
Neural Enhanced Dynamic Message Passing

Fei Gao

Beijing Normal University
feig@mail.bnu.edu.cn

Yan Zhang

Beijing Normal University
zyan2408@mail.bnu.edu.cn

Jiang Zhang

Beijing Normal University
zhangjiang@bnu.edu.cn

Abstract

Predicting stochastic spreading processes on complex networks is critical in epidemic control, opinion propagation, and viral marketing. We focus on the problem of inferring the time-dependent marginal probabilities of states for each node which collectively quantifies the spreading results. Dynamic Message Passing (DMP) has been developed as an efficient inference algorithm for several spreading models, and it is asymptotically exact on locally tree-like networks. However, DMP can struggle in diffusion networks with lots of local loops. We address this limitation by using Graph Neural Networks (GNN) to learn the dependency amongst messages implicitly. Specifically, we propose a hybrid model in which the GNN module runs jointly with DMP equations. The GNN module refines the aggregated messages in DMP iterations by learning from simulation data. We demonstrate numerically that after training, our model’s inference accuracy substantially outperforms DMP in conditions of various network structure and dynamics parameters. Moreover, compared to pure data-driven models, the proposed hybrid model has a better generalization ability for out-of-training cases, profiting from the explicitly utilized dynamics priors in the hybrid model. A PyTorch implementation of our model is at <https://github.com/FeiGSSS/NEDMP>.

1 INTRODUCTION

Stochastic spreading processes on complex networks are widely used for modeling the epidemic spreading,

Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151. Copyright 2022 by the author(s).

information propagation, and transmission of social behaviors. Given the initial states, accurately predicting the spreading is of primary importance in many domains. Since the stochastic nature of the processes, the more reasonable is to predict the marginal probabilities of each state for each node, as shown in Figure 1. In addition to being an accurate description of the spreading, the predicted marginal probabilities could also be used for inferring the source of spreading (Zhu & Ying, 2016), maximizing the influence by selecting a fixed size of initially activated nodes (Kempe, Kleinberg, & Tardos, 2015), and determining an optimal set of nodes to immunize (Pastor-Satorras & Vespignani, 2002).

Dynamics Message Passing (DMP) (Karrer & Newman, 2010; Shrestha & Moore, 2014; Shrestha, Scarpino, & Moore, 2015; A. Lokhov, Mézard, & Zdeborová, 2015), a special case of Belief Propagation (BP) on time trajectories, is an efficient algorithm for inferring the marginal probabilities for stochastic spreading processes on graphs. For spreading process with unidirectional dynamics (e.g., SIR, SEIR), DMP is exact on tree graphs and asymptotically exact on locally tree-like graphs, and has a linear computational complexity in the number of edges and spreading time steps. It typically yields accurate estimations on real sparse networks for a large class of spreading dynamics (A. Lokhov et al., 2015). Therefore, as an analytical inference machine, it has been used for inferring the patient zero (A. Lokhov, Mézard, Ohta, & Zdeborová, 2014), reconstructing the dynamics parameters (A. Lokhov, 2016; Wilinski & Lokhov, 2021), optimal deployment of resources (A. Lokhov & Saad, 2017), and functional immunization of networks (S. Li et al., 2020).

However, same as other Belief Propagation algorithms, the fundamental assumption of DMP is the independence of neighboring messages, which limits the ability of DMP to capture high-order interdependencies, i.e., DMP can struggle in graphs with short local loops. As illustrated in Figure 2, since the existence of local loops, DMP fails to approximate marginal probabili-

ties in an example graph with only four nodes. Unfortunately, it is extremely challenging to handle message dependence in those loops analytically (Cantwell & Newman, 2019).

The success of Graph Neural Networks (GNNs) in modelling complex pair-wise interactions (Sanchez-Gonzalez et al., 2020; Fetaya et al., 2018; Cranmer et al., 2020; Bapst et al., 2020) motivates us to model the dependence of messages in local loops using trainable GNNs. Specifically, GNNs can be used to learn a refined aggregation operation for messages beyond the naive independent assumption used in DMP. Several works have applied GNNs to improve the estimation accuracy of Belief Propagation algorithms. Gated recurrent GNNs are used in (Yoon et al., 2019) as an end-to-end trainable inference algorithm for Binary Markov Random Fields, and learned GNNs outperforms BP in loopy graphs. More recent works focus on integrating the advantages of data-driven and model-based methods. Instead of using a scalar damping factor, which is proposed to accelerate the convergence of BP, Kuck et al. (Kuck et al., 2020) replace the constant scalar factor with a trainable Neural Networks layer. The proposed hybrid model converges much faster than using scalar factors while returning an estimate of comparable quality. Methods in (Satorras & Welling, 2021; Liang & Meyer, 2021) aim to improve the estimation accuracy of BP for discrete and continuous random variables, respectively. Since the inaccuracy of BP comes from the oversimplified aggregation scheme of neighboring messages, those methods incorporate the GNNs into BP iterations as a trainable aggregation block, which is used to refine the aggregated messages in BP by learning from supervision data the complex dependence of messages. The resulting hybrid model runs BP and GNNs co-jointly, and benefits from the combination of physics prior and data-driven neural networks. However, no work has applied GNNs to improve the performance of DMP for better estimation of marginal probabilities of spreading process on graphs.

Inspired by (Yoon et al., 2019; Satorras & Welling, 2021), in this work, instead of analytically modeling the complex dependence of messages in local loops, we use a GNNs module to learn from simulation data. In order to utilize the dynamical priors encoded in DMP equations, we use GNN module only to refine the inaccuracy aggregation operation in the iteration of DMP, while the exact iteration rules in DMP remain the same. The resulting hybrid model, which we call Neural Enhanced Dynamic Message Passing (NEDMP), runs DMP and GNN jointly, and benefits from complementation of model-based and data-driven components. For better training, we design a penalty

term to enforce the monotone of predicted probabilities, which conforms the physical prior of dynamics. To verify the effectiveness of the hybrid model, utilizing the popular epidemic model Susceptible-Infected-Recovered (SIR), we conduct experiments of marginal probabilities inference on various graph structures and dynamics parameters. The main contributions can be summarized as follows: (1) We propose a customized GNNs that runs on line graph, (2) We propose a hybrid model for marginal inference, which runs GNNs and DMP jointly, (3) We design a dynamics-inspired penalty term to train the model, and (4) We conduct experiments on marginal probabilities inference problem on various graphs and dynamics parameters. The results show that all the proposed models outperform DMP, and the hybrid model, NEDMP, generalizes better than pure data-driven models.

2 BACKGROUND

Dynamic message passing has been derived as the inference algorithm for a wide range of diffusion processes. Without loss of generality, we use SIR model to demonstrate the spreading process on networks as well as the marginal probability inference problem.

2.1 Problem Setting

For the discrete SIR model on a diffusion graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where \mathcal{V} is the set of nodes and \mathcal{E} is the set of interactions, each node will take one of the three states: susceptible (S), infected (I) or recovered (R) at a specific time. Let $\sigma_i^t \in \{S, I, R\}$ be the state of node i at time step t , the states transition follows:

$$\begin{aligned} \mathbf{P}(\sigma_i^{t+1} = I | \sigma_i^t = S) &= 1 - \prod_{j \in \mathcal{N}_i} (1 - \beta_{ji} \delta_{\sigma_j^t, I}) \\ \mathbf{P}(\sigma_i^{t+1} = R | \sigma_i^t = I) &= \gamma_i, \end{aligned} \quad (1)$$

where \mathcal{N}_i is the set of neighbors of node i , $\beta_{ji} \in [0, 1]$ is the infection rate of edge $(j \rightarrow i)$, $\gamma_i \in [0, 1]$ is the recovery rate of node i , and δ is the Kronecker function.

Define the marginal probabilities of node i as:

$$\begin{aligned} P_S^i(t) &= \mathbf{P}(\sigma_i^t = S), \\ P_I^i(t) &= \mathbf{P}(\sigma_i^t = I), \\ P_R^i(t) &= \mathbf{P}(\sigma_i^t = R), \end{aligned}$$

we now formulate the problem as follows:

Marginal Probability Inference (MPI). For the SIR model on directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with infection rates $\{\beta_{ij}\}_{(i \rightarrow j) \in \mathcal{E}}$, recovery rates $\{\gamma_i\}_{i \in \mathcal{V}}$, and initial infectious nodes $\mathbb{S} \subseteq \mathcal{V}$, the goal is to infer the marginal

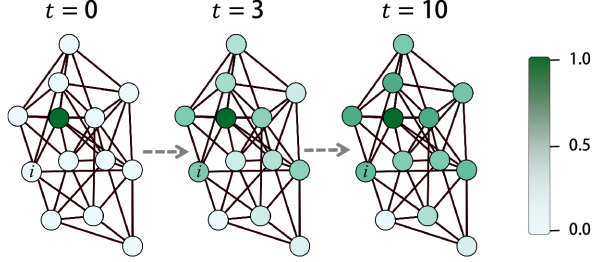


Figure 1: **Visualization of the marginal probabilities over time**, we use the SI model for illustration. Given a diffusion network with infected nodes (in darkest color) at $t = 0$, the problem aims to compute $\{P_i^i(t)\}_{i \in \mathcal{V}, t \geq 1}$, i.e., the time-dependent marginal probabilities (indicated by color) of been infected for each node. The values are obtained via 10^6 Monte Carlo simulations.

probabilities conditioned on β , γ and \mathbb{S} :

$$\{P_S^i(t), P_I^i(t), P_R^i(t)\}_{i \in \mathcal{V}, 1 \leq t \leq T} \quad (2)$$

where T is a preset time step or the convergence time, and $P_S^i(t) + P_I^i(t) + P_R^i(t) = 1$.

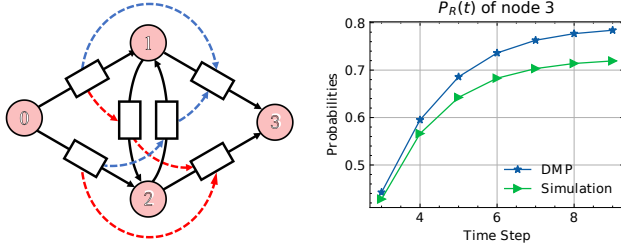


Figure 2: **An example that illustrates the limitations of DMP.** (Left): A simple example graph. Color dashed lines indicate the dependencies of θ in DMP. (Right): The approximated marginal probability $P_R(t)$ of node 3 by DMP, compared to the *true values* obtained via 10^6 Monte Carlo simulations, where the initial conditions are $\mathbb{S} = \{0\}$, $\{\beta_{ij}\} = \{\gamma_i\} = 0.5$.

In general, the exact computation of the marginals above is #P-hard (Wang, Chen, & Wang, 2012). Figure 1 is a visualization of marginals evolving over time on an example graph. We define MPI problem based on the SIR model only for convenience, and the definition can be easily generalized to other spreading models on graph, such as Independent Cascade Model (Kempe et al., 2015).

2.2 Dynamic Message Passing

Dynamic message passing is the state-of-the-art method for inferring the time-dependent marginal probabilities in stochastic processes, e.g., SIR. Utilizing the unidirectionality property of the SIR model,

the DMP equations for SIR are rigorously derived (A. Lokhov et al., 2015) from the dynamic cavity method (also known as Belief Propagation, BP) on nodes' time trajectories. It is proved (A. Y. Lokhov & Saad, 2019) that DMP is exact on trees and graphs without short loops. We briefly introduce the DMP equations for the SIR model here, and the detailed derivation is presented in the Appendix.

We begin with defining three intermediate dynamics variables:

- $\theta^{j \rightarrow i}(t)$: the probability that disease has not spread through the edge ($j \rightarrow i$) up to time t
- $P_S^{i \rightarrow k}(t)$: the probability that $\sigma_i^t = S$ when node i ignores infection from its neighbor node k ;
- $\phi^{j \rightarrow i}(t)$: the probability that disease has not spread through the edge ($j \rightarrow i$) up to time t and node j is infected at time t (i.e., $\sigma_j^t = I$).

Then, the marginal probabilities of SIR model can be approximated using:

$$P_R^i(t) = P_R^i(t-1) + \gamma_i P_I^i(t-1), \quad (3)$$

$$P_I^i(t) = 1 - P_R^i(t-1) - P_S^i(t-1), \quad (4)$$

$$P_S^i(t) = P_S^i(0) \prod_{j \in \mathcal{N}_i} \theta^{j \rightarrow i}(t), \quad (5)$$

$$P_S^{i \rightarrow k}(t) = P_S^i(0) \prod_{j \in \mathcal{N}_i \setminus k} \theta^{j \rightarrow i}(t), \quad (6)$$

The value of $\theta^{j \rightarrow i}(t)$ is computed by closed iteration:

$$\theta^{j \rightarrow i}(t) = \theta^{j \rightarrow i}(t-1) - \beta_{ji} \phi^{j \rightarrow i}(t-1), \quad (7)$$

$$\begin{aligned} \phi^{j \rightarrow i}(t) = & (1 - \beta_{ji})(1 - \gamma_j) \phi^{j \rightarrow i}(t-1) \\ & + \left(P_S^{j \rightarrow i}(t-1) - P_S^{j \rightarrow i}(t) \right), \end{aligned} \quad (8)$$

with the initial values:

$$\theta^{j \rightarrow i}(0) = 1, \phi^{j \rightarrow i}(0) = \delta_{\sigma_j^0, I} \quad (9)$$

Following the equations above, we can recursively compute the marginal probabilities of SIR model.

Like other BP algorithms, DMP builds on the assumption that the graph structure is a tree, i.e., messages from different neighbors are independent of each other. This allows DMP to take the simplest approach (i.e., \prod) to aggregate the messages from neighbors, as in Equation (26) and (27).

However, the assumption also prevents DMP from handling the high-order dependencies between neighboring messages introduced by the local loops. As

illustrated in Figure 2, when executing DMP in the simple graph, $\theta^{1 \rightarrow 3}(t)$ and $\theta^{2 \rightarrow 3}(t)$ are no longer independent when $t \geq 2$, since they all root from $\theta^{0 \rightarrow 1}(t)$ and $\theta^{0 \rightarrow 2}(t)$ through blue and red paths respectively. Thus, the resulting marginal probabilities, when estimated using Equations (26) and (27), are no longer accurate, i.e., $P_S^i(t) \neq P_S^i(0) \prod_{j \in \mathcal{N}_i} \theta^{j \rightarrow i}(t)$ and $P_S^{i \rightarrow k}(t) \neq P_S^{i \rightarrow k}(0) \prod_{j \in \mathcal{N}_i \setminus k} \theta^{j \rightarrow i}(t)$.

Analytically extending DMP to high-order neighborhood structures is very challenging (Cantwell & Newman, 2019). Therefore, a reasonable choice is to use a data-driven approach to learn a more appropriate message aggregation function that can capture the complex dependencies between messages.

3 METHOD

This section introduces our hybrid model, in which the DMP iterates jointly with a GNNs module. The GNNs module is designed to refine the aggregated messages in the iteration process of DMP.

3.1 Graph Neural Networks on Line Graph

Notice that DMP for SIR is essentially a message-passing process on line graph $\mathcal{L} = (\mathcal{N}_{\mathcal{L}}, \mathcal{E}_{\mathcal{L}})$ with non-backtracking adjacency, i.e., $\mathcal{N}_{\mathcal{L}} = \mathcal{E}$ and $\mathcal{E}_{\mathcal{L}} = \{(i \rightarrow j) \rightarrow (j \rightarrow k)\}_{i,j,k \in \mathcal{V}, i \neq k}$. To enable better integration of DMP and GNNs, we first define a special case of GNNs on line graph \mathcal{L} . Specifically, we customize and extend Gated Graph Neural Networks (Y. Li, Tarlow, Brockschmidt, & Zemel, 2016) on graph \mathcal{L} .

Mathematically, at every time step $t > 0$, each node $(i \rightarrow j)$ in graph \mathcal{L} is associated with a hidden state $h^{i \rightarrow j}(t) \in \mathbb{R}^D$. In our scenarios, node $(i \rightarrow j)$ receives time-varying inputs $x^{i \rightarrow j}(t) \in \mathbb{R}^F$. We combine the inputs with hidden states as the messages $\tilde{h}^{i \rightarrow j}(t)$:

$$\tilde{h}^{i \rightarrow j}(t) = \phi_m(h^{i \rightarrow j}(t) \oplus \phi_e(x^{i \rightarrow j}(t))), \quad (10)$$

where the \oplus is the concatenation operator. We then aggregate the incoming messages for target node $(i \rightarrow j)$:

$$\tilde{h}^{\rightarrow(i \rightarrow j)}(t) = \phi_a\left(\sum_{k \neq j, (k \rightarrow i) \in \mathcal{N}_{\mathcal{L}}} \tilde{h}^{k \rightarrow i}(t)\right), \quad (11)$$

Finally, we update the hidden states for every node $(i \rightarrow j)$ based on the aggregated messages and current states via gated recurrent unit (GRU):

$$h^{i \rightarrow j}(t+1) = GRU\left(\tilde{h}^{\rightarrow(i \rightarrow j)}(t), h^{i \rightarrow j}(t)\right). \quad (12)$$

The functions ϕ_m , ϕ_e and ϕ_a are nonlinear functions mapping input to \mathbb{R}^D , and are instantiated as multilayer perceptron (MLP) with Rectified Linear Unit

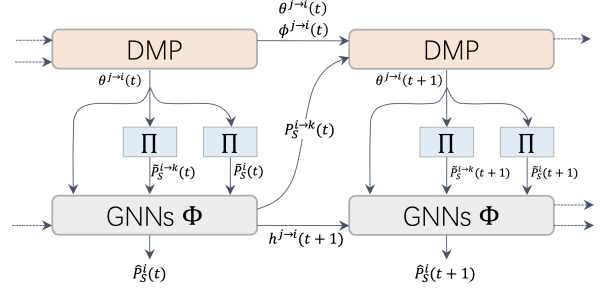


Figure 3: **Visualization of NEDMP.**

(ReLU) as the activation function. The parameters of ϕ_m , ϕ_e , ϕ_a and GRU are shared by all nodes $(i \rightarrow j)$ and time steps t .

The equations above define one iteration of GNNs from time step t to $t+1$ on line graph \mathcal{L} . To achieve time-dependent prediction, we can iterate those equations and feed $\{\tilde{h}^{i \rightarrow j}(t)\}_{t>0}$ to a nonlinear readout function $\phi_{\mathcal{R}}$:

$$\hat{y}^i(t) = \phi_{\mathcal{R}}\left(\sum_{k, (k \rightarrow i) \in \mathcal{N}_{\mathcal{L}}} \tilde{h}^{k \rightarrow i}(t)\right), \quad (13)$$

$$\hat{y}^{i \rightarrow j}(t) = \phi_{\mathcal{R}}(\tilde{h}^{\rightarrow(i \rightarrow j)}(t)), \quad (14)$$

where $\hat{y}^i(t)$ and $\hat{y}^{i \rightarrow j}(t)$ corresponding to the node-wise and edge-wise prediction in graph \mathcal{G} .

3.2 Neural Enhanced Dynamic Message Passing (NEDMP)

As discussed in Section (2.2), the inference inaccuracy of DMP is due to its inability to capture the high-order interdependencies while the graph neural networks are good at capturing high-order relations (Wu et al., 2019). In order to break this limitation of DMP and also to preserve the precise physically inspired part of DMP, we propose a hybrid model of GNNs and DMP, which is an extension of NEBP in (Satorras & Welling, 2021).

Specifically, the hybrid model runs GNNs and DMP on the line graph \mathcal{L} jointly. The GNNs module preserves and updates the hidden states for each node by incorporating the messages from DMP iterations; in return, the GNNs module outputs a refinement of aggregated messages in DMP. We name this model Neural Enhanced Dynamic Message Passing (NEDMP), which benefits from the complementary strengths.

We adopt the GNNs variant introduced in Section (3.1), termed as Φ . And the messages $\{\theta^{i \rightarrow j}(t)\}$ from DMP are provided as the inputs $\{x^{i \rightarrow j}(t)\}$ to Φ . We initialize $\{h^{i \rightarrow j}(0)\}$ as:

$$h^{i \rightarrow j}(0) = \phi_e(\theta^{i \rightarrow j}(0)) \quad (15)$$

Since we aim to refine the aggregated messages $\tilde{P}_S^i(t)$ and $\tilde{P}_S^{i \rightarrow k}(t)$ in DMP iteration, it is reasonable to integrate those values into Φ as the prediction baseline. Therefore, we modify the readout function in Φ as follows:

$$\xi^i(t), \zeta^i(t) = \phi_{\mathcal{R}} \left(\tilde{P}_S^i(t) \oplus \sum_{k, (k \rightarrow i) \in \mathcal{N}_{\mathcal{L}}} \tilde{h}^{k \rightarrow i}(t) \right), \quad (16)$$

$$\xi^{i \rightarrow j}(t), \zeta^{i \rightarrow j}(t) = \phi_{\mathcal{R}} \left(\tilde{P}_S^{i \rightarrow j}(t) \oplus \tilde{h}^{i \rightarrow j}(t) \right) \quad (17)$$

where $\phi_{\mathcal{R}}$ is a nonlinear function mapping inputs to $[0, 1]^2$, and is instantiated as multilayer perceptron (MLP) with Sigmoid as the activation function. Utilizing the readouts of module Φ , we can finally refine the messages by applying affine transformation:

$$P_S^i(t) = \tilde{P}_S^i(t) \cdot \xi^i(t) + \zeta^i(t), \quad (18)$$

$$P_S^{i \rightarrow k}(t) = \tilde{P}_S^{i \rightarrow k}(t) \cdot \xi^{i \rightarrow k}(t) + \zeta^{i \rightarrow k}(t), \quad (19)$$

The refined $P_S^i(t)$ and $P_S^{i \rightarrow k}(t)$ are fed back to DMP for the next iteration. The whole framework of NEDMP is visualized in Figure 3.

3.3 Training

We can obtain the *ground-truth* marginal probabilities $q^i(t) = [q_S^i(t), q_I^i(t), q_R^i(t)]$ by extensive Monte Carlo simulations. Therefore, the simplest way to optimize the proposed model NEDMP is to minimize the cross-entropy (*CE*) loss:

$$l(P, q) = \frac{1}{|\mathcal{V}|T} \sum_{i \in \mathcal{V}} \sum_{1 \leq t \leq T} CE(P^i(t), q^i(t)) \quad (20)$$

where T is the preset time length or the convergence time step, $P^i(t)$ is the predicted probabilities by model. It is worth noticing that the dynamics of SIR model ensure the monotone of node's marginals, i.e., $\forall t_1 \leq t_2, i \in \mathcal{V}: P_S^i(t_1) \geq P_S^i(t_2)$ and $P_R^i(t_1) \leq P_R^i(t_2)$, which the sole objective in Equation (20) may fail to capture. Thus, we add a regularization term to enforce this limitation:

$$Re = \sum_{i \in \mathcal{V}} \sum_{1 \leq t \leq T} [ReLU(P_S^i(t+1) - P_S^i(t)) + \text{relu}(P_R^i(t) - P_R^i(t+1))] \quad (21)$$

We combine the regularization to the cross-entropy loss with a factor λ as the final objective function:

$$L(P, q) = l(P, q) + \lambda \cdot Re \quad (22)$$

4 EXPERIMENTS AND RESULTS

We conduct experiments on a set of synthetic and real networks to evaluate the performance of the proposed model NEDMP with respect to the MPI problem of the SIR model. Specifically, in Section 4.1 we evaluate the performance of all methods on diverse sets of graphs and dynamics parameters, where the training and testing sets come from the same distribution. In Section 4.2, we evaluate the generalization ability of the proposed models on instances that have graph structure or dynamics parameters out of the training distribution.

Data Generation. For the training and testing in all experiments, we obtain the *ground-truth* of marginal probabilities by averaging over 10^5 Monte Carlo simulations.

Baselines. We compare three methods for the MPI problem of SIR model:

- **DMP** (A. Lokhov et al., 2014): An efficient inference algorithm for SIR model, and it is asymptotically exact on locally tree-like networks.
- **GNN** (Yoon et al., 2019): This work proposed two pure data-driven models for inferring the marginal probabilities for probabilistic graphical models. Considering the variant msg-GNN (similar to the model proposed in Section (3.1)) increases the computational consumption, with no improvement in accuracy, we adapt the variant node-GNN for our problem (details in Appendix).
- **NEDMP**: A hybrid model in which DMP and GNNs module runs jointly. The GNNs module is designed to improve the accuracy of messages in DMP by learning the high-order dependencies from simulation data.

Evaluation Metrics. Averaged L1 error is used as the performance metrics for all experiments:

$$L_1 = \frac{1}{|\mathcal{N}|T} \sum_{i \in \mathcal{N}} \sum_{1 \leq t \leq T} \|P^i(t) - q^i(t)\|_1 \quad (23)$$

Training Details. We optimize the proposed models with loss function in Eq.(22) with $\lambda = 5$. The maximum time step T is 30. We use the Adam optimizer (Kingma & Ba, 2015) with the learning rate 0.01 and batch size of 1. The learning rate reduces by a factor of 0.5 whenever the learning stagnates. We use early stopping with patience 15. The dimensions of all hidden states are $D = 32$. Each dataset in the following experiments is split by 6:2:2 for training, validation, and testing, respectively.

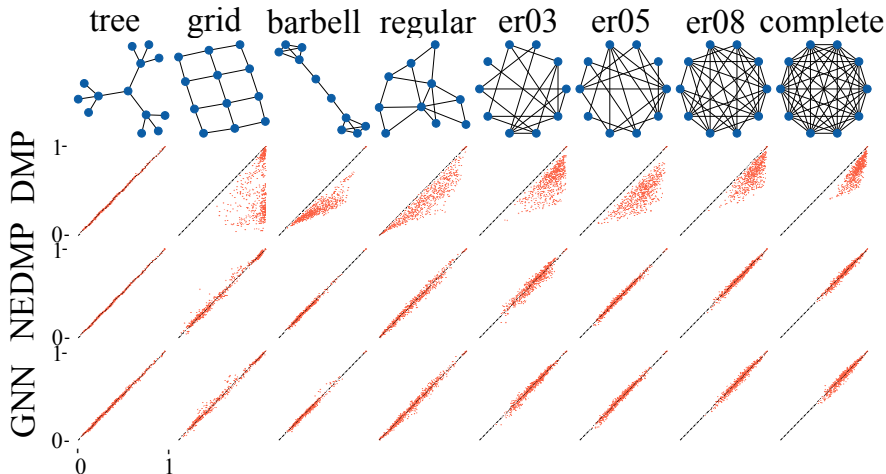


Figure 4: **Performance on diverse graph structure.** For each graph structure, the estimated marginals are plotted as horizontal coordinates while the vertical coordinates represent the ground truth. For visual simplicity, we only show the marginal probability $P_R^i(T)$ for each node i in testing graphs. The corresponding dot will lie on the diagonal line if it is accurately estimated.

4.1 Performance Within the Training Distribution

The complexity of MPI comes from two parts: the topology structure of the graph as well as the values of dynamics parameters, i.e., β , γ and \mathbb{S} . Therefore, it is necessary to train and test the methods over different graph structures and parameter ranges. Particularly, the training and testing set are generated within the same distribution of β , γ and \mathbb{S} . We also evaluate the methods on several real networks.

Graph Structures. As shown in Figure 4, we choose eight types of graph structure with the number of nodes $|\mathcal{N}| \approx 12$. For each structure, we generate 200 samples for training and testing. Each of the samples has one randomly chosen node as the initial infectious seed, and dynamics parameters from $\beta \sim \mathcal{U}(0.4, 0.6)$ and $\gamma \sim \mathcal{U}(0.2, 0.5)$. After training, we run each method on testing graphs, and plot the inferred marginal probabilities, paired with ground truth, at time step T for each method in Figure 4. The first row in Figure 4 verifies the inaccuracy of DMP in loopy graphs, as well as that DMP, is the upper bound of true influence (A. Y. Lokhov & Saad, 2019). The flexibility and powerful inference ability of both GNN and NEDMP are validated by the bottom two rows which show almost accurate inferences in all kinds of graphs.

Dynamics Parameters. With the structure fixed as the n -regular graph, we vary the dynamics parameters β , γ , and \mathbb{S} around the tipping point to evaluate the performance of proposed models, as shown in Figure

5. For each combination of parameters, we generate 200 samples for training and testing. When parameters approach the poles of the x-axis, the diffusion steps are too few to form loopy paths, resulting in trivial instances for MPI problem. This is why the error curve of DMP is bell-shaped. The two trained models GNN and NEDMP have similar performance, and both outperform DMP in all ranges of parameters. As parameters approach the area with loopier diffusion paths, the performances of the learning-based models degrade much slower than DMP.

Real Diffusion Networks. We also train and test all the methods on six true diffusion networks (Rossi & Ahmed, 2015), and all the graphs are preprocessed as undirected graphs. Again, we generate 200 samples for each graph as the training and testing set. For each sample, two randomly selected nodes are set as the initially infected nodes. And dynamics parameters are randomly sampled from $\beta \sim \mathcal{U}(0, 0.3)$ and $\gamma \sim \mathcal{U}(0.1, 0.4)$. Same as previous, within the distribution of training set, GNN and NEDMP have comparable performances, and all outperform DMP based on metrics L_1 .

4.2 Out of Distribution Generalization

It is well known that the neural network model suffers from poor generalization. Using *hand-engineering* and *end-to-end* learning cooperatively, a hybrid model which benefits from their complementary strengths can be one way to break the generalization limitation (Shlezinger, Whang, Eldar, & Dimakis, 2020). The proposed NEDMP is such a hybrid model while GNN

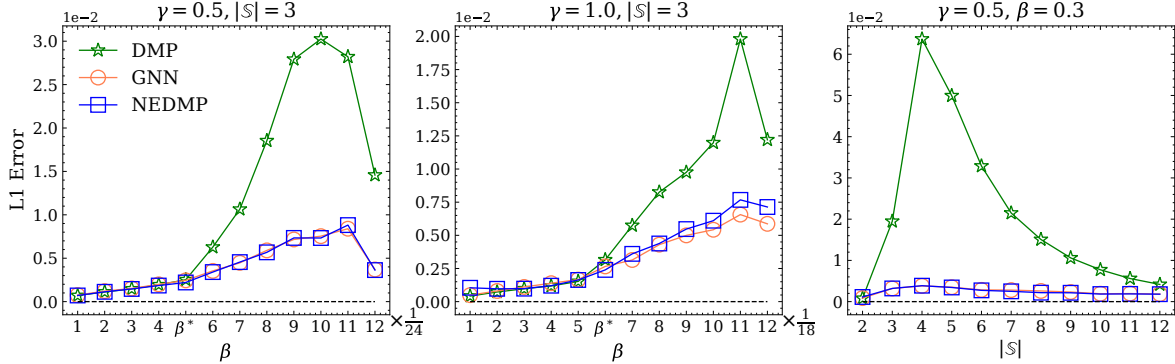


Figure 5: **Performance of various dynamics parameters.** In the **left** and **middle** figures, the graph structures are fixed to a 3-regular graph with $|\mathcal{N}| = 20$, and we increase the graph size to $|\mathcal{N}| = 100$ in the **right** figure. β^* in the **left** and **middle** figure is the tipping point of diffusion.

Table 1: **L_1 error on real networks (lower is better).**

	dolphins	fb-food	fb-social	norwegian	openflights	top-500
#Nodes	62	620	1899	1482	2939	500
#Edges	159	2102	20296	4006	15677	2980
DMP	0.089 ± 0.126	0.096 ± 0.177	0.023 ± 0.062	0.103 ± 0.199	0.077 ± 0.125	0.064 ± 0.089
GNN	0.028 ± 0.033	0.030 ± 0.039	0.021 ± 0.037	0.018 ± 0.042	0.032 ± 0.044	0.033 ± 0.048
NEDMP	0.027 ± 0.036	0.034 ± 0.044	0.024 ± 0.043	0.023 ± 0.054	0.048 ± 0.064	0.032 ± 0.044

is a purely data-driven model. To better understand the generalization ability of those two kinds of models, we conduct experiments on structure and parameter generalization.

Structure Generalization. We freeze the models trained on specific graph structures in Section 4.1, and then test them on all eight graph structure datasets. For instance, a GNN model trained on tree graphs is then tested on all graph structures, and the eight testing L_1 errors are visualized as the first row in Figure 7(a). All the off-diagonal cells in Figure 7(a,b) are the generalization errors for the corresponding model. The average of generalization errors are 0.128, 0.035 for GNN and NEDMP respectively. The superiority of NEDMP in generalization verifies the strengths of the hybrid model.

Dynamics Parameters Generalization. We consider the model generalization over β , γ and \mathcal{S} . We fix the graph structure as the Watts–Strogatz graph. As shown in Figure 6, for each parameter spectrum, we train the GNN and NEDMP on a small range of parameter values (bounded by the two vertical dashed lines). Then the trained models are tested on the whole parameter spectrum; the out-of-set results are presented in Figure 6. As expected, GNN degrades rapidly as the parameter away from the training set. Surprisingly, NEDMP can maintain almost the same performance outside the training distribution, and it even outperforms DMP under all unseen parameters.

NEDMP is so robust to parameters because the GNN module in NEDMP does not interact with parameters directly; parameters are encoded by well-established updating equations in the DMP module, which enables the hybrid model to generalize to unseen values.

5 DISCUSSION

In this work, we propose a hybrid model Neural Enhanced Dynamic Message Passing (NEDMP), which runs DMP and GNN jointly, for the marginal probabilities inference problem of the SIR model. We test the performance of inference in diverse sets of graph structures and dynamics parameters, and the results show that after training the proposed model significantly outperforms DMP in all kinds of graph structure within the training distribution. The generalization ability of the proposed model is evaluated by inferring on graph structures and parameters that are out of the training distribution. The results show that NEDMP generalizes much better than the pure data-driven model since it incorporates the informative dynamics-based prior bias from DMP module.

This work is also a demonstration of the emerging field of hybrid physics-guided machine learning (Rai & Sahu, 2020). We use physics priors (DMP) to guide the design of the neural networks and regularize the training process. In future work, incorporating the physics prior and graph neural networks more compactly can

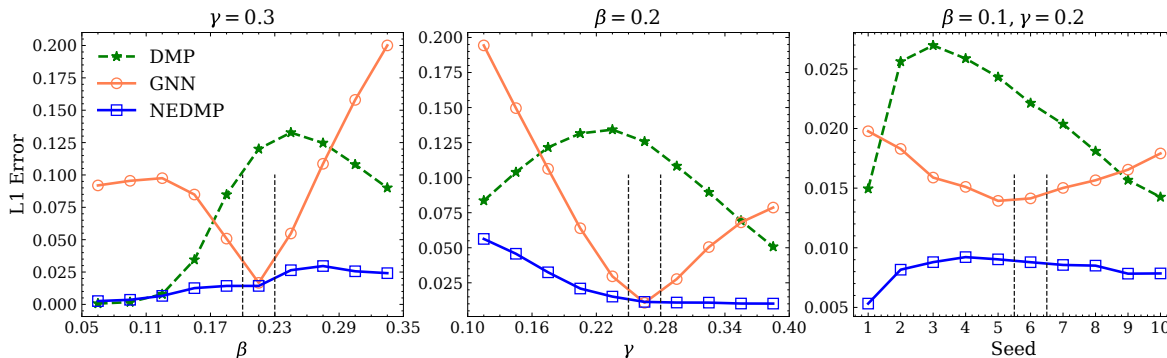


Figure 6: **Generalization Performance on Dynamics Parameters.** GNN and NEDMP are trained in a small range of parameter values, bounded by two vertical dashed lines, and then tested out-of-set. The graph structure is fixed to be Watts–Strogatz graph with 50 nodes, each node connects 5 nearest neighbors and the probability of rewiring each edge is $p = 0.2$.

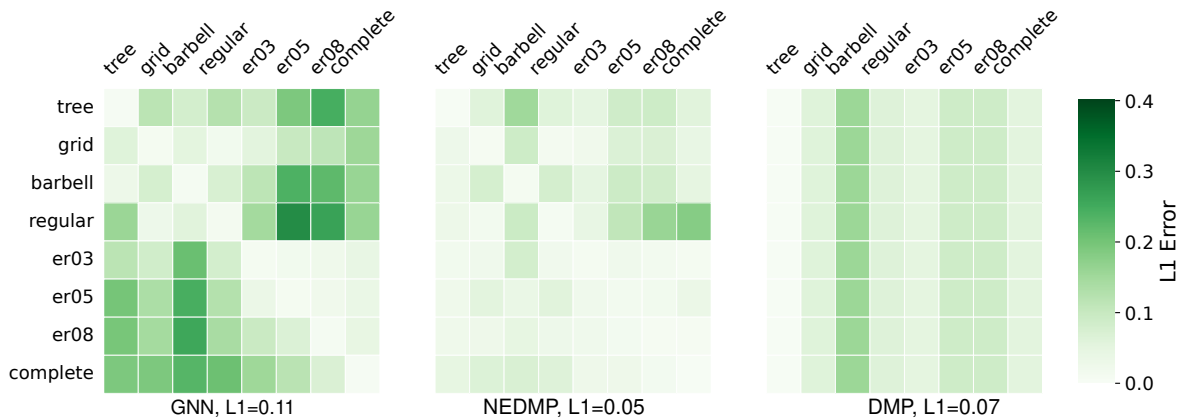


Figure 7: **Generalization on Graph Structure.** (a,b): GNN and NEDMP are trained on a specific graph structure and then test on all other graph structures. The diagonals are the training errors while the off-diagonal cells represent the generalization errors for each model. The average of all off-diagonal cells is labelled at the bottom of each heatmap.

be a promising direction.

As demonstrated in the experiments, NEDMP can be an efficient estimator for the spreading processing on graphs, which makes it potentially useful for forecasting and controlling epidemics (e.g., COVID-19) or rumor spreading on social networks. However, since NEDMP relies on the line graph of spreading networks, it is computationally hard to be applied to large social networks. Reducing the complexity of NEDMP is requisite for realistic scenarios, which is left for future work.

Acknowledgments

We are grateful for supporting from "Save 2050 Project" which is sponsored by Swarma Club and X-Order, and we also thank Muyun Mou and Jing Liu for technics supporting and insightful discussion.

References

- Bapst, V., Keck, T., Grabska-Barwinska, A., Donner, C., Cubuk, E. D., Schoenholz, S., ... Kohli, P. (2020). Unveiling the predictive power of static structure in glassy systems. *Nature Physics*, 16, 448-454. 1
- Cantwell, G. T., & Newman, M. (2019). Message passing on networks with loops. *Proceedings of the National Academy of Sciences*, 116, 23398 - 23403. 1, 2.2
- Cranmer, M., Sanchez-Gonzalez, A., Battaglia, P., Xu, R., Cranmer, K., Spergel, D., & Ho, S. (2020). Discovering symbolic models from deep learning with inductive biases. *ArXiv, abs/2006.11287*. 1
- Fetaya, T., Wang, E., Welling, K.-C., Zemel, M., Kipf,

- T., Fetaya, E., ... Zemel, R. (2018). Neural relational inference for interacting systems. *arXiv: Machine Learning*. 1
- Karrer, B., & Newman, M. (2010). Message passing approach for general epidemic models. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 82 1 Pt 2, 016101. 1
- Kempe, D., Kleinberg, J., & Tardos, É. (2015). Maximizing the spread of influence through a social network. *Theory Comput.*, 11, 105-147. 1, 2.1
- Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980. 4
- Kuck, J., Chakraborty, S., Tang, H., Luo, R., Song, J., Sabharwal, A., & Ermon, S. (2020). Belief propagation neural networks. *34th Conference on Neural Information Processing Systems (NeurIPS 2020)*. 1
- Li, S., Zhao, D., Wu, X., Tian, Z., Li, A., & Wang, Z. (2020). Functional immunization of networks based on message passing. *Appl. Math. Comput.*, 366. 1
- Li, Y., Tarlow, D., Brockschmidt, M., & Zemel, R. S. (2016). Gated graph sequence neural networks. *CoRR*, abs/1511.05493. 3.1
- Liang, M.-K., & Meyer, F. (2021). Neural enhanced belief propagation for cooperative localization. *ArXiv*, abs/2105.12903. 1
- Lokhov, A. (2016). Reconstructing parameters of spreading models from partial observations. In *Nips*. 1
- Lokhov, A., Mézard, M., Ohta, H., & Zdeborová, L. (2014). Inferring the origin of an epidemic with dynamic message-passing algorithm. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 90 1, 012801. 1, 4
- Lokhov, A., Mézard, M., & Zdeborová, L. (2015). Dynamic message-passing equations for models with unidirectional dynamics. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 91 1, 012811. 1, 2.2, A
- Lokhov, A., & Saad, D. (2017). Optimal deployment of resources for maximizing impact in spreading processes. *Proceedings of the National Academy of Sciences*, 114, E8138 - E8146. 1
- Lokhov, A. Y., & Saad, D. (2019). Scalable influence estimation without sampling. *arXiv preprint arXiv:1912.12749*. 2.2, 4.1
- Pastor-Satorras, R., & Vespignani, A. (2002). Immunization of complex networks. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 65 3 Pt 2A, 036104. 1
- Rai, R., & Sahu, C. K. (2020). Driven by data or derived through physics? a review of hybrid physics guided machine learning techniques with cyber-physical system (cps) focus. *IEEE Access*, 8, 71050-71073. 5
- Rossi, R. A., & Ahmed, N. K. (2015). The network data repository with interactive graph analytics and visualization. In *Proceedings of the twenty-ninth aaai conference on artificial intelligence*. Retrieved from <http://networkrepository.com> 4.1
- Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., & Battaglia, P. (2020). Learning to simulate complex physics with graph networks. *ArXiv*, abs/2002.09405. 1
- Satorras, V. G., & Welling, M. (2021). Neural enhanced belief propagation on factor graphs. In *Aistats*. 1, 3.2
- Shlezinger, N., Whang, J., Eldar, Y. C., & Dimakis, A. G. (2020). Model-based deep learning. *ArXiv*, abs/2012.08405. 4.2
- Shrestha, M., & Moore, C. (2014). A message-passing approach for threshold models of behavior in networks. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 89 2, 022805. 1
- Shrestha, M., Scarpino, S., & Moore, C. (2015). A message-passing approach for recurrent-state epidemic models on networks. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 92 2, 022821. 1
- Wang, C., Chen, W., & Wang, Y. (2012). Scalable influence maximization for independent cascade model in large-scale social networks. *Data Mining and Knowledge Discovery*, 25, 545-576. 2.1
- Wilinski, M., & Lokhov, A. (2021). Prediction-centric learning of independent cascade dynamics from partial observations. In *Icml*. 1
- Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P. S. (2019). A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32, 4-24. 3.2
- Yoon, K., Liao, R., Xiong, Y., Zhang, L., Fetaya, E., Urtasun, R., ... Pitkow, X. (2019). Inference in probabilistic graphical models by graph neural networks. *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, 868-875. 1, 4, B
- Zhu, K., & Ying, L. (2016). Information source detection in the sir model: A sample-path-based approach. *IEEE/ACM Transactions on Networking*, 24, 408-421. 1

Supplementary Material: Neural Enhanced Dynamic Message Passing

A Derivation of DMP Equations for SIR Model

We follow (A. Lokhov et al., 2015) to introduce the DMP equations for the SIR model as well as the physical sense of intermediate dynamic variables.

We start with the updated rules of $P_R^i(t)$ and $P_I^i(t)$, derived directly from SIR model:

$$P_R^i(t) = P_R^i(t-1) + \gamma_i P_I^i(t-1) \quad (24)$$

$$P_I^i(t) = 1 - P_R^i(t-1) - P_S^i(t-1) \quad (25)$$

Let $\theta^{j \rightarrow i}(t)$ be the probability that disease has not spread through the edge ($j \rightarrow i$) up to time t , and $P_S^i(t)$ is updated by Eq.(26) which is exact on the tree and approximate on general graphs :

$$P_S^i(t) = P_S^i(0) \prod_{j \in \mathcal{N}_i} \theta^{j \rightarrow i}(t) \quad (26)$$

Before deriving the close form of $\theta^{j \rightarrow i}(t)$, we introduce two useful intermediate dynamic variables:

- $P_S^{i \setminus k}(t)$: the probability that $\sigma_i^t = S$ when node i ignores all infection from its neighbor node k ;
- $\phi^{j \rightarrow i}(t)$: the probability that disease has not spread through the edge ($j \rightarrow i$) up to time t and node j is infected at time t (i.e., $\sigma_j^t = I$).

By excluding node k from \mathcal{N}_i in Eq.(26), we have

$$P_S^{i \setminus k}(t) = P_S^i(0) \prod_{j \in \mathcal{N}_i \setminus k} \theta^{j \rightarrow i}(t) \quad (27)$$

Utilizing variable $\phi^{j \rightarrow i}(t-1)$, we have the update rule for $\theta^{j \rightarrow i}(t)$:

$$\theta^{j \rightarrow i}(t) = \theta^{j \rightarrow i}(t-1) - \beta_{ji} \phi^{j \rightarrow i}(t-1) \quad (28)$$

The change of $\phi^{j \rightarrow i}(t-1)$ comes from two parts: (1) When node j is infected at time $t-1$, it becomes recovery with probability γ_j or infects node i with rate β_{ji} in the next time step; (2) When node j is susceptible at time $t-1$, it turns into infected with probability $P_S^{j \setminus i}(t-1) - P_S^{j \setminus i}(t)$. Therefore, we have the update rule for $\phi^{j \rightarrow i}(t)$:

$$\phi^{j \rightarrow i}(t) = (1 - \beta_{ji})(1 - \gamma_j) \phi^{j \rightarrow i}(t-1) + \left(P_S^{j \setminus i}(t-1) - P_S^{j \setminus i}(t) \right) \quad (29)$$

To complete the recursion updating rules, we give the initial values as:

$$\theta^{j \rightarrow i}(0) = 1, \phi^{j \rightarrow i}(0) = \delta_{\sigma_j^0, I} \quad (30)$$

B Baseline GNN

Graph Neural Networks (GNNs) model the pair-wise interactions by implementing a trainable recurrent message passing mechanism in the graph structure, and has three main steps: *Message Passing*, *Update* and *Readout*. After $t-1$ iterations of GNNs, let $m^u(t-1)$ as node u 's hidden states. In the next iteration, *Message Passing*

step firstly aggregates all information from u 's neighborhood \mathcal{N}_u as a message vector $m^{\rightarrow u}(t)$, and then node u updates its hidden status to $m^u(t)$ by combining $m^u(t-1)$ and $m^{\rightarrow u}(t)$ in the step *Update*. The updated hidden states are then fed into a task-specified *Readout* function \mathcal{R} for node-wise predictions in step t .

The marginal probabilities are dependent on the nodes initial infection status $S \in \{0, 1\}^{|\mathcal{V}|}$, nodes attributes $\gamma \in [0, 1]^{|\mathcal{V}|}$, as well as the edge attributes $\beta \in [0, 1]^{|\mathcal{E}|}$. Therefore, we first embed those attributes into vectors:

$$X_0 = \phi_n(S \oplus \gamma), \quad E_0 = \phi_e(\beta), \quad (31)$$

where \oplus is the concatenation operator, $\phi_n : R^2 \rightarrow R^D$, $\phi_e : R \rightarrow R^D$ and D is the preset number of hidden dimension, $X_0 \in R^{|\mathcal{V}| \times D}$, $E_0 \in R^{|\mathcal{E}| \times D}$. The nonlinear functions ϕ_n, ϕ_e are shared by all nodes and edges, respectively.

We present the details of proposed specific GNN for MPI as follows:

- **Step 1: Hidden status Initialization.** Unlike (Yoon et al., 2019) initializing hidden status as zeros, we initialize the hidden states from nodes attributes:

$$m^i(0) = \phi_{init}(X_0^i), \quad (32)$$

where ϕ_{init} is a nonlinear function mapping the inputs to R^D .

- **Step 2: Message Passing.** For node i in GNN, the incoming messages $m^{\rightarrow i}(t)$ are the summations of the adjacent hidden states. To enable the messages explicitly dependent on initial conditions, before summation, we concatenate the E_0, X_0 to the corresponding hidden states. Then we have:

$$m^{\rightarrow i}(t) = \phi_2 \left(\sum_{j \in \mathcal{N}_i} \phi_1 \left(m^j(t-1) \oplus E_0^{j \rightarrow i} \right) \right), \quad (33)$$

where ϕ_1 and ϕ_2 are nonlinear functions mapping the inputs to R^D .

- **Step 3: Update.** Before updating, we concatenate the incoming messages with the target-nodes initial attributes, which is an imitation of Eq.(27) in DMP, to filter the incoming messages. We then update nodes hidden status with a Gated Recurrent Unit (GRU):

$$m^i(t) = \text{GRU} \left(\phi_3 \left(m^{\rightarrow i}(t) \oplus X_0^i \right), m^i(t-1) \right), \quad (34)$$

and ϕ_3 is a trainable nonlinear function.

- **Step 4: Readout.** At each time step t (i.e. GNN t th layer), the predicted marginals for node i in GNN is computed by a Softmax function \mathcal{R} :

$$\hat{P}^i(t) = \mathcal{R} \left(\phi_4(m^i(t) \oplus X_0^i) \right) \quad (35)$$

where ϕ_4 is trainable nonlinear functions. The concatenation of hidden status and X_0^i aims to imitate Eq.(26) in DMP.

Repeat **Step 2-4** until termination condition is satisfied.

In our implementation, all the nonlinear functions $\phi_n, \phi_e, \phi_{init}, \phi_{1,2,3,4}$ are specified by MLP with ReLU as the activation function, and $\phi_{1,2,3,4}$ are shared across all time steps (GNN layers).

C The benefits of using the NEDMP in the within-distribution case.

Obtaining the ground truth as the training labels for large graphs is computationally expensive. This requires a sample-efficient model. Compared to the pure data-driven baseline GNN, the proposed model NEDMP is a physics-informed hybrid model, to which the DMP module provides strong regularization. Therefore, the NEDMP is much more sample-efficient than the baseline. For two real networks, we increase the number of training samples from 3 to 52, and the validation error of NEDMP and GNN is plotted in Figures 8 and 9. The NEDMP can achieve satisfactory error for both networks with limited training samples, while the GNN requires more training data. Therefore, using NEDMP is a better choice when obtaining the training data is expensive.

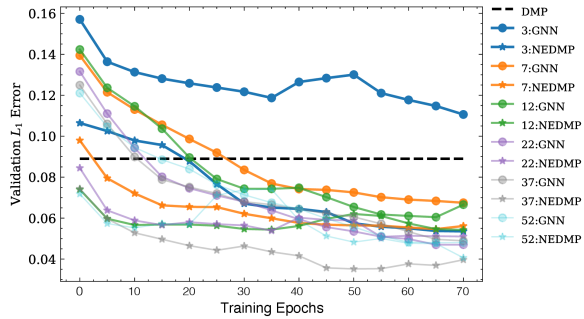


Figure 8: Val. error for network dolphins

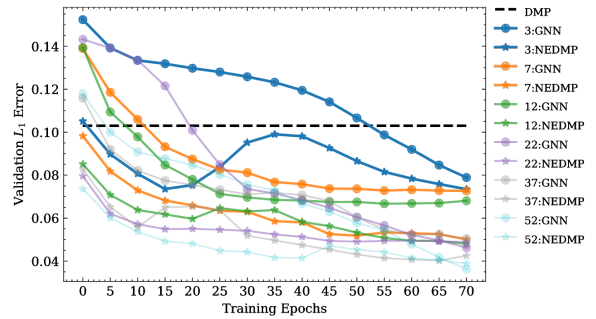


Figure 9: Val. error for network norwegian

D Implementation Details

We implement our model utilizing Pytorch¹ and Pytorch-Geometric. We train and test our model on Ubuntu 18.04.5 with Intel Xeon Gold 6248 CPU and NVIDIA Tesla V100 GPU.

¹<https://github.com/pytorch/>