; Sanchez-Gonzalez et al. 2018; Kipf et al. 2018) have been proposed based on the assumption that neural networks can learn complex physical interactions and simulate unseen dynamics based on a current state. The models along this direction are based on common *relational inductive biases* (Santoro et al. 2017; Battaglia et al. 2018), i.e., functions connecting entities and relations are shared and can be learned from a given sequence of simulated dynamics. (Chang et al. 2017; Battaglia et al. 2016; Sanchez-Gonzalez et al. 2018) commonly assumed that the objects’ behaviors were governed by classical kinetic physics equations. Then, object- and relation-centric functions were proposed to learn the transition from the current state to the next state without explicitly injecting the equations into the model. Discovering latent physics by data-driven learning has been actively studied (Long et al. 2018; Brunton, Proctor, and Kutz 2016). While the properly constrained filters enable us to identify the governing PDEs, it is only applicable when we are aware of the form of target PDEs. Unlike this line of works that extracts latent patterns from data only, our proposed model can incorporate known physics and at the same time extract latent patterns from data which cannot be captured by existing knowledge.

Background

In this section, we introduce how differential operators in Euclidean domain are analogously defined on the discrete graph domain and briefly show that the graph networks module is able to efficiently express the differential operators.

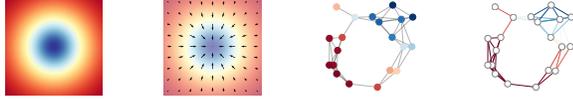
Calculus on Graphs

Preliminary Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where \mathcal{V} and \mathcal{E} are a set of vertices $\mathcal{V} = \{1, \dots, n\}$ and edges $\mathcal{E} \subseteq \binom{\mathcal{V}}{2}$, respectively, two types of real functions can be defined on the vertices, $f : \mathcal{V} \rightarrow \mathbb{R}$, and edges, $F : \mathcal{E} \rightarrow \mathbb{R}$, of the graph. It is also possible to define multiple functions on the vertices or edges as multiple feature maps of a pixel in CNNs. Since f and F can be viewed as scalar and vector fields in differential geometry (Figure 2), the corresponding discrete operators on graphs can be defined as follow (Bronstein et al. 2017).

Gradient on graphs The *gradient* on a graph is the linear operator defined by

$$\begin{aligned} \nabla : L^2(\mathcal{V}) &\rightarrow L^2(\mathcal{E}) \\ (\nabla f)_{ij} &= (f_j - f_i) \quad \text{if } \{i, j\} \in \mathcal{E} \text{ and } 0 \text{ otherwise.} \end{aligned}$$

where $L^2(\mathcal{V})$ and $L^2(\mathcal{E})$ denote Hilbert spaces of vertex and edge functions, respectively, thus $f \in L^2(\mathcal{V})$ and $F \in L^2(\mathcal{E})$. As the gradient in Euclidean space measures the rate



(a) Scalar field (b) Vector field (c) Vertex func (d) Edge func

Figure 2: Scalar/vector fields on Euclidean space and vertex/edge functions on a graph.

and direction of change in a scalar field, the gradient on a graph computes *differences* of the values between two adjacent vertices and the differences are defined along the directions of the corresponding edges.

Divergence on graphs The *divergence* in Euclidean space maps vector fields to scalar fields. Similarly, the divergence on a graph is the linear operator defined by

$$\begin{aligned} \text{div} : L^2(\mathcal{E}) &\rightarrow L^2(\mathcal{V}) \\ (\text{div } F)_i &= \sum_{j:(i,j) \in \mathcal{E}} w_{ij} F_{ij} \quad \forall i \in \mathcal{V} \end{aligned}$$

where w_{ij} is a weight on the edge (i, j) . It denotes a weighted sum of incident edge functions to a vertex i , which is interpreted as the netflow at a vertex i .

Laplacian on graphs *Laplacian* ($\Delta = \nabla^2$) in Euclidean space measures the difference between the values of the scalar field with its average on infinitesimal balls. Similarly, the graph Laplacian is defined as

$$\begin{aligned} \Delta : L^2(\mathcal{V}) &\rightarrow L^2(\mathcal{V}) \\ (\Delta f)_i &= \sum_{j:(i,j) \in \mathcal{E}} w_{ij} (f_i - f_j) \quad \forall i \in \mathcal{V} \end{aligned}$$

The graph Laplacian can be represented as a matrix form, $L = D - W$ where $D = \text{diag}(\sum_{j:j \neq i} w_{ij})$ is a degree matrix and W denotes a weighted adjacency matrix. Note that $L = \Delta = -\text{div}\nabla$ and the minus sign is required to make L positive semi-definite.

Based on the core differential operators on a graph, we can re-write differentiable physics equations (e.g., Diffusion equation or Wave equation) on a graph.

Graph Networks

Battaglia et al. (2018) proposed a graph networks framework, which generalizes relations among vertices, edges, and a whole graph. Graph Networks (GN) describe how edge, node, and global attributes are updated by propagating information among themselves.

Given a set of nodes (\mathbf{v}), edges (\mathbf{e}), and global (\mathbf{u}) attributes, the steps of computation in a graph networks block are as follow:

1. $\mathbf{e}'_{ij} \leftarrow \phi^e(\mathbf{e}_{ij}, \mathbf{v}_i, \mathbf{v}_j, \mathbf{u})$ for all $\{i, j\} \in \mathcal{E}$ pairs.
2. $\mathbf{v}'_i \leftarrow \phi^v(\mathbf{v}_i, \bar{\mathbf{e}}'_i, \mathbf{u})$ for all $i \in \mathcal{V}$.
 $\bar{\mathbf{e}}'_i$ is an aggregated edge attribute related to the node i .
3. $\mathbf{u}' \leftarrow \phi^u(\mathbf{u}, \bar{\mathbf{e}}', \bar{\mathbf{v}}')$
 $\bar{\mathbf{e}}'$ and $\bar{\mathbf{v}}'$ are aggregated attributes of all edges and all nodes in a graph, respectively.

Mapping	Equation	Physics example
node → edge	$\mathbf{e}_{ij} = \phi^e(\mathbf{v}_i, \mathbf{v}_j)$ $= (\nabla \mathbf{v})_{ij}$	$\nabla \phi = -E$ (Electric field)
edge → node	$\mathbf{v}_i = \phi^v(\mathbf{e}_{ij})$ $= (\text{div } \mathbf{e})_i$	$\nabla \cdot E = \rho/\epsilon_0$ (Maxwell's eqn.)
node → node	$\mathbf{v}_i = \phi^v(\mathbf{v}_i, \{\mathbf{v}_{j:(i,j) \in \mathcal{E}}\})$ $= (\Delta \mathbf{v})_i$	$\Delta \phi = 0$ (Laplace's eqn.)

Table 1: Examples of static equations in Graph networks

where ϕ^e, ϕ^v, ϕ^u are edge, node, and global update functions, respectively, and they can be implemented by learnable neural networks. Note that the computation order is flexible. The aggregators can be chosen freely once it is invariant to permutations of their inputs.

As ϕ^e is a mapping function from vertices to edges, it can be replaced by the graph gradient operator to describe the known relation explicitly. Similarly, ϕ^v can learn divergence-like mapping (edge to node) functions. For curl-involved functions, it is required to add another updating function, ϕ^c , which is mapping from nodes/edges/global attributes to a 3-clique attribute and vice versa. In other words, the graph networks have highly flexible modules which are able to imitate the differential operators in a graph explicitly or implicitly.

Physics-aware Graph Networks

As deep learning models are successful to model complex behaviors or extract abstract features in data, it is natural to focus on how the data-driven modeling can solve practical problems in physics or engineering fields. In this section, we provide how domain knowledge described in physics can be incorporated with the graph networks framework.

Static Physics

Many fields in physics dealing with static properties, such as Electrostatic, Magnetostatic, or Hydrostatic, describe a number of physics phenomena at rest. Among the various phenomena, it is easy to express differentiable physics rules in discrete forms on a graph with the operators in previous Section . For instances, the Poisson equation ($\nabla^2 \phi = -\frac{\rho}{\epsilon_0}$) in Electrostatics is realized as a simple matrix multiplication of graph Laplacian with a vertex function. Table 1 provides some differential formulas in Electrostatic and how the updating functions are defined in graph networks.

Dynamic Physics

More practical equations have been written in the dynamic forms, which describe how a given physical quantity is changing in a given region over time. GN can be regarded as a module that updates a graph state including the attributes of node, edge, and a whole graph.

$$\mathcal{G}' = \text{GN}(\mathcal{G}) \quad (1)$$

Equation	Physics example
$\begin{aligned} \mathbf{v}'_i &= \mathbf{v}_i + \alpha \phi^v(\mathbf{v}_i, \{\mathbf{v}_{j:(i,j) \in \mathcal{E}}\}) \\ &= \mathbf{v}_i + \alpha(\Delta \mathbf{v})_i \end{aligned}$	$\begin{aligned} \dot{u} &= \alpha \Delta u \\ &\text{(Diffusion eqn.)} \end{aligned}$
$\begin{aligned} \mathbf{v}''_i &= 2\mathbf{v}'_i - \mathbf{v}_i + c^2 \phi^v(\mathbf{v}'_i, \{\mathbf{v}'_{j:(i,j) \in \mathcal{E}}\}) \\ &= 2\mathbf{v}'_i - \mathbf{v}_i + c^2(\Delta \mathbf{v}')_i \end{aligned}$	$\begin{aligned} \ddot{u} &= c^2 \Delta u \\ &\text{(Wave eqn.)} \end{aligned}$

Table 2: Examples of dynamic equations in Graph networks

where \mathcal{G}' is the updated graph state. Dynamic physics formulas are written as a function of time and spatial derivatives:

$$f\left(\frac{\partial u}{\partial t}, \dots, \frac{\partial^M u}{\partial t^M}, \frac{\partial u}{\partial x}, \dots, \frac{\partial^N u}{\partial x^N}\right) = 0 \quad (2)$$

where u is a physical quantity spatiotemporally varying and x is the direction where u is defined on. M and N denote the highest order of time and spatial derivatives, respectively. Under the state updating view in Equation 1, any types of PDEs written in Equation 2 can be represented as a form of finite differences. Table 2 provides the examples of the dynamic physics. \dot{u} and \ddot{u} are the first and second order time derivatives, respectively.

Physics in Latent Space

We provide how the differential operators are implemented in a GN module in a previous section. However, it is hardly practical for modeling complicated real-world problems with the differential operators solely because it is only possible when all physics equations governing the observed phenomena are explicitly known. For example, although we are aware that there are a number of physics equations involved in climate observations, it is almost infeasible to include all required equations for modeling the observations. Thus, it is necessary to utilize the learnable parameters in GN to fill the missing dynamics which is not described by given equations.

There is another advantage to utilize learnable parameters. There are a number of unknown parameters, which need to be pre-defined to specify the physics equations, and the parameters can be inferred by the learnable parameters. For example, while we have knowledge that input signal has a wave property, the speed of waves (c in Table 2) should be given to fully describe the wave equation. It will be even worse when multiple input signals are involved since each signal is governed by different parameters in the same kind of equation. While both temperature and surface pressure are continuous and diffusive, they should have different diffusion coefficients (α in Table 2) in the same diffusion equation. To address the issue we can transform the input signals to latent space and use one equation in the latent space instead of imposing multiple equations to input signals separately. Then, the parameters in Encoder make the different signals follow the equation differently. We formalize how this idea is implemented as follow.

Forward/Recurrent computation Figure 3 provides how the desired physics knowledge is integrated with the graph networks. Given a graph $\mathcal{G} = \{\mathbf{v}, \mathbf{e}, \mathbf{u}\}$, it is fed into an encoder which transforms a set of attributes of nodes (\mathbf{v}), edges (\mathbf{e}), and a whole graph (\mathbf{u}) into latent spaces.

$$\tilde{\mathbf{v}}, \tilde{\mathbf{e}}, \tilde{\mathbf{u}} = \text{Encoder}(\mathbf{v}, \mathbf{e}, \mathbf{u}) \quad (3)$$

After the encoder, the encoded graph $\mathcal{H} = \{\tilde{\mathbf{v}}, \tilde{\mathbf{e}}, \tilde{\mathbf{u}}\}$ is repeatedly updated within the core block as many as the required time steps T . For each step, \mathcal{H} is updated to \mathcal{H}' which denotes the next state of the encoded graph.

$$\mathcal{H}' = \text{GN}(\mathcal{H}) \quad (4)$$

Finally, the sequentially updated attributes are re-transformed to the original spaces by a decoder.

$$\mathbf{v}', \mathbf{e}', \mathbf{u}' = \text{Decoder}(\tilde{\mathbf{v}}', \tilde{\mathbf{e}}', \tilde{\mathbf{u}}') \quad (5)$$

There are two types of objective function in this architecture, physics knowledge and supervised objective. First, we define physics-informed constraint, which is a form of equations in Table 1 and 2 depending on given physics knowledge and even mixed.

$$f_{phy}^s(\mathcal{H}'_t), f_{phy}^d(\mathcal{H}'_t, \dots, \mathcal{H}'_{t+M}) \quad (6)$$

$$\mathcal{L}_{phy} = \sum_t f_{phy}^s(\mathcal{H}'_t) + f_{phy}^d(\mathcal{H}'_t, \dots, \mathcal{H}'_{t+M}) \quad (7)$$

where $f_{phy}^s(\mathcal{H}'_t)$ and $f_{phy}^d(\mathcal{H}'_t, \dots, \mathcal{H}'_{t+M})$ are the static and dynamic physics-informed quantity, respectively. For example, we can impose gradient constraint or the diffusion equation between node/edge latent representations as follow:

$$\begin{aligned} f_{phy}^s(\mathcal{H}'_t) &= \|\tilde{\mathbf{e}}'_t - \nabla \tilde{\mathbf{v}}'_t\|^2 \\ f_{phy}^d(\mathcal{H}'_t, \mathcal{H}'_{t+1}) &= \|\tilde{\mathbf{v}}'_{t+1} - \tilde{\mathbf{v}}'_t - \alpha \nabla^2 \tilde{\mathbf{v}}'_t\|^2 \end{aligned}$$

Secondly, the supervised loss function between the predicted graph, \mathcal{G}' , and the target graph, \mathcal{G} . This loss function is constructed based on the task, such as the cross-entropy or the mean squared error (MSE). Finally, the total objective function is a sum of the two constraints:

$$\mathcal{L} = \mathcal{L}_{sup} + \lambda \mathcal{L}_{phy} \quad (8)$$

where λ controls the importance of the physics term.

Experiment

In this section, we evaluate PaGN on a real-world climate dataset on the Southern California region.

Climate Data

For the evaluation on real-world data, we used the hourly simulated climate observations for 16 days on the Southern California region (Zhang et al. 2018). In this dataset, we sampled small regions randomly from two area (Los Angeles and San Diego, Figure 4) encompassing urban and rural meteorological features to generate spatially discrete observations. To build a graph, we connected a pair of the sampled regions by using k -nearest neighbors algorithm ($k = 3$). This data preprocessing is required to verify the proposed

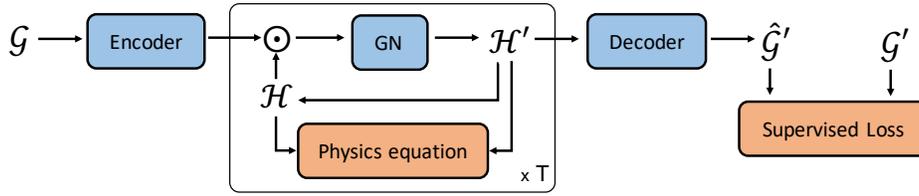


Figure 3: Recurrent architecture to incorporate physics equation on GN. The blue blocks have learnable parameters and the orange blocks are objective functions. \odot is a concatenation operator and the middle core block can be repeated as many as the required time steps (T).

idea as well as evaluate PaGN on the spatiotemporally sparse setting, which is more common for sensor-based datasets.

The vertex attributes consist of 10 climate observations, *Air temperature, Albedo, Precipitation, Soil moisture, Relative humidity, Specific humidity, Surface pressure, Planetary boundary layer height, and Wind vector (2 directions)*. While the edge attributes are not given explicitly, we could specify the type of each edge by using the type of connected regions. There are 13 different land-usage types and each type summarizes how the corresponding land is used. Based on the types of connected regions, we assigned different embedding vectors to edges.

PaGN Architecture

As explained in Section , PaGN consists of three modules, graph encoder, GN block, and graph decoder (Figure 3). The encoder contains two feed forward networks, ϕ^v and ϕ^e , applied to node and edge features, respectively. By passing the encoder, the features are transformed to the latent space (\mathcal{H}) where we will impose physics equations.

In the GN block, the node/edge/graph features are updated by the GN algorithm described in Section . The latent graph states, \mathcal{H} and \mathcal{H}' , indicate the hidden states of the current and next observations. For the physics constraint, we informed the diffusion and wave equation in Table 2, which describe the behavior of the continuous physical quantities. As the most of the climate observations are varying continuously, the diffusion equation, as a part of the continuity equation, is one of the inductive bias that should be considered for modeling. In addition, the wave equation is useful to describe atmospheric phenomena, especially 1 solar day harmonics (e.g., Atmospheric tide). Note that the physics equations are not directly applied to the input observations, but rather to the latent representations. The state-updating process is repeated at least as many as the order of the equations to provide the finite difference equation. For multistep predictions, the recurrent module is repeated as many as the number of the predictions and the physics equation will be also applied multiple times as well. Finally, the decoder takes \mathcal{H}' as input to return the next predictions. The following objective is the total loss function of PaGN with the diffusion equation.

$$\mathcal{L} = \sum_{i=1}^T \|\hat{\mathbf{y}}'_i - \mathbf{y}'_i\|^2 + \lambda \sum_{i=1}^T \|\tilde{\mathbf{v}}'_i - \tilde{\mathbf{v}}_{i-1} - \alpha \nabla^2 \tilde{\mathbf{v}}_{i-1}\|^2 \quad (9)$$

where \mathbf{y}' is a vector of the target observations (i.e. node vectors) and α adjusts the diffusivity of the latent representa-

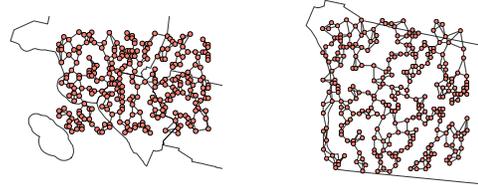


Figure 4: Sampled regions in Southern California area. (Left) Los Angeles (274 nodes) and (Right) San Diego (282 nodes) area

Model	LA area	SD area
MLP	0.8140±0.0651	0.7735±0.0539
LSTM	0.7855±0.0644	0.8123±0.0875
GN-only	0.5951±0.0517	0.6947±0.1859
GN-skip	0.5906±0.0620	0.6456±0.1499
PaGN (wave)	0.5366±0.0631	0.6413±0.1549
PaGN (diff)	0.5289±0.0405	0.5746 ±0.0471

Table 3: One step prediction error (MSE)

tions, which is found through cross validation. Note that the equation term can be replaced by other equations properly.

Experimental Settings

In our experiments, we used the air temperature as a target observation and other 9 observations were used as input. We first evaluated our model by performing the one-step and multistep prediction tasks on the two different area with a mean square error metric. For both regions, we commonly trained the model with input observations for 10 timesteps ($t - 10 : t - 1$) and predicted targets from $t - 9$ to t . First 65% of a total length was used as a training set and remaining series was split into validation (10%) and test sets (25%).

We explored several baselines: MLP, LSTM, and GN-only ignoring the physics constraint in PaGN. We also compared GN-skip which connects between \mathcal{H} and \mathcal{H}' with the skip-connection (He et al. 2016) without the physics constraint.

One step Prediction

Table 3 shows the prediction error of the baselines and PaGN on different areas. MLP and LSTM are shared over all stations and their performances are outperformed by other models leveraging a given graph structure. It implies that knowing neighboring information is significantly helpful to infer its own state and it is intuitive since climate behaviors are

Model	LA area	SD area
LSTM	1.9022±0.2078	1.2489±0.2295
GN-only	1.6137±0.1128	1.5532±0.2023
GN-skip	1.5429±0.0932	1.4423±0.1622
PaGN(diff)	1.4656±0.0474	1.0999±0.0435

Table 4: Multistep prediction error (MSE)

spatiotemporally continuous. Among the graph-based models, PaGN(diff) provides the least MSEs. It validates that the diffusive property provides a strong inductive bias with the latent representation learning. Note that the standard deviations from PaGN(diff) are significantly smaller than those of other baselines and it implies that the integrated physics knowledge properly stabilizes optimization process by introducing additional objective.

Multistep Prediction

To evaluate the effectiveness of the state-wise regularization more carefully, we conducted the multistep prediction task (10 forecast horizon). For the task, the recurrent modules are modified to predict input observations as well and the predicted one is re-fed in the model for future timesteps. While the models having a recurrent module are able to predict a few more steps reasonably, there are a couple of things we should pay attention. First, the results imply that utilizing the neighboring information is important because GN-only model shows similar or better MSEs compared to LSTM for the multistep tasks, even though it has a simple recurrent module that is not as good as that of LSTM. Second, we found that the diffusion equation in PaGN gives the stable state transition and the property provides slowly varying latent states which are desired particularly for the climate forecasting. Note that the skip-connection in GN-skip is also able to restrict the rapid changes of \mathcal{H} . However, it is necessary to more carefully optimize the parameters in GN-skip to learn the residual term in $\mathcal{H}' = \mathcal{H} + \text{GN}(\mathcal{H})$ properly.

Effectiveness of Physics Constraint

One of the benefits of physics-aware learning is data efficiency. We explore how much the physics constraint is helpful by testing if PaGN can be well-trained when the number of data for the supervised objective is limited for the one-step prediction task. We randomly sampled training data which were used to optimize the total loss function (Equation 9) and the left unsampled data were only used to minimize the physics constraint:

$$\begin{aligned} \mathcal{L} &= \mathcal{L}_{sup}^i + \lambda \mathcal{L}_{phy}^i, & i \text{ is a sampled step} \\ \mathcal{L} &= \lambda \mathcal{L}_{phy}^i, & \text{otherwise} \end{aligned}$$

We found that the diffusion equation can benefit to optimize PaGN even if the target observations are partially available (Figure 5a). Although the overall performances of PaGN are degraded when less number of sampled data are used, the error are not far deviated from those of GN-only. Even the GN-only model is outperformed by PaGN when only 70% training data are used with the state-wise constraint.

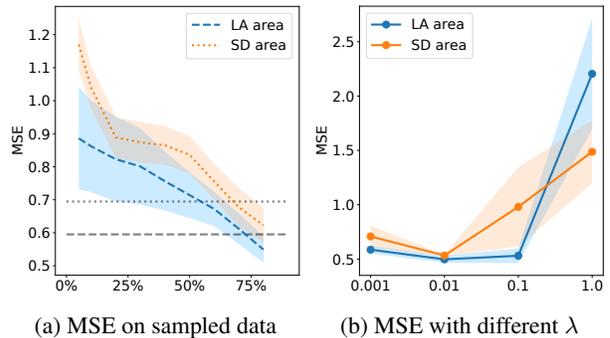


Figure 5: In (a) MSEs of PaGN are almost as good as GN-only (gray lines) despite the less number of training data. (b) provides how the prediction performance is dependent on the physics term.

Model	LA area	SD area
PaGN(rand)	1.1406	0.7073
PaGN(diff+wave)	0.5624	0.6724

Table 5: One step prediction MSE with different constraints.

Importance of Physics Constraint

To study the importance of the physics term, we trained PaGN with different λ controlling the importance of the physics term. While we found that the physics term is substantially helpful from Table 3 and 4, the term is not supposed to be dominant (See Figure 5b) but tuned properly. This is intuitive since the term only provides partial knowledge (diffusive input signals), which changes loss surface to help parameters more stable to predict next signals, instead of governing the dynamics explicitly. Scaling down the physics term is similar to what Sabour, Frosst, and Hinton (2017) did for reconstruction error not to dominate margin loss but to help the optimization process.

We also present MSEs from PaGN(rand) defined by randomly sampling $(\alpha, \beta) \in [-2.5, 2.5]$ in the constraint $\|v'' + \alpha v' + \beta v - c \Delta v\|^2$, and PaGN(diff+wave) superposing the two equations. Table 5 shows that the random equation significantly degrades the overall prediction quality. Note that the simple superposition of two equations does not always guarantee lower error even if each equation is helpful separately. When the two equations are non-linearly connected in the unknown (fully) governing equation, the superposition cannot provide meaningful inductive bias. The results demonstrate that the physics term is a useful inductive bias when it is properly defined.

Conclusion

In this work, we introduce a new architecture PaGN based on graph networks to incorporate prior knowledge given as a form of PDEs over time and space. While existing works more focus on how to discover equations in data generated by explicit physics rules, we propose a method to leverage weakly given inductive bias describing data. We empirically analyze the performance of PaGN across a range of prediction experiments on the climate observations.

References

- Battaglia, P.; Pascanu, R.; Lai, M.; Rezende, D. J.; et al. 2016. Interaction networks for learning about objects, relations and physics. In *Advances in neural information processing systems*, 4502–4510.
- Battaglia, P. W.; Hamrick, J. B.; Bapst, V.; Sanchez-Gonzalez, A.; Zambaldi, V.; Malinowski, M.; Tacchetti, A.; Raposo, D.; Santoro, A.; Faulkner, R.; et al. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261* .
- Bronstein, M. M.; Bruna, J.; LeCun, Y.; Szlam, A.; and Vandergheynst, P. 2017. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine* 34(4): 18–42.
- Brunton, S. L.; Proctor, J. L.; and Kutz, J. N. 2016. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences* 113(15): 3932–3937.
- Chang, M. B.; Ullman, T.; Torralba, A.; and Tenenbaum, J. B. 2017. A Compositional Object-Based Approach to Learning Physical Dynamics. *International Conference on Learning Representations* .
- Cressie, N.; and Wikle, C. K. 2015. *Statistics for spatio-temporal data*. John Wiley & Sons.
- de Bezenac, E.; Pajot, A.; and Gallinari, P. 2018. Deep Learning for Physical Processes: Incorporating Prior Scientific Knowledge. In *International Conference on Learning Representations*.
- Grzeszczuk, R.; Terzopoulos, D.; and Hinton, G. 1998. Neuroanimator: Fast neural network emulation and control of physics-based models. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, 9–20. ACM.
- He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.
- Hinton, G.; Deng, L.; Yu, D.; Dahl, G. E.; Mohamed, A.-r.; Jaitly, N.; Senior, A.; Vanhoucke, V.; Nguyen, P.; Sainath, T. N.; et al. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine* 29(6): 82–97.
- Kipf, T.; Fetaya, E.; Wang, K.-C.; Welling, M.; and Zemel, R. 2018. Neural Relational Inference for Interacting Systems. *International Conference on Machine Learning* .
- Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.
- Long, Z.; Lu, Y.; Ma, X.; and Dong, B. 2018. PDE-Net: Learning PDEs from Data. In *Proceedings of the 35th International Conference on Machine Learning*. URL <http://proceedings.mlr.press/v80/long18a.html>.
- Raissi, M. 2018. Deep Hidden Physics Models: Deep Learning of Nonlinear Partial Differential Equations. *arXiv preprint arXiv:1801.06637* .
- Raissi, M.; and Karniadakis, G. E. 2018. Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics* 357: 125–141.
- Raissi, M.; Perdikaris, P.; and Karniadakis, G. E. 2017a. Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations. *arXiv preprint arXiv:1711.10561* .
- Raissi, M.; Perdikaris, P.; and Karniadakis, G. E. 2017b. Physics Informed Deep Learning (Part II): Data-driven Discovery of Nonlinear Partial Differential Equations. *arXiv preprint arXiv:1711.10566* .
- Sabour, S.; Frosst, N.; and Hinton, G. E. 2017. Dynamic routing between capsules. In *Advances in neural information processing systems*, 3856–3866.
- Sanchez-Gonzalez, A.; Heess, N.; Springenberg, J. T.; Merel, J.; Riedmiller, M.; Hadsell, R.; and Battaglia, P. 2018. Graph Networks as Learnable Physics Engines for Inference and Control. In *Proceedings of the 35th International Conference on Machine Learning*.
- Santoro, A.; Raposo, D.; Barrett, D. G.; Malinowski, M.; Pascanu, R.; Battaglia, P.; and Lillicrap, T. 2017. A simple neural network module for relational reasoning. In *Advances in neural information processing systems*, 4967–4976.
- Watters, N.; Tacchetti, A.; Weber, T.; Pascanu, R.; Battaglia, P.; and Zoran, D. 2017. Visual interaction networks. *NIPS* .
- Wu, Y.; Schuster, M.; Chen, Z.; Le, Q. V.; Norouzi, M.; Macherey, W.; Krikun, M.; Cao, Y.; Gao, Q.; Macherey, K.; et al. 2016. Google’s neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* .
- Zhang, J.; Mohegh, A.; Li, Y.; Levinson, R.; and Ban-Weiss, G. 2018. Systematic Comparison of the Influence of Cool Wall versus Cool Roof Adoption on Urban Climate in the Los Angeles Basin. *Environmental science & technology* 52(19): 11188–11197.