Physics-aware Difference Graph Networks for Sparsely-Observed Dynamics

Anonymous authors
Paper under double-blind review

Abstract

Sparsely available data points cause a numerical error on finite differences which hinder to modeling the dynamics of physical systems. The discretization error becomes even larger when the sparse data are irregularly distributed so that the data defined on an unstructured grid. It is hard to build deep learning models to handle physics-governing observations on the unstructured grid. In this paper, we propose a novel architecture named Physics-aware Difference Graph Networks (PA-DGN) that exploit neighboring information to learn finite differences inspired by physics equations. PA-DGN further leverages data-driven end-to-end learning to discover underlying dynamical relations between the spatial and temporal differences in given observations. We demonstrate the superiority of PA-DGN in the approximation of directional derivatives and the prediction of graph signals on the synthetic data and the real-world climate observations from land-based weather stations.

1 Introduction

Modeling real world phenomena, such as climate observations, traffic flow, physics and chemistry simulation (Li et al., 2018; Geng et al., 2019; Long et al., 2018; de Bezenac et al., 2018; Sanchez-Gonzalez et al., 2018; Gilmer et al., 2017), is important but extremely challenging. While deep learning has achieved remarkable successes in prediction tasks by learning latent representations from *data-rich* applications such as image recognition (Krizhevsky et al., 2012), text understanding (Wu et al., 2016), and speech recognition (Hinton et al., 2012), we are confronted with many challenging scenarios in modeling natural phenomena by deep neural networks when a limited number of observations are only available. Particularly, the sparsely available data points cause substantial numerical error and the limitation requires a more principled way to redesign deep learning models.

Although many works have been proposed to model physics-simulated observations using deep learning, many of them are designed under the assumption that input is on a continuous domain. For example, Raissi et al. (2017a;b) proposed Physics-informed neural networks (PINNs) to learn nonlinear relations between input (spatial- and temporal-coordinates $(\boldsymbol{x}, t)$) and output simulated by a given PDE. Since Raissi et al. (2017a;b) use the coordinates as input and compute derivatives based on the coordinates to represent a given PDE, the setting is only valid when the data are continuously observed over spatial and temporal space.

Under the similar direction of PINNs, Chen et al. (2015) proposed a method to leverage the nonlinear diffusion process for image restoration. de Bezenac et al. (2018) incorporated the transport physics (advection-diffusion equation) with deep neural networks for forecasting sea surface temperature by extracting the motion field. Lutter et al. (2019) introduced Deep Lagrangian Networks specialized to learn Lagrangian mechanics with learnable parameters. Seo & Liu (2019) proposed a physics-informed regularizer to impose data-specific physics equations. In common, the methods in Chen et al. (2015); de Bezenac et al. (2018); Lutter et al. (2019) are not efficiently applicable to sparsely discretized input as only a small number of data points are available and continuous properties on given space are not easily recovered. It is inappropriate to directly use continuous differential operators to provide local behaviors because it is hard to approximate the continuous derivatives precisely with the sparse points (Amenta & Kil, 2004; Luo et al., 2009; Shewchuk, 2002). Furthermore, they are only applicable when the specific physics equations are explicitly given and still hard to be generalized to incorporate other types of equations.

As another direction to modeling physics-simulated data, Long et al. (2018) proposed PDE-Net which uncovers the underlying hidden PDEs and predicts the dynamics of complex systems. Ruthotto & Haber (2018) derived new CNNs: parabolic and hyperbolic CNNs based on ResNet (He et al., 2016) architecture motivated by PDE theory. While Long et al. (2018); Ruthotto & Haber (2018) are flexible to uncover hidden physics from the constrained kernels, it is still restrictive to a regular grid where the proposed constraints on the learnable filters are easily defined.

Reasoning physical dynamics of discrete objects has been actively studied (Sanchez-Gonzalez et al., 2018; Battaglia et al., 2016; Chang et al., 2016) as the appearance of graph-based neural networks (Kipf & Welling, 2017; Santoro et al., 2017; Gilmer et al., 2017). Although these models can handle sparsely located data points without explicitly given physics equations, they are purely data-driven so that the physics-inspired inductive bias, exploiting *finite differences*, is not considered at all. In contrast, our method consists of physics-aware modules allowing efficiently leveraging the inductive bias to learn spatiotemporal data from the physics system.

In this paper, we propose Physics-aware Difference Graph Networks (PA-DGN) whose architecture is motivated to leverage *differences* of sparsely available data from the physical systems. The *differences* are particularly important since most of the physics-related dynamic equations (e.g., Navier–Stokes equations) handle differences of physical quantities in spatial and temporal space instead of using the quantities directly. Inspired by the property, we first propose Spatial Difference Layer (SDL) to efficiently learn the local representations by aggregating neighboring information in the sparse data points. The layer is based on Graph Networks (GN) as it easily leverages structural features to learn the localized representations and the parameters for computing the localized features are shared. Then, the layer is connected with Recurrent Graph Networks (RGN) to be combined with temporal difference which is another core component of physics-related dynamic equations. PA-DGN is applicable to various tasks and we provide two representative tasks; the approximation of directional derivatives and the prediction of graph signals.

Our contributions are:

- We tackle a limitation of the sparsely discretized data which cause numerical error to model the physical system by proposing Spatial Difference Layer (SDL) for efficiently exploiting neighboring information under the limitation of sparsely observable points.
- We combine SDL with Recurrent Graph Networks to build PA-DGN which automatically learns the underlying spatiotemporal dynamics in graph signals.
- We verify that PA-DGN is effective in approximating directional derivatives and predicting graph signals in synthetic data. Then, we conduct exhaustive experiments to predict climate observations from land-based weather stations and demonstrate that PA-DGN outperforms other baselines.

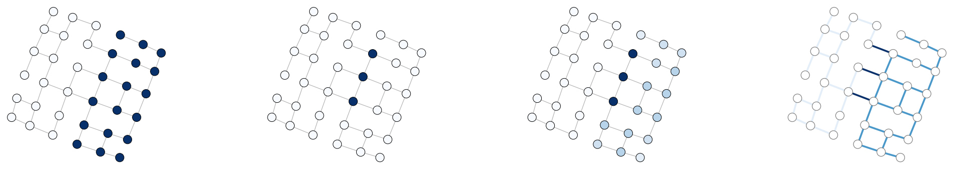
2 Physics-aware Difference Graph Network

In this section, we introduce the building module used to learn spatial differences of graph signals and describe how the module is used to predict signals in the physics system.

2.1 Difference Operators on Graph

As approximations of derivatives in continuous domain, difference operators have been used as a core role to compute numerical solutions of (continuous) differential equations. Since it is hard to derive closed-form expressions of derivatives in real-world data, the difference operators have been considered as alternative tools to describe and solve PDEs in practice. The operators are especially important for physics-related data (e.g., meteorological observations) because the governing rules behind the observations are mostly differential equations.

Graph signal Given a graph $\mathcal{G} = (\mathbb{V}, \mathbb{E})$ where $\mathbb{V}$ is a set of vertices $\mathbb{V} = \{1, \ldots, N_v\}$ and $\mathbb{E}$ a set of edges $\mathbb{E} \subseteq \{(i,j) | i, j \in \mathbb{V}\}$ ($|\mathbb{E}| = N_e$), graph signals on all nodes at time t are $f(t) \in \mathbb{R}^{N_v}$ where $f : \mathbb{V} \to \mathbb{R}$. In addition, graph signals on edges can be defined similarly, $F(t) \in \mathbb{R}^{N_e}$ where $F : \mathbb{E} \to \mathbb{R}$. Note that both signals can be multidimensional.



(a) Original graph signals (b) Detected edge (c) Sharpened signals (d) Modulated gradients

Figure 1: Examples of difference operators applied to graph signal. Filters used for the processing are (b) $\sum_j (f_i - f_j)$ (c) $\sum_j (1.1 f_i - f_j)$, (d) $f_j - 0.5 f_i$.

Gradient on graph The *gradient* (∇) of a function on nodes of a graph is represented by finite difference

$$\nabla : L^2(\mathbb{V}) \to L^2(\mathbb{E}), \qquad (\nabla f)_{ij} = (f_j - f_i) \quad \text{if } (i,j) \in \mathbb{E} \text{ and } 0 \text{ otherwise.}$$

where $L^2(\mathbb{V})$ and $L^2(\mathbb{E})$ denote vector spaces of node/edge functions, respectively. The gradients on a graph provide *finite differences* of graph signals and they become corresponding edge (i, j) features.

Laplace-Beltrami operator *Laplace-Beltrami* operator (or Laplacian, Δ) in graph domain is defined as

$$\Delta : L^2(\mathbb{V}) \to L^2(\mathbb{V}), \qquad (\Delta f)_i = \sum_{j:(i,j)\in\mathbb{E}} (f_i - f_j) \quad \forall i, j \in \mathbb{V}$$

This operator is usually regarded as a matrix form in other literature, $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$ where $\boldsymbol{A}$ is an adjacency matrix and $\boldsymbol{D} = \text{diag}(\sum_{j:j\neq i} \boldsymbol{A}_{ij})$ is a degree matrix.

2.2 Difference Operators on Triangulated Mesh

According to Crane (2018), the gradient and Laplacian operator on the triangulated mesh can be discretized by incorporating the coordinates of nodes. To obtain the gradient operator, the per-face gradient of each triangular face is calculated first. Then, the gradient on each node is the area-weighted average of all its neighboring faces, and the gradient on edge (i, j) is defined as the dot product between the per-node gradient value and the direction vector $\boldsymbol{e}_{ij}$. The Laplacian operator can be discretized with Finite Element Method (FEM):

$$(\Delta f)_i = \frac{1}{2} \sum_{j:(i,j)\in\mathbb{E}} (\cot \alpha_j + \cot \beta_j)\,(f_j - f_i)$$

where node j belongs to node i's immediate neighbors ($j \in \mathbb{N}_i$) and (α_j, β_j) are two opposing angles of the edge (i, j).

2.3 Spatial Difference Layer

While the difference operators are generalized in Riemannian manifolds (Lai et al., 2013; Lim, 2015), there are numerical error compared to those in continuous space and it can be worse when the nodes are spatially far from neighboring nodes because the connected nodes ($j \in \mathbb{N}_i$) of i's node fail to represent local features around the i-th node. Furthermore, the error is even larger if available data points are sparsely distributed (e.g., sensor-based observations). In other words, since the difference operators are highly limited to immediate neighboring information only, they are unlikely to discover meaningful spatial variations behind the sparse observations. To mitigate the limitation, we propose Spatial Difference Layer (SDL) which consists of a set of parameters to define learnable difference operators as a form of *gradient* and *Laplacian* to fully utilize neighboring information:

$$(\nabla_w f)_{ij} = w_{ij}^{(g_1)}(f_j - w_{ij}^{(g_2)} f_i), \qquad (\Delta_w f)_i = \sum_{j:(i,j)\in\mathbb{E}} w_{ij}^{(l_1)}(f_i - w_{ij}^{(l_2)} f_j) \tag{1}$$

where w_{ij} are the parameters tuning the difference operators along with the corresponding edge direction $\boldsymbol{e}_{ij}$. Note that the two forms (Eq 1) are associated with edge and node features, respectively.

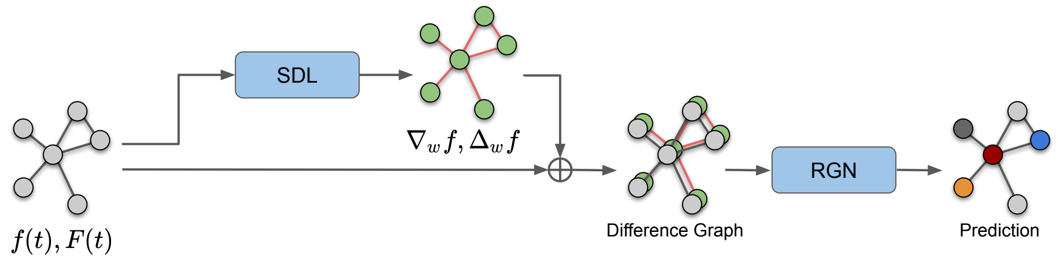


Figure 2: Physics-aware Difference Graph Networks for graph signal prediction. Blue boxes have learnable parameters and all parameters are trained through end-to-end learning. Note that the nodes/edges can be multidimensional.

The subscript in ∇_w and Δ_w denotes that the difference operators are functions of the learnable parameters w. $w_{ij}^{(g)}$ and $w_{ij}^{(l)}$ are obtained by integrating local information as follow:

$$w_{ij} = g(\{f_k, F_{mn} | f_k, F_{mn} \in h\text{-hop neighborhood of edge } (i,j)\}) \tag{2}$$

While the standard difference operators consider two connected nodes only (i and j) for each edge (i,j), Eq 2 uses a larger view (h-hop) to represent the differences between i and j nodes. Since Graph Networks (GN) (Battaglia et al., 2018) are efficient networks to aggregate neighboring information, we use GN for $g(\cdot)$ function and w_{ij} are edge features from the output of GN. Note that Eq 2 can be viewed as a higher-order difference equation because nodes/edges which are multi-hop apart are considered.

w_{ij} has a similar role of parameters in convolution kernels of CNNs. For example, while the standard gradient operator can be regarded as an example of simple edge-detecting filters, the operator can be a sharpening filter if $w_{ij}^{(g_1)} = 1$ and $w_{ij}^{(g_2)} = \frac{|\mathbb{N}_i|+1}{|\mathbb{N}_i|}$ for i node and the operators over each edge are summed. In other words, by modulating w_{ij}, it is readily extended to conventional kernels including edge detection or sharpening filters and even further complicated kernels. On top of w_{ij}, the *difference* forms in Eq 1 make an optimizing process for learnable parameters based on the differences instead of values themselves intentionally. Thus, Eq 1 naturally provides the physics-inspired inductive bias which is particularly effective for modeling physics-related observations. Furthermore, it is possible to increase the number of channels for $w_{ij}^{(g)}$ and $w_{ij}^{(l)}$ to be more expressive. Figure 1 illustrates how the exemplary filters convolve the given graph signals.

2.4 Recurrent Graph Networks

Difference graph Once the modulated spatial differences $(\nabla_w f(t), \Delta_w f(t))$ are obtained, they will be concatenated with the current signals $f(t)$ to construct node-wise ($\boldsymbol{z}_i$) and edge-wise ($\boldsymbol{z}_{ij}$) features and the graph is called a *difference graph*. Note that the difference graph includes all information to describe spatial variations.

Recurrent graph networks Given a snapshot $(f(t), F(t))$ of a sequence of graph signals, one difference graph is obtained and it is used to predict next graph signals. While a non-linear layer can be used to combine the learned spatial differences to predict the next signals, it is limited to discover spatial relations only among the features in the difference graph. Since many equations describing physics-related phenomena are non-static (e.g., Navier–Stokes equations), we adopt Recurrent Graph Networks (RGN) (Sanchez-Gonzalez et al., 2018) with a graph state $\mathcal{G}_h$ as input to combine the spatial differences with temporal variations. RGN returns a graph state ($\mathcal{G}_h^* = (\boldsymbol{h}^{*(v)}, \boldsymbol{h}^{*(e)})$) and next graph signal $\boldsymbol{z}_i^*$ and $\boldsymbol{z}_{ij}^*$. The update rule is described as follow:

1. $(\boldsymbol{z}_{ij}^*, \boldsymbol{h}^{*(e)}) \leftarrow \phi^e(\boldsymbol{z}_{ij}, \boldsymbol{z}_i, \boldsymbol{z}_j, \boldsymbol{h}^{(e)})$ for all $(i,j) \in \mathbb{E}$ pairs.
2. $(\boldsymbol{z}_i^*, \boldsymbol{h}^{*(v)}) \leftarrow \phi^v(\boldsymbol{z}_i, \bar{\boldsymbol{z}}_i', \boldsymbol{h}^{(v)})$ for all $i \in \mathbb{V}$.
   $\bar{\boldsymbol{z}}_i'$ is an aggregated edge attribute related to the node i.

where ϕ^e, ϕ^v are edge and node update functions, respectively, and they can be a recurrent unit (e.g., GRU cell). Finally, the prediction is made through a decoder by feeding the graph signal, $\boldsymbol{z}_i^*$ and $\boldsymbol{z}_{ij}^*$.

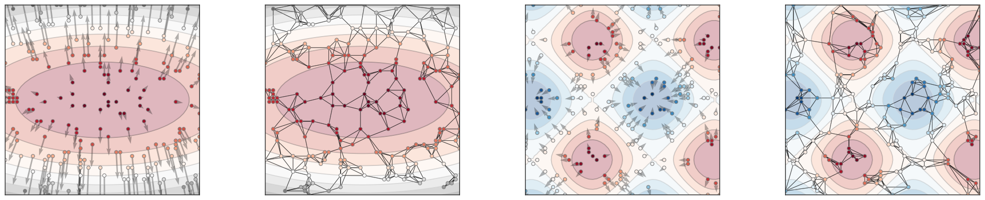


Figure 4: Gradients and graph structure of sampled points. Left: the synthetic function is $f_1(x,y) = 0.1x^2 + 0.5y^2$. Right: the synthetic function is $f_2(x,y) = \sin(x) + \cos(y)$.

Learning objective Let $\hat{f}$ and $\hat{F}$ denote predictions of the target node/edge signals. PA-DGN is trained by minimizing the following objective:

$$\mathcal{L} = \sum_{i \in \mathbb{V}} ||f_i - \hat{f}_i||^2 + \sum_{(i,j) \in \mathbb{E}} ||F_{ij} - \hat{F}_{ij}||^2 \tag{3}$$

For multistep predictions, $\mathcal{L}$ is summed as many as the number of predicting steps. If only one type (node or edge) of signal is given, the corresponding term in Eq 3 is used to optimize the parameters in SDL and RGN simultaneously.

3 Effectiveness of Spatial Difference Layer

To investigate if the proposed spatial difference forms (Eq 1) can be beneficial to learning physics-related patterns, we use SDL to two different tasks: (1) approximate directional derivatives and (2) predict synthetic graph signals.

3.1 Approximation of Directional Derivatives

As we claimed in Section 2.3, the standard difference forms (gradient and Laplacian) on a graph can become easily inaccurate because they are susceptible to a distance of two points and variations of a given function. To evaluate the applicability of the proposed SDL, we train SDL to approximate directional derivatives on a graph. First, we define a synthetic function and its gradients on 2D space and sample 200 points (x_i, y_i). Then, we construct a graph on the sampled points by using k-NN algorithm ($k = 4$). With the known gradient $\left(\nabla f = (\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y})\right)$ at each point (a node in the graph), we can compute directional derivatives by projecting ∇f to a connected edge $\boldsymbol{e}_{ij}$ (See Figure 3). We compare against four baselines: (1) the finite gradient (**FinGrad**) (2) Multilayer Perceptron Layer (**MLP**) (3) Graph Networks (**GN**) (4) a different form of Eq 1 (**One-w**). For the finite gradient ($(f_j - f_i)/||\boldsymbol{x}_j - \boldsymbol{x}_i||$), there is no learnable parameter and it only uses two points. For MLP, we feed $(f_i, f_j, \boldsymbol{x}_i, \boldsymbol{x}_j)$ as input to see whether learnable parameters can benefit the approximation or not. For GN, we use distances of two connected points as edge features and function values on the points as node features. The edge feature output of GN is used as a prediction for the directional derivative on the edge. Finally, we modify the proposed form as $(\nabla_w f)_{ij} = w_{ij} * f_j - f_i$. GN and the modified form are used to verify the effectiveness of Eq 1. Note that we define two synthetic functions (Figure 4) which have different property; (1) monotonically increasing from a center and (2) periodically varying.

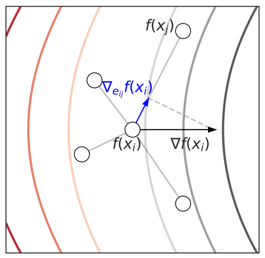


Figure 3: Directional derivative on graph

Table 1: Mean squared error (10^{-2}) for approximation of directional derivatives.

Functions	FinGrad	MLP	GN	One-w	SDL
$f_1(x,y) = 0.1x^2 + 0.5y^2$	6.42	2.12	1.05	1.41	**0.97**
$f_2(x,y) = \sin(x) + \cos(y)$	5.90	2.29	2.17	6.73	**1.26**

Approximation accuracy As shown in Table 1, the proposed spatial difference layer outperforms others by a large margin. As expected, FinGrad provides the largest error since it only considers two points without learnable parameters. It is found that the learnable parameters can significantly benefit to approximate the directional derivatives even if the input is the same (FinGrad vs. MLP). Note that utilizing neighboring information is generally helpful to learn spatial variations properly. However, simply training parameters in GN is not sufficient and explicitly defining *difference*, which is important to understand spatial variations, provides more robust inductive bias. One important thing we found is that One-w is not effective as much as GN and it can be even worse than FinGrad. It is because of its limited degree of freedom. As implied in the form $(\nabla_w f)_{ij} = w_{ij} * f_j - f_i$, only one w_{ij} adjusts the relative difference between f_i and f_j, and this is not enough to learn whole possible linear combinations of f_i and f_j. The unstable performance supports that the form of SDL is not ad-hoc but more effectively designed.

3.2 Graph Signal Prediction

We evaluate PA-DGN on the synthetic data sampled from the simulation of specific convection-diffusion equations, to provide if the proposed model can predict next signals of the simulated dynamics from observations on discrete nodes only. For the simulated dynamics, we use an equation similar to the one in Long et al. (2018).

$$\frac{df_i(t)}{dt} = a(i)(\nabla f)_{\hat{x}} + b(i)(\nabla f)_{\hat{y}} + c(i)\Delta f, \qquad f_i(0) = f_o(i) \tag{4}$$

where the index i is for pointing the i-th node whose coordinate is (x_i, y_i) in the 2D space ($[0, 2\pi] \times [0, 2\pi]$) and $\hat{x}$ and $\hat{y}$ indicate x- and y-direction in the space. $a(i) = 0.5(\cos(y_i) + x_i(2\pi - x_i)\sin(x_i)) + 0.6$, $b(i) = 2(\cos(y_i) + \sin(x_i)) + 0.8$, and $c(i) = 0.5\left(1 - \frac{\sqrt{(x_i-\pi)^2+(y_i-\pi)^2}}{\sqrt{2}\pi}\right)$. Then, we uniformly sample 250 points in the above 2D space. The task is to predict signal values of all points in the future M steps given observed values of first N steps. For our experiments, we choose $N = 5$ and $M = 15$. Since there is no a priori graph structure on sampled points, we construct a graph with k-NN algorithm ($k = 4$) using the Euclidean distance. Figure 5 shows the dynamics and the graph structure of sampled points.

To evaluate the effect of the proposed SDL on the above prediction task, we cascade SDL and a linear regression model as our prediction model since the dynamics follows a linear partial differential equation. We compare its performance with four baselines: (1) Vector Auto-Regressor (**VAR**); (2) Multi-Layer Perceptron (**MLP**); (3) **StandardOP**: the standard approximation of differential operators in Section 2.1 followed by a linear regressor; (4) **MeshOP**: similar to StandardOP but use the discretization on triangulated mesh in Section 2.2 for differential operators.

Table 2: Mean absolute error (10^{-2}) for graph signal prediction.

VAR	MLP	StandardOP	MeshOP	SDL
17.30	16.27	12.00	12.87	**11.04**

Prediction Performance Table 2 shows the prediction performance of different models measured with mean absolute error. The prediction model with our proposed spatial differential layer outperforms other baselines. All models incorporating any form of spatial differential operators (StandardOP, MeshOP and SDL) outperform those without spatial differential operators (VAR and MLP), showing that introducing spatial differences information inspired by the intrinsic dynamics helps prediction. However, in cases where points with observable signal are sparse in the space, spatial differential operators approximated with fixed rules can be inaccurate and sub-optimal for prediction since the locally linear assumption which they are based on no longer holds. Our proposed spatial differential layer, to the contrary, is capable of bridging the gap between approximated difference operators and accurate ones by introducing learnable coefficients utilizing neighboring information, and thus improves the prediction performance of the model.

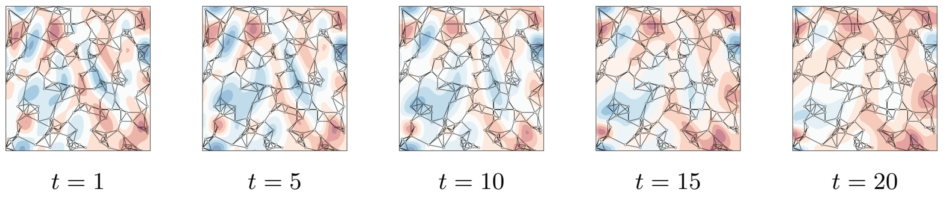


Figure 5: Synthetic dynamics and graph structure of sampled points.

4 Prediction: Graph Signals on Land-based Weather Sensors

We evaluate the proposed model on the task of predicting climate observations (*Temperature*) from the land-based weather stations located in the United States.

4.1 Experimental Set-up

Data and task We sample the weather stations located in the United States from the Online Climate Data Directory of the National Oceanic and Atmospheric Administration (NOAA) and choose the stations which have actively measured meteorological observations during 2015. We choose two geographically close but meteorologically diverse groups of stations: the Western and Southeastern states. We use k-Nearest Neighbor (NN) algorithm ($k = 4$) to generate graph structures and the final adjacency matrix is $A = (A_k + A_k^\top)/2$ to make it symmetric where A_k is the output adjacency matrix from k-NN algorithm.

Figure 6 shows the distributions of the land-based weather stations and their connectivity. Since the stations are not synchronized and have different timestamps for the observations, we aggregate the time series hourly. The 1-year sequential data are split into the train set (8 months), the validation set (2 months), and the test set (2 months), respectively.

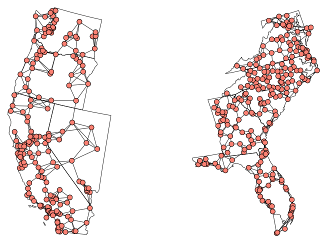


Figure 6: Weather stations in (left) western (right) southeastern states in the United States and k-NN graph.

Our main task is to predict the next graph signals based on the current and past graph signals. All methods we evaluate are trained through the objective (Eq 3) with the Adam optimizer and we use scheduled sampling (Bengio et al., 2015) for the models with recurrent modules. We evaluate PA-DGN and other baselines on two prediction tasks, (1) 1-step and (2) multistep-ahead predictions. Furthermore, we demonstrate the ablation study that provides how much the spatial derivatives are important signals to predict the graph dynamics.

4.2 Graph Signal Predictions

We compare against the widely used baselines (VAR, MLP, and GRU) for 1-step and multistep prediction. Then, we use Recurrent Graph Neural Networks (RGN) (Sanchez-Gonzalez et al., 2018) to examine how much the graph structure is beneficial. Finally, we evaluate PA-DGN to verify if the proposed architecture (Eq 1) is able to improve the prediction quality. Experiment results for the prediction task are summarized in Table 3.

Overall, RGN and PA-DGN are better than other baselines and it implies that the graph structure provides useful inductive bias for the task. It is intuitive as the meteorological observations are continuously changing over the space and time and thus, the observations of the closer stations from the i-th station are strongly related to observations at the i-th station.

PA-DGN outperforms RGN and the discrepancy comes from the fact that the spatial derivatives (Eq 1) we feed in PA-DGN are beneficial and this finding is expected because the meteorological signals at a certain point are a function of not only its previous signal but also the relative differences between neighbor signals and itself. Knowing the relative differences among local observations is particularly essential to understand physics-related dynamics. For example, Diffusion equation, which describes how physical quantities (e.g., heat) are transported through space over time, is also a function of relative differences of the quantities ($\frac{df}{dt} = D\Delta f$) rather than values of the neighbor signals. In other words, spatial differences are physics-aware features and it is desired to leverage the features as input to learn dynamics related to physical phenomena.

Table 3: Graph signal prediction results (MAE) on multistep predictions. In each row, we report the average with standard deviations from all baselines and PA-DGN. One step is 1 hour time interval.

Region	Method	1-step	6-step	12-step
West	VAR	0.1241 ± 0.0234	0.4295 ± 0.1004	0.4820 ± 0.1298
	MLP	0.1040 ± 0.0003	0.3742 ± 0.0238	0.4998 ± 0.0637
	GRU	0.0913 ± 0.0047	0.1871 ± 0.0102	0.2707 ± 0.0006
	RGN	0.0871 ± 0.0033	0.1708 ± 0.0024	0.2666 ± 0.0252
	RGN(StandardOP)	0.0860 ± 0.0018	0.1674 ± 0.0019	0.2504 ± 0.0107
	RGN(MeshOP)	0.0840 ± 0.0015	0.2119 ± 0.0018	0.4305 ± 0.0177
	PA-DGN	$\mathbf{0.0840 \pm 0.0004}$	$\mathbf{0.1614 \pm 0.0042}$	$\mathbf{0.2439 \pm 0.0163}$
SouthEast	VAR	0.0889 ± 0.0025	0.2250 ± 0.0013	0.3062 ± 0.0032
	MLP	0.0722 ± 0.0012	0.1797 ± 0.0086	0.2514 ± 0.0154
	GRU	0.0751 ± 0.0037	0.1724 ± 0.0130	0.2446 ± 0.0241
	RGN	0.0790 ± 0.0113	0.1815 ± 0.0239	0.2548 ± 0.0210
	RGN(StandardOP)	0.0942 ± 0.0121	0.2135 ± 0.0187	0.2902 ± 0.0348
	RGN(MeshOP)	0.0905 ± 0.0012	0.2052 ± 0.0012	0.2602 ± 0.0062
	PA-DGN	$\mathbf{0.0721 \pm 0.0002}$	$\mathbf{0.1664 \pm 0.0011}$	$\mathbf{0.2408 \pm 0.0056}$

4.3 Contribution of Spatial Derivatives

We further investigate if the modulated spatial derivatives (Eq 1) are effectively advantageous compared to the spatial derivatives defined in Riemannian manifolds. First, RGN without any spatial derivatives is assessed for the prediction tasks on Western and Southeastern states graph signals. Note that this model does not use any extra features but the graph signal, $f(t)$. Secondly, we add (1) **StandardOP**, the discrete spatial differences (Gradient and Laplacian) in Section 2.1 and (2) **MeshOP**, the triangular mesh approximation of differential operators in Section 2.2 separately as additional signals to RGN. Finally, we incorporate with RGN our proposed Spatial Difference Layer.

Table 3 shows the contribution of each component. As expected, PA-DGN provides much higher drops in MAE (3.56%,5.50%,8.51% and 8.73%,8.32%,5.49% on two datasets, respectively) compared to RGN without derivatives and the results demonstrate that the derivatives, namely, relative differences from neighbor signals are effectively useful. However, neither RGN with StandardOP nor with MeshOP can consistently outperform RGN. We also found that PA-DGN consistently shows positive effects on the prediction error compared to the fixed derivatives. This finding is a piece of evidence to support that the parameters modulating spatial derivatives in our proposed Spacial Difference Layer are properly inferred to optimize the networks.

5 Conclusion

In this paper, we introduce a novel architecture (PA-DGN) that approximates spatial derivatives to use them to represent PDEs which have a prominent role for physics-aware modeling. PA-DGN effectively learns the modulated derivatives for predictions and the derivatives can be used to discover hidden physics describing interactions between temporal and spatial derivatives.

REFERENCES

Nina Amenta and Yong Joo Kil. Defining point-set surfaces. In *ACM Transactions on Graphics (TOG)*, volume 23, pp. 264–270. ACM, 2004.

Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. In *Advances in neural information processing systems*, pp. 4502–4510, 2016.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pp. 1171–1179, 2015.

Michael B Chang, Tomer Ullman, Antonio Torralba, and Joshua B Tenenbaum. A compositional object-based approach to learning physical dynamics. *arXiv preprint arXiv:1612.00341*, 2016.

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

Keenan Crane. Discrete differential geometry: An applied introduction. *Notices of the AMS, Communication*, 2018.

Emmanuel de Bezenac, Arthur Pajot, and Patrick Gallinari. Deep learning for physical processes: Incorporating prior scientific knowledge. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=By4HsfWAZ`.

Xu Geng, Yaguang Li, Leye Wang, Lingyu Zhang, Qiang Yang, Jieping Ye, and Yan Liu. Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting. In *2019 AAAI Conference on Artificial Intelligence (AAAI'19)*, 2019.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1263–1272. JMLR. org, 2017.

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

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Rongjie Lai, Jiang Liang, and Hongkai Zhao. A local mesh method for solving pdes on point clouds. *Inverse Problems & Imaging*, 7(3), 2013.

Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=SJiHXGWAZ`.

Lek-Heng Lim. Hodge laplacians on graphs. *arXiv preprint arXiv:1507.05379*, 2015.

Zichao Long, Yiping Lu, Xianzhong Ma, and Bin Dong. Pde-net: Learning pdes from data. *International Conference on Machine Learning*, 2018.

Chuanjiang Luo, Issam Safa, and Yusu Wang. Approximating gradients for meshes and point clouds via diffusion metric. In *Computer Graphics Forum*, volume 28, pp. 1497–1508. Wiley Online Library, 2009.

Michael Lutter, Christian Ritter, and Jan Peters. Deep lagrangian networks: Using physics as model prior for deep learning. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=BklHpjCqKm`.

Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017a.

Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part ii): Data-driven discovery of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10566*, 2017b.

Lars Ruthotto and Eldad Haber. Deep neural networks motivated by partial differential equations. *arXiv preprint arXiv:1804.04272*, 2018.

Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin Riedmiller, Raia Hadsell, and Peter Battaglia. Graph networks as learnable physics engines for inference and control. *International Conference on Machine Learning*, 2018.

Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. In *Advances in neural information processing systems*, pp. 4967–4976, 2017.

Sungyong Seo and Yan Liu. Differentiable physics-informed graph networks. *arXiv preprint arXiv:1902.02950*, 2019.

Jonathan Richard Shewchuk. What is a good linear finite element? interpolation, conditioning, anisotropy, and quality measures (preprint). *University of California at Berkeley*, 73:137, 2002.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

A Appendix

A.1 Simulated Data

For the simulated dynamics, we discretize the following partial differential equation similar to the one in Long et al. (2018) to simulate the corresponding linear variable-coefficient convection-diffusion equation on graphs.

In a continuous space, we define the linear variable-coefficient convection-diffusion equation as:

$$\begin{cases} \frac{\partial f}{\partial t} & = a(x,y)f_x + b(x,y)f_y + c(x,y)\Delta f \\ f|_{t=0} & = f_0(x,y) \end{cases} \tag{5}$$

, with $\Omega = [0, 2\pi] \times [0, 2\pi]$, $(t, x, y) \in [0, 0.2] \times \Omega$, $a(x, y) = 0.5(\cos(y) + x(2\pi - x)\sin(x)) + 0.6$, $\quad b(x,y) = 2(\cos(y) + \sin(x)) + 0.8$, $c(x,y) = 0.5\left(1 - \frac{\sqrt{(x_i-\pi)^2+(y_i-\pi)^2}}{\sqrt{2}\pi}\right)$.

We follow the setting of initialization in Long et al. (2018):

$$f_0(x,y) = \sum_{|k|,|l|\le N} \lambda_{k,l}\cos(kx+ly) + \gamma_{k,l}\sin(kx+ly) \tag{6}$$

, where $N = 9$, $\lambda_{k,l}, \gamma_{k,l} \sim \mathcal{N}\left(0, \frac{1}{50}\right)$, and k and l are chosen randomly.

We use spatial difference operators to approximate spatial derivatives:

$$\begin{cases} f_x(x_i,y_i) & = \frac{1}{2s}(f(x_i,y_i) - f(x_i - s, y_i)) - \frac{1}{2s}(f(x_i,y_i) - f(x_i + s, y_i)) \\ f_y(x_i,y_i) & = \frac{1}{2s}(f(x_i,y_i) - f(x_i, y_i - s)) - \frac{1}{2s}(f(x_i,y_i) - f(x_i, y_i + s)) \\ f_{xx}(x_i,y_i) & = \frac{1}{s^2}(f(x_i,y_i) - f(x_i - s, y_i)) + \frac{1}{s^2}(f(x_i,y_i) - f(x_i + s, y_i)) \\ f_{yy}(x_i,y_i) & = \frac{1}{s^2}(f(x_i,y_i) - f(x_i, y_i - s)) + \frac{1}{s^2}(f(x_i,y_i) - f(x_i, y_i + s)) \end{cases} \tag{7}$$

, where s is the spatial grid size for discretization.
Then we rewrite (5) with difference operators defined on graphs:

$$\begin{cases} \frac{\partial f}{\partial t} = a(i)(\nabla f)_{\hat{x}} + b(i)(\nabla f)_{\hat{y}} + c(i)((\Delta f)_{\hat{x}\hat{x}} + (\Delta f)_{\hat{y}\hat{y}}) \\ f_i(0) = f_o(i) \end{cases} \tag{8}$$

, where

$$a(i)(x_j,y_j) = \begin{cases} \frac{a(x_i,y_i)}{2s} & \text{if } x_i = x_j + s,\ y_i = y_j \\ -\frac{a(x_i,y_i)}{2s} & \text{if } x_i = x_j - s,\ y_i = y_j \end{cases} \tag{9}$$

$$b(i)(x_j,y_j) = \begin{cases} \frac{b(x_i,y_i)}{2s} & \text{if } x_i = x_j,\ y_i = y_j + s \\ -\frac{b(x_i,y_i)}{2s} & \text{if } x_i = x_j,\ y_i = y_j - s \end{cases} \tag{10}$$

$$c(i)(x_j,y_j) = \frac{c}{s^2} \tag{11}$$

.

Then we replace the gradient w.r.t time in (8) with temporal discretization:

$$\begin{cases} f(t+1) = \Delta t(a(i)(\nabla f)_{\hat{x}} + b(i)(\nabla f)_{\hat{y}} + c(i)((\Delta f)_{\hat{x}\hat{x}} + (\Delta f)_{\hat{y}\hat{y}})) + f(t) \\ f_i(0) = f_o(i) \end{cases} \tag{12}$$

, where Δt is the time step in temporal discretization.

Equation (12) is used for simulating the dynamics described by the equation (5). Then, we uniformly sample 250 points in the above 2D space and choose their corresponding time series of u as the dataset used in our synthetic experiments. We generate 1000 sessions on a 50×50 regular mesh with time step size $\Delta t = 0.01$. 700 sessions are used for training, 150 for validation and 150 for test.

A.2 Experiment Settings

Here we provide additional details for models we used in this work, including model architecture settings and hyper-parameter settings.

A.2.1 Model Settings

Unless mentioned otherwise, all models use a hidden dimension of size 64.

- **VAR**: A vector autoregression model with 2 lags. Input is the concatenated features of previous 2 frames. The weights are shared among all nodes in the graph.
- **MLP**: A multilayer perceptron model with 2 hidden layers. Input is the concatenated features of previous 2 frames. The weights are shared among all nodes in the graph.
- **GRU**: A Gated Recurrent Unit network with 2 hidden layers. Input is the concatenated features of previous 2 frames. The weights are shared among all nodes in the graph.
- **RGN**: A recurrent graph neural network model with 2 GN blocks. Each GN block has an edge update block and a node update block, both of which use a 2-layer GRU cell as the update function. We set its hidden dimension to 73 so that it has the same number of learnable parameters as our proposed model PA-DGN.
- **RGN(StandardOP)**: Similar to RGN, but use the output of difference operators in Section 2.1 as extra input features. We set its hidden dimension to 73.
- **RGN(MeshOP)**: Similar to RGN(StandardOP), but the extra input features are calculated using opeartors in Section 2.2. We set its hidden dimension to 73.
- **PA-DGN**: Our proposed model. The spatial derivative layer uses a message passing neural network (MPNN) with 2 GN blocks using 2-layer MLPs as update functions. The forward network part uses a recurrent graph neural network with 2 recurrent GN blocks using 2-layer GRU cells as update functions.

The numbers of learnable parameters of all models are listed as follows:

Table 4: Numbers of learnable parameters.

Model	**VAR**	**MLP**	**GRU**	**RGN**	**RGN(StandardOP)**	**RGN(MeshOP)**	**PA-DGN**
# Params	3	4417	37889	345876	341057	342152	340001

A.2.2 Training Settings

The number of evaluation runs We performed 3 times for every experiment in this paper to report the mean and standard deviations.

Length of prediction For experiments on synthetic data, all models take first 5 frames as input and predict the following 15 frames. For experiments on NOAA datasets, all models take first 12 frames as input and predict the following 12 frames.

Training hyper-parameters We use Adam optimizer with learning rate 1e-3, batch size 8, and weight decay of 5e-4. All experiments are trained for a maximum of 2000 epochs with early stopping. All experiments are trained using inverse sigmoid scheduled sampling with the coefficient $k = 107$.

Environments All experiments are implemented with Python3.6 and PyTorch 1.1.0, and are run with NVIDIA GTX 1080 Ti GPUs.

A.3 Effect of different graph structures

In this section, we evaluate the effect of 2 different graph structures on baselines and our models: (1) **k-NN**: a graph constructed with k-NN algorithm ($k = 4$); (2) **TriMesh**: a graph generated with Delaunay Triangulation. All graphs use the Euclidean distance.

Table 5: Mean absolute error (10^{-2}) for graph signal prediction on the synthetic dataset.

VAR	MLP	StandardOP		MeshOP		SDL	
		k-NN	TriMesh	k-NN	TriMesh	k-NN	TriMesh
17.30	16.27	12.00	12.29	12.87	12.82	**11.04**	12.40

Table 6: Graph signal prediction results (MAE) on multistep predictions. In each row, we report the average with standard deviations from all baselines and PA-DGN. One step is 1 hour time interval.

Region	Method	Graph	1-step	6-step	12-step
West	VAR	-	0.1241 ± 0.0234	0.4295 ± 0.1004	0.4820 ± 0.1298
	MLP	-	0.1040 ± 0.0003	0.3742 ± 0.0238	0.4998 ± 0.0637
	GRU	-	0.0913 ± 0.0047	0.1871 ± 0.0102	0.2707 ± 0.0006
	RGN	k-NN	0.0871 ± 0.0033	0.1708 ± 0.0024	0.2666 ± 0.0252
		TriMesh	0.0897 ± 0.0030	0.1723 ± 0.0116	0.2800 ± 0.0414
	RGN (StandardOP)	k-NN	0.0860 ± 0.0018	0.1674 ± 0.0019	0.2504 ± 0.0107
		TriMesh	0.0842 ± 0.0011	0.1715 ± 0.0027	0.2517 ± 0.0369
	RGN (MeshOP)	k-NN	0.0840 ± 0.0015	0.2119 ± 0.0018	0.4305 ± 0.0177
		TriMesh	0.0846 ± 0.0017	0.2090 ± 0.0077	0.4051 ± 0.0457
	PA-DGN	k-NN	**0.0840 ± 0.0004**	0.1614 ± 0.0042	**0.2439 ± 0.0163**
		TriMesh	0.0849 ± 0.0012	**0.1610 ± 0.0029**	0.2473 ± 0.0162
SouthEast	VAR	-	0.0889 ± 0.0025	0.2250 ± 0.0013	0.3062 ± 0.0032
	MLP	-	0.0722 ± 0.0012	0.1797 ± 0.0086	0.2514 ± 0.0154
	GRU	-	0.0751 ± 0.0037	0.1724 ± 0.0130	0.2446 ± 0.0241
	RGN	k-NN	0.0790 ± 0.0113	0.1815 ± 0.0239	0.2548 ± 0.0210
		TriMesh	0.0932 ± 0.0105	0.2076 ± 0.0200	0.2854 ± 0.0211
	RGN (StandardOP)	k-NN	0.0942 ± 0.0121	0.2135 ± 0.0187	0.2902 ± 0.0348
		TriMesh	0.0868 ± 0.0132	0.1885 ± 0.0305	0.2568 ± 0.0328
	RGN (MeshOP)	k-NN	0.0913 ± 0.0016	0.2069 ± 0.0031	0.2649 ± 0.0092
		TriMesh	0.0877 ± 0.0020	0.2043 ± 0.0026	0.2579 ± 0.0057
	PA-DGN	k-NN	**0.0721 ± 0.0002**	**0.1664 ± 0.0011**	**0.2408 ± 0.0056**
		TriMesh	0.0876 ± 0.0096	0.2002 ± 0.0163	0.2623 ± 0.0180

Table 5 and Table 6 show the effect of different graph structures on the synthetic dataset used in Section 3.2 and the real-world dataset in Section 4.2 separately. We find that for different models the effect of graph structures is not homogeneous. For RGN and PA-DGN, k-NN graph is more beneficial to the prediction performance than TriMesh graph, because these two models rely more on neighboring information and a k-NN graph incorporates it better than a Delaunay Triangulation graph. However, switching from TriMesh graph to k-NN graph is harmful to the prediction accuracy of RGN(MeshOP) since Delaunay Triangulation is a well-defined method for generating triangulated mesh in contrast to k-NN graphs. Given the various effect of graph structures on different models, our proposed PA-DGN under k-NN graphs always outperforms other baselines using any graph structure.

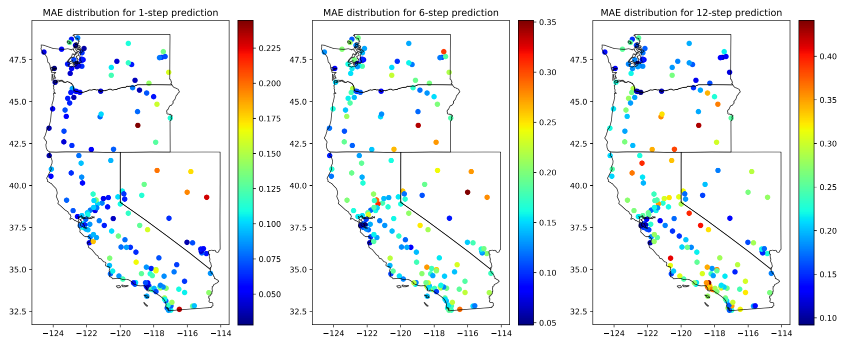

MAE distribution for 1-step prediction
MAE distribution for 6-step prediction
MAE distribution for 12-step prediction

Figure 7: MAE across the nodes.

A.4 The distribution of prediction error across nodes

Figure 7 provides the distribution of MAEs across the nodes of PA-DGN applied to the graph signal prediction task of the west coast region of the real-world dataset in Section 4.2. As shown in the figure, nodes with the highest prediction error for short-term prediction are gathered in the inner part where the observable nodes are sparse, while for long-term prediction nodes in the area with a limited number of observable points no longer have the largest MAE. This implies that PA-DGN can utilize neighboring information efficiently even under the limitation of sparsely observable points.