DIFFERENTIABLE PHYSICS-INFORMED GRAPH NETWORKS

Sungyong Seo & Yan Liu

University of Southern California
{sungyons, yanliu.cs}@usc.edu

ABSTRACT

While physics conveys knowledge of nature built from an interplay between observations and theory, it has been considered less importantly in deep neural networks. Especially, there are few works leveraging physics behaviors when the knowledge is given less explicitly. In this work, we propose a novel architecture called Differentiable Physics-informed Graph Networks (DPGN) to incorporate implicit physics knowledge which is given from domain experts by informing it in latent space. Using the concept of DPGN, we demonstrate that climate prediction tasks are significantly improved.

1 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 et al., 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 or there is much noise in the data.

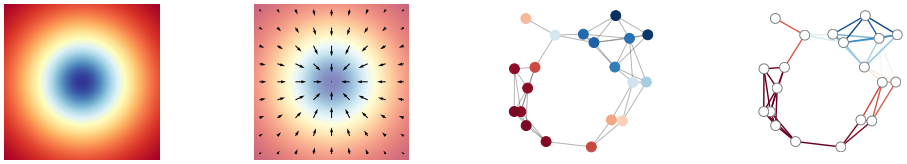
Physics is one of the fundamental pillars describing how the real-world behaves. It is imperative that physics-informed learning models are powerful solutions to modeling natural phenomena. Incorporating domain knowledge has several benefits: first, it helps an optimized solution to be more stable and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

Meanwhile, there exist a series of challenges when we incorporate physics principles into machine learning models. First, a model needs to be able to properly handle the spatial and temporal constraints. 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 common for all objects because physical phenomena apply to all objects. Finally, the model should be flexible to extract unknown patterns instead of being strictly constrained to physics knowledge.

In this paper, we address the problem of modeling dynamical systems based on graph-based neural networks by incorporating useful knowledge described as differentiable physics equations. We propose a generic architecture, differentiable physics-informed graph networks (DPGN), which can leverage explicitly required physics and learn implicit patterns from data. The proposed model properly handles spatially located objects and their relations as vertices and edges in a graph, and temporal dependencies are learned by recurrent computations.

Our contributions of this work are summarized as follows:

- We develop a novel physics-informed learning architecture, DPGN, which incorporates differentiable physics equations with a graph network framework.
- We investigate the effectiveness of DPGN for climate modeling in terms of prediction.



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

Figure 1: Scalar/vector fields on Euclidean space and vertex/edge functions on a graph.

2 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, $f : \mathcal{V} \to \mathbb{R}$, and edges, $F : \mathcal{E} \to \mathbb{R}$, 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 fields and tangent vector fields in differential geometry (Figure 1), the corresponding discrete operators on graphs can be defined as follow Lim (2015).

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

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

where $L^2(\mathcal{V})$ and $L^2(\mathcal{E})$ denote Hilbert spaces of vertex functions 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 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

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

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

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

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.

Curl on graphs The *curl* on a graph is a more complicated concept defined on 3-cliques, the set of triangles $\mathcal{T} \subseteq \binom{\mathcal{V}}{3}$ where $\{i, j, k\} \in \mathcal{T}$ if and only if $\{i, j\}, \{i, k\}, \{j, k\} \in \mathcal{E}$.

$$\text{curl} : L^2(\mathcal{E}) \to L^2(\mathcal{T}) \qquad (\text{curl } F)(i, j, k) = F(i, j) + F(j, k) + F(k, i)$$

$$\text{curl}^* : L^2(\mathcal{T}) \to L^2(\mathcal{E}) \qquad (\text{curl}^* \, \mathtt{F})(i, j) = \textstyle\sum_k \frac{w_{ijk}}{w_{ij}} \mathtt{F}(i, j, k) \quad \text{if } \{i, j, k\} \in \mathcal{T} \text{ and 0 otherwise.}$$

where $\mathtt{F}(\cdot) \in L^2(\mathcal{T})$ is a 3-clique function and w_{ijk} is a weight defined on the clique $\{i, j, k\}$. curl* is an expression for the adjoint of the curl operator. Note that some vector calculus properties (e.g., solenoidal or irrotational) can be seamlessly verified (div curl* $F = 0$ or curl $\nabla f = 0$).

3 DIFFERENTIABLE PHYSICS-INFORMED GRAPH NETWORKS

While differential operators in Section 2 can be easily implemented in a GN module, it is hardly practical for modeling complicated real-world problems with the operators solely because it is only

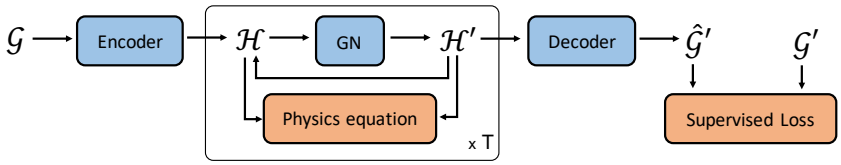


Figure 2: Recurrent architecture to incorporate physics equation on GN. The blue blocks have learnable parameters and the orange blocks are objective functions. The middle core block can be repeated as many as the required time steps (T).

possible when all physics equations governing the observed phenomena are explicitly known. For example, although we understand 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 extract latent representations in the data. Then, some known physics knowledge will be incorporated with the latent representations.

Forward/Recurrent computation Figure 2 provides how the desired physics knowledge is integrated with the learnable GN. Given a graph $\mathcal{G} = \{\boldsymbol{v}, \boldsymbol{e}, \boldsymbol{c}, \boldsymbol{u}\}$, it is fed into an encoder which transforms a set of attributes of nodes ($\boldsymbol{v}$), edges ($\boldsymbol{e}$), 3-cliques ($\boldsymbol{c}$), and a whole graph ($\boldsymbol{u}$) into latent spaces. The encoded graph $\mathcal{H} = \{\tilde{\boldsymbol{v}}, \tilde{\boldsymbol{e}}, \tilde{\boldsymbol{c}}, \tilde{\boldsymbol{u}}\}$ 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. Finally, the sequentially updated attributes are re-transformed to the original spaces by a decoder.

$$\tilde{\boldsymbol{v}}, \tilde{\boldsymbol{e}}, \tilde{\boldsymbol{c}}, \tilde{\boldsymbol{u}} = \text{Encoder}(\boldsymbol{v}, \boldsymbol{e}, \boldsymbol{c}, \boldsymbol{u}) \tag{1}$$

$$\mathcal{H}' = \text{GN}(\mathcal{H}) \tag{2}$$

$$\boldsymbol{v}', \boldsymbol{e}', \boldsymbol{c}', \boldsymbol{u}' = \text{Decoder}(\tilde{\boldsymbol{v}}', \tilde{\boldsymbol{e}}', \tilde{\boldsymbol{c}}', \tilde{\boldsymbol{u}}') \tag{3}$$

Objective functions There are two types of objective function in this architecture. First, we define physics-informed constraints (Equation 4) between the previous and updated states based on the known knowledge.

$$\mathcal{L}^i_{phy} = f_{phy}(\mathcal{H}_i, \mathcal{H}'_{i+1}, \cdots, \mathcal{H}'_{i+M-1}) \tag{4}$$

$$\mathcal{L}_{phy} = \sum_i \mathcal{L}^i_{phy} \tag{5}$$

where $\mathcal{L}^i_{phy}$ is the physics-informed quantity from the input at time step i to the predicted $M-1$ steps. For example, if we are aware that the observations should have a diffusive property, the diffusion equation can be used as the physics-informed constraint.

$$f_{phy}(\mathcal{H}, \mathcal{H}') = \|\boldsymbol{v}' - \boldsymbol{v} - \alpha\nabla^2\boldsymbol{v}\|^2$$

Secondly, the supervised loss function between the predicted graph, $\hat{\mathcal{G}}'$, 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:

$$\mathcal{L} = \mathcal{L}_{sup} + \lambda\mathcal{L}_{phy} \tag{6}$$

where λ controls the importance of the physics term.

4 Experiment

4.1 Climate Data

We found that the simulated climate observations over 16 days around the Southern California region Zhang et al. (2018) by using the Weather Research and Forecasting (WRF) model Skamarock et al. (2008). We provide the details on dataset in Appendix.

Table 1: Prediction error (MSEs)

Model	LA area (one-step)	SD area (one-step)	LA area (multistep)	SD area (multistep)
MLP	0.7930 ±0.2327	1.0645 ±0.2634	-	-
LSTM	0.7378 ±0.0514	0.9213 ±0.1049	1.4943 ±0.0970	1.0873 ±0.0664
GN-only	0.6035 ±0.0832	0.7007 ±0.0848	1.3415 ±0.1195	1.0422 ±0.0673
GN-skip	0.56546 ±0.1015	0.6543 ±0.1195	1.0257 ±0.1912	0.9872 ±0.2425
DPGN	**0.4435 ±0.0378**	**0.5149 ±0.0831**	**0.8677 ±0.1033**	**0.6714 ±0.1106**

4.2 DPGN ARCHITECTURE

In the GN block, the node/edge/graph features are updated by the GN algorithm in Sanchez-Gonzalez et al. (2018). 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 equation, which describes the behavior of the continuous physical quantities resulting from the random movement. As the most of the climate observations are varying continuously, the diffusion equation is one of the equations that should be considered for modeling. The following objective function is the total loss function of DPGN with the diffusion equation.

$$\mathcal{L} = \sum_{i=1}^{T} \|\hat{\boldsymbol{y}}_i - \boldsymbol{y}_i\|^2 + \lambda \sum_{i=1}^{T} \|\tilde{\boldsymbol{v}}_i - \tilde{\boldsymbol{v}}_{i-1} - \alpha \nabla^2 \tilde{\boldsymbol{v}}_{i-1}\|^2 \tag{7}$$

where $\boldsymbol{y}$ is a vector of the target observations and α adjusts the diffusivity of the latent physics quantities. Note that $\tilde{\boldsymbol{v}}_0$ is the latent node representation of the input $\boldsymbol{v}$ and $\tilde{\boldsymbol{v}}_{i:i>0}$ are the updated latent representations in the GN block.

4.3 ANALYSIS AND RESULTS

We evaluated our model by performing the one-step and multistep ($T = 10$) prediction tasks on the two different area with a mean square error metric. We explored several baselines, MLP which is only able to do one-step prediction, LSTM, and GN-only that ignores the physics constraint in DPGN. Moreover, we compared GN-skip which connects between $\mathcal{H}$ and $\mathcal{H}'$ with the skip-connection He et al. (2016) without the physics constraint.

Table 1 shows the prediction error of the baselines and DPGN on different areas. As expected, MLP and LSTM show the similar performance at the one-step prediction because the recurrent module in LSTM is practically a feed forward module in the setting. The results from the models leveraging the given graph structure outperform MLP and LSTM. It means that knowing neighboring information is significantly helpful to infer its own state. Among the graph-based models, DPGN provides the least MSEs. It proves that it is valid reasoning to incorporate the partially given physics rule, such as the continuously varying property, with the latent representation learning.

To evaluate the effectiveness of the state-wise regularization more carefully, we conducted the multistep prediction task. Commonly, the models having a recurrent module are able to predict a few more steps reasonably. However, 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 DPGN gives the stable state transition and the property provides slowly varying latent states which are desired particularly for the climate forecasting.

5 CONCLUSION

In this work, we introduce a new architecture based on graph networks to incorporate prior knowledge given as a form of PDEs over time and space. We first provide how the graph networks framework generalize the differential operators in a graph. Then, we present a regularization which is a function of consecutive latent states and spatial differences of the states to inject a given physics equation, and examine if the spatiotemporal constraint is valid across a range of experiments on the climate observations.

REFERENCES

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.

Michael B Chang, Tomer Ullman, Antonio Torralba, and Joshua B Tenenbaum. A compositional object-based approach to learning physical dynamics. *International Conference on Learning Representations*, 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.

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.

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

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. In *Proceedings of the 35th 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.

W. C. Skamarock, J. B. Klemp, J. Dudhia, D. O. Gill, M. Barker, K. G. Duda, X. Y Huang, W. Wang, and J. G. Powers. A description of the advanced research WRF version 3. Technical report, National Center for Atmospheric Research, 2008.

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.

Jiachen Zhang, Arash Mohegh, Yun Li, Ronnen Levinson, and George Ban-Weiss. 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, 2018.

A APPENDIX

A.1 CLIMATE DATA

We found that the simulated climate observations over 16 days around the Southern California region Zhang et al. (2018) by using the Weather Research and Forecasting (WRF) model Skamarock et al. (2008). Instead of using all patches at once, we sampled two subsets of the patches, Los Angeles and San Diego areas (Figure 3), for training DPGN. To build a graph, we considered each patch as a vertex (similar to Santoro et al. (2017)) and connect a pair of adjacent pixels to define an edge.

In this dataset, the region (Latitude: 32.22 to 35.14, Longitude: -119.59 to -116.29) is divided into 18,189 grid patches and the observations are recorded hourly.

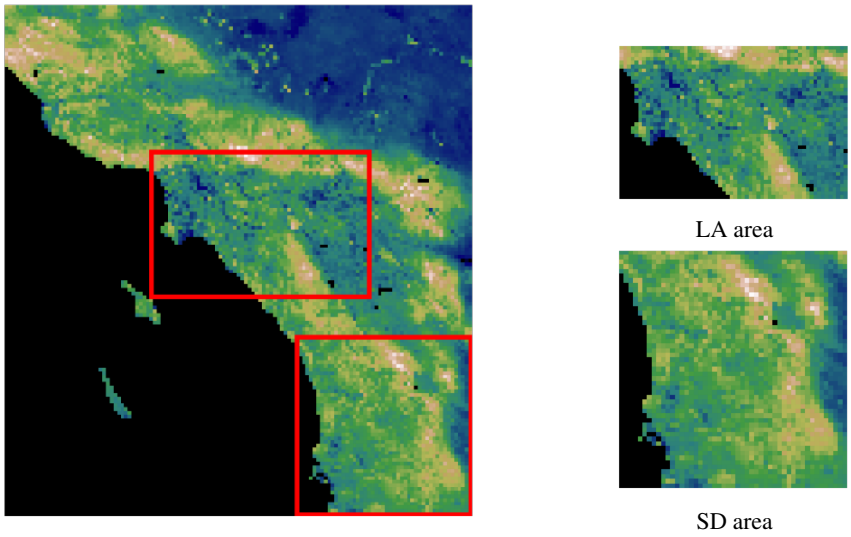


Figure 3: Southern CA region

Instead of using all patches at once, we sampled two subsets of the patches, Los Angeles and San Diego areas (Figure 3), for training DPGN. To build a graph, we considered each patch as a vertex (similar to Santoro et al. (2017)) and connect a pair of adjacent pixels to define an edge.

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).* Although the edge attributes are not given, we could specify the type of each edge by using the type of connected patches. There are 13 different land-usage types and each type summarizes how the corresponding land is used. For example, some patches are classified as commercial/industrial land (e.g., Downtown LA) but some other patches are grassland. Based on the type of connected patches, we assigned different embedding vectors to edges.

Here we provide the details for the climate observations and basic information.

Table 2: Description of climate data

Feature	Description
Timestamp	Every 60 minute
T2 (3D)	Near-surface (2 meter) air temperature (unit: K) (time varying)
XLAT (3D)	Latitude (unit: degree north) (time invariant)
XLONG (3D)	Longitude (unit: degree east) (time invariant)
ALBEDO (3D)	Albedo (unit: -) (time varying)
FRC_URB2D (3D)	Impervious fraction, urban fraction (unit: -) (time invariant)
VEGFRA (3D)	Vegetation fraction (unit: -) (time invariant)
LU_INDEX (3D)	Land use classification (unit: -) (time invariant) Paved index: (31: Low-intensity residential, 32: High-intensity residential, 33: Commercial/Industrial)
U (4D)	Wind vector, x-wind component (unit: $m * s^{-1}$) (west to east vector) (time varying)
V (4D)	Wind vector, y-wind component (unit: $m * s^{-1}$) (south to north vector) (time varying)
RAINNC (3D)	Accumulated total grid scale precipitation (unit: mm) (time varying)
SMOIS (4D)	Soil moisture (unit: $m^3 m^{-3}$) (time varying)
PBLH (3D)	Boundary layer height (unit: m) (time varying)
RH2 (3D)	2-meter relative humidity (unit: -) (time varying)
Q2 (3D)	2-meter specific humidity (units: $kgkg^{-1}$) (time varying)
PSFC (3D)	surface pressure (units: Pa) (time varying)

Table 3: Description of sub-regions

	LA area	SD area
# of total patches	2400	2499
# of sampled patches	280	272
# unpaved patches	130	217
# paved patches	148	49
# unknown patches	2	6
# edge attributes	45	42

A.2 Model Details

Here we provide additional details for all models we used in the work, including the exact hyper-parameter and architecture settings. All models were trained using the Adam optimiser, with exponential decay rate parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$. All experiments were done on GeForce GTX 1080 Ti.

Table 4: Hyper-parameter of DPGN

	hyper-parameter
Edge embedding dim	64
Edge hidden dim	64
Node hidden dim	64
Global hidden dim	64
Learning rate	0.001
Iterations	30,000
λ	1e-5
Diffusion coefficient	0.001

Table 5: Functions in DPGN

	hyper-parameter
Node encoder	MLP [9,64] with ReLU
Edge encoder	Embedding matrix
Edge update func, ϕ^e	$\boldsymbol{e}'_{ij} = \phi^e(\boldsymbol{e}_{ij}, \boldsymbol{v}_i, \boldsymbol{v}_j, \boldsymbol{u})$ MLP [256,64] with ReLU
Node update func, ϕ^v	$\boldsymbol{v}'_i = \phi^v(\boldsymbol{v}_i, \bar{\boldsymbol{e}}'_{i\cdot}, \bar{\boldsymbol{e}}'_{\cdot i}, \boldsymbol{u})$ MLP [256,64] with ReLU
Global update func, ϕ^u	$\boldsymbol{u}' = \phi^u(\boldsymbol{u}, \bar{\boldsymbol{e}}', \bar{\boldsymbol{v}}')$ MLP [192,64] with ReLU
Edge aggregator for $\boldsymbol{v}$	$\rho^e(\boldsymbol{e}_i) = \text{Add}(\boldsymbol{e}_i)$
Edge aggregator for $\boldsymbol{u}$	$\rho^e(\boldsymbol{e}_i) = \text{Avg}(\boldsymbol{e}_i)$
Node aggregator for $\boldsymbol{u}$	$\rho^v(\boldsymbol{v}_i) = \text{Avg}(\boldsymbol{v}_i)$
Node decoder	MLP [64,1]