Journal of the American Statistical Association

ISSN: (Print) (Online) Journal homepage: www.tandfonline.com/journals/uasa20

A Model-Agnostic Graph Neural Network for Integrating Local and Global Information

Wenzhuo Zhou, Annie Qu, Keiland W. Cooper, Norbert Fortin & Babak Shahbaba

To cite this article: Wenzhuo Zhou, Annie Qu, Keiland W. Cooper, Norbert Fortin & Babak Shahbaba (15 Nov 2024): A Model-Agnostic Graph Neural Network for Integrating Local and Global Information, Journal of the American Statistical Association, DOI: 10.1080/01621459.2024.2404668

To link to this article: https://doi.org/10.1080/01621459.2024.2404668

JOURNAL OF THE AMERICAN STATISTICAL ASSOCIATION
2024, VOL. 00, NO. 0, 1–14: Theory and Methods
https://doi.org/10.1080/01621459.2024.2404668

A Model-Agnostic Graph Neural Network for Integrating Local and Global Information

Wenzhuo Zhou[a], Annie Qu[a], Keiland W. Cooper[b], Norbert Fortin[b], and Babak Shahbaba[a]

[a]Department of Statistics, University of California Irvine, Irvine, CA; [b]Department of Neurobiology and Behavior, University of California Irvine, Irvine, CA

ABSTRACT

Graph Neural Networks (GNNs) have achieved promising performance in a variety of graph-focused tasks. Despite their success, however, existing GNNs suffer from two significant limitations: a lack of interpretability in their results due to their black-box nature, and an inability to learn representations of varying orders. To tackle these issues, we propose a novel Model-agnostic Graph Neural Network (MaGNet) framework, which is able to effectively integrate information of various orders, extract knowledge from high-order neighbors, and provide meaningful and interpretable results by identifying influential compact graph structures. In particular, MaGNet consists of two components: an estimation model for the latent representation of complex relationships under graph topology, and an interpretation model that identifies influential nodes, edges, and node features. Theoretically, we establish the generalization error bound for MaGNet via empirical Rademacher complexity, and demonstrate its power to represent layer-wise neighborhood mixing. We conduct comprehensive numerical studies using simulated data to demonstrate the superior performance of MaGNet in comparison to several state-of-the-art alternatives. Furthermore, we apply MaGNet to a real-world case study aimed at extracting task-critical information from brain activity data, thereby highlighting its effectiveness in advancing scientific research. Supplementary materials for this article are available online, including a standardized description of the materials available for reproducing the work.

ARTICLE HISTORY
Received August 2023
Accepted September 2024

KEYWORDS
Empirical Rademacher complexity; Graph representation; Information aggregation

1. Introduction

Graph-structured data is ubiquitous throughout the natural and social sciences, from brain networks to social relationships. A graph is simply a collection of nodes representing entities such as people, genes, and brain regions, along with a set of edges representing interactions between pairs of nodes. By representing such interconnected entities as graphs, it is feasible to leverage their geometric topology to study statistical relationships among nodes using network-based frameworks. Among graph representation methods, the family of graph neural networks (GNNs) has achieved remarkable success in real-world graph-based tasks (Veličković 2023). In general, GNNs iteratively aggregate and combine node representations within a graph, through a process called message passing, to generate a set of learned hidden representation features. The main neural architectures of GNNs include graph convolutional networks (GCNs; Kipf and Welling 2016), graph attention networks (GATs; Veličković et al. 2017), graph transformer networks (GTNs; Yun et al. 2019), among many other variants.

While GNNs are capable of capturing subgraph information through message passing, they can be prone to over-smoothing the learned representations when applying multiple rounds of message passing operations. This can cause models to treat all nodes uniformly, leading to node representations that converge into indistinguishable entities (Li, Han, and Wu 2018). Additionally, over-smoothing could limit the ability to capture high-order information, which can be only aggregated through a sufficient number of message passing operations (Hamilton 2020). Several studies suggest that over-smoothing significantly contributes to deep GNN performance degradation (Bodnar et al. 2022). To address this issue, (Zhang et al. 2022) advocated for using shallow GNNs (e.g., up to three layers). However, this approach fails to capture high-order information due to insufficient message passing. Additionally, Zhao and Akoglu (2019) and Yang et al. (2020) developed normalization layers to prevent node embeddings from becoming indistinguishable, but this increases training difficulty and limits the expressive power of GNNs.

A main premise of this article is that to enhance the overall representation power of GNNs with statistical guarantees, we need to develop new learning mechanisms that directly incorporate and effectively combine information from neighbors at different orders. Statistically, by integrating both low-order (i.e., immediate neighbors) and high-order (i.e., neighbors beyond the immediate vicinity) information, GNNs can learn a richer and more complete representation under the graph topology. Models that follow the principle of effectively combining information from different orders are known as multi-scale GNNs, as they enable the exploration and integration of information at different levels of granularity within a graph (Xu et al. 2018; Sun, Zhu, and Lin 2019; Oono and Suzuki 2020; Liu et al. 2022). The main idea is to direct the outputs of intermediate layers to contribute to the final representation. Existing methods however struggle to effectively integrate representations of different orders in a sequential manner due to their memo-

CONTACT Babak Shahbaba babaks@uci.edu ISEB 2222, Department of Statistics, University of California Irvine, Irvine, CA.
Supplementary materials for this article are available online. Please go to www.tandfonline.com/r/JASA.

ryless property: at each GNN layer, node representations are updated entirely based on the current input from their immediate neighbors, without directly retaining information from previous layers. Furthermore, Multi-scale GNNs still suffer from over-smoothing issues and generally fail to capture high-order latent representations.

To address the aforementioned issues, we develop a novel Model-agnostic Graph neural Network (MaGNet) framework consisting of two components: the *estimation model* and the *interpretation model*. The estimation model captures the complex relationship between the feature information and a target outcome under graph topology, allowing for powerful latent representation corresponding to unique-order graph information. The interpretation model, on the other hand, identifies a compact subgraph structure—specifically, influential nodes and edges—along with a small subset of node features that play a crucial role in the learned estimation model. The main advantages of the proposed framework, along with our contributions, are outlined below. First, the proposed neural architecture of the model alleviates the over-smoothing issue and thus can effectively extract knowledge from high-order neighbors. The proposed *actor-critic* neural architecture effectively integrates multi-order information by resolving the memoryless issue. It adaptively combines representations from actor graph neural networks, each focused on a specific order, while a critic network evaluates the quality of the learned representations. Second, we develop an interpretation framework, formulated as an optimization task that maximizes information gain over the distribution of possible subgraph structures. This approach is model-agnostic as there is no assumption on the true statistical models or data-generating mechanisms. Third, we study the ability to integrate various-order information as well as the statistical complexity of the proposed model via an empirical Rademacher complexity. Unlike existing analyses limited to standard message passing neural networks, our results are applicable to a mix of message passing and feedforward neural networks, with message passing networks being a special case in our framework. Furthermore, we provide a statistical generalization error bound for the MaGNet estimation model that consists of sequential deep-learning component models.

2. Preliminaries

2.1. Graph Structure

In this section, we present some preliminaries and notations used throughout the article. Let $G = (V, E)$ represent the graph, where V represents the vertex set consisting of nodes $\{v_1, v_2, \ldots, v_N\}$, and $E \in V \times V$ denotes the edge set with (i, j)th element e_{ij}. The number of total nodes in the graph is denoted by N. A graph can be described by a symmetric (typically sparse) adjacency matrix $A \in \{0, 1\}^{N\times N}$ derived from V and E. In this setting, $a_{ij} = 0$ indicates that the edge e_{ij} is missing, whereas $a_{ij} = 1$ indicates that the corresponding edge exists. There is a T-dimensional set of features, X_i, associated with each node, v_i so that the entire feature set is denoted as $X \in \mathbb{R}^{N\times T}$. Suppose we have observed n graph instances, each consisting of a fixed graph structure but with different node features. Let G_i denote the ith instance of a graph, where $i \in 1, 2, \ldots, n$. While our approach can be used for predictive models in general, here we focus on classification problems, where the objective is to assign a binary label $s \in \{-1, 1\}$ to each graph instance.

2.2. Neural Message Passing

The basic graph neural network (GNN) model can be motivated in a variety of ways. The same fundamental GNN model has been derived as a generalization of convolutions to non-Euclidean data (Bodnar et al. 2022), and as a differentiable variant of belief propagation (Dabkowski and Gal 2017), as well as by analogy to classic graph isomorphism tests (Graham, Wang, and Ravanbakhsh 2019). Regardless of the motivation, the defining feature of a GNN is that it uses a form of neural message passing in which vector messages are exchanged between nodes and updated using neural networks (Abu-El-Haija et al. 2019). During each round of message passing in a GNN, a hidden embedding corresponding to each node $v \in \mathcal{V}$, denoted as $H_v^{(k)}$ for the kth layer where $k = 1, \ldots, K$, is updated according to information aggregated from v's graph neighborhood $\mathcal{N}(v)$. This message passing update can be expressed as follows:

$$H_v^{(k+1)} = f_{\text{update}}{}^{(k)}\left(H_v^{(k)}, f_{\text{agg}}{}^{(k)}\left(\left\{H_u^{(k)}, \forall u \in \mathcal{N}(v)\right\}\right)\right) = f_{\text{update}}{}^{(k)}\left(H_v^{(k)}, M_{\mathcal{N}(v)}^{(k)}\right),$$

where f_{update} and f_{agg} are the update and aggregate functions, which are arbitrary differentiable functions (here, neural networks). The term $M_{\mathcal{N}(v)}$ is the "message" that is aggregated from v's graph neighborhood $\mathcal{N}(v)$. We use superscripts to distinguish the embeddings and functions at different rounds of message passing. At each round of message passing, the aggregate function takes as input the set of embeddings of the nodes in v's graph neighborhood $\mathcal{N}(v)$ and generates a message $M_{\mathcal{N}(v)}^{(k)}$ based on this aggregated neighborhood information. The update function then combines the message $M_{\mathcal{N}(v)}^{(k)}$ with the previous embedding $H_v^{(k-1)}$ of node v to generate the updated embedding $H_v^{(k)}$.

2.3. Graph Convolutional Network (GCN)

Let $\tilde{D}$ be the degree matrix corresponding to the augmented adjacency matrix $\tilde{A} = A + I$ with $\tilde{D}_{ii} = \sum_{j=1}^{N} \tilde{A}_{ij}$. The hidden graph representation of nodes with two graph convolutional layers (Kipf and Welling 2016) can be formulated in a matrix form:

$$H = \widetilde{\mathcal{L}}\,\text{ReLU}(\widetilde{\mathcal{L}}XW^{(0)})W^{(1)}, \tag{1}$$

where $H \in \mathbb{R}^{N\times T^{(1)}}$ is the final embedding matrix of nodes and $T^{(1)}$ is the dimension of the node hidden representation (embedding). The graph Laplacian is defined as $\widetilde{\mathcal{L}} = D^{-\frac{1}{2}}\tilde{A}D^{-\frac{1}{2}}$. In addition, the weight matrix $W^{(0)} \in \mathbb{R}^{T\times T^{(0)}}$ is the input-to-hidden weight matrix for a hidden layer with $T^{(0)}$ feature maps, and $W^{(1)} \in \mathbb{R}^{T^{(0)}\times T^{(1)}}$ is the hidden-to-output weight matrix. Here we consider the two-layer case that aims to simplify the

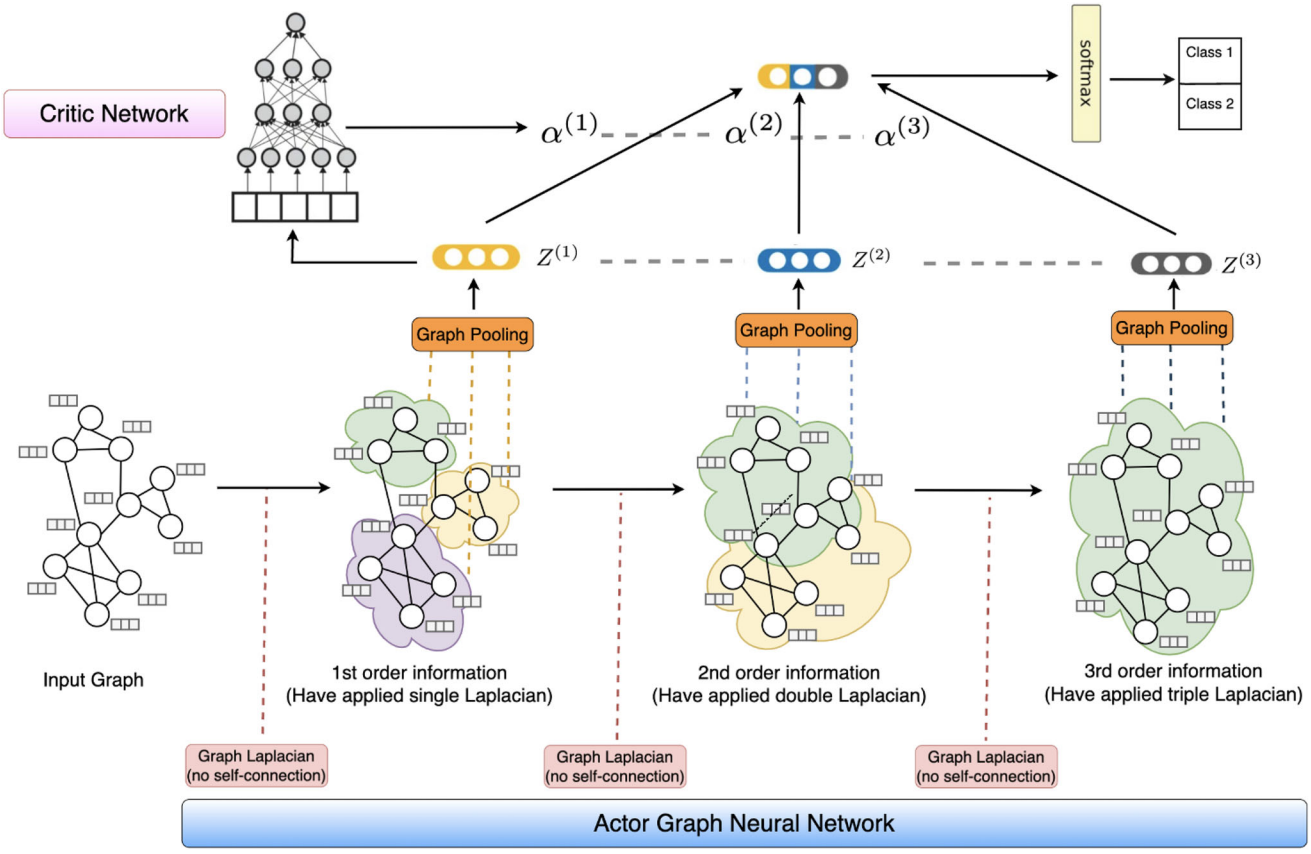


Figure 1. An illustrative example of three-layers neural architecture of the actor-critic graph neural network.

notation; the above definition can be easily extended to k graph convolutional layers with $k > 2$.

3. Estimation Model

In this section, we introduce a novel graph neural network, which aims to represent feature information and capture it relationship to an outcome of interest. To achieve this goal, it effectively integrates both low-order and high-order neighbor node information to form a powerful latent representation. Here, the low-order information refers to information aggregated from the local neighbors of a node, while the high-order information means the messages aggregated beyond just the immediate/close neighbors, capturing the global graph information.

To characterize the feasibility and effectiveness of capturing various-order information rigorously, we first introduce a generalized 2-order Δ-representer, which is defined by Abu-El-Haija et al. (2019), that is, the corresponding K-order counterpart for $K \geq 3$ as follows.

Definition 3.1. Given a graph neural network, $\Delta(K)$-representer represents K-order node neighbor information for $K \in \mathbb{N}$, for example, there exists a real-valued vector $\nu = (\nu_1, \nu_2, \ldots, \nu_K)$ and an injective (one-to-one) mapping function $g(\cdot)$, such that the output embedding of this graph neural network can be represented as

$$g\left(\sum_{k=0}^{K} \nu_k \cdot \mathcal{L}^k X\right) := \Delta(K),$$

for any type of graph Laplacian $\mathcal{L}$ operation and input node feature matrix X.

Learning such a representer enables GNNs to capture feature differences among K-order node neighbor's information. When a candidate GNN model learns the $\Delta(K)$-representer, it effectively captures the K-order neighborhood information in the hidden representation.

In GCNs, the graph representation is obtained through interactions of neighboring nodes during multiple rounds of learned message passing. Ideally, one could consider a deep architecture via stacking K GCN layers in order to learn a $\Delta(K)$-representer. However, most of the existing GCN models employ shallow architectures, typically using only second- or third-order information (Zhang, Cui, and Zhu 2020). The reason behind this limitation is 2-fold. First, when repeatedly applying Laplacian smoothing, GCNs may mix node features from different clusters, rendering them indistinguishable. This phenomenon is known as the *over-smoothing* issue (Li et al. 2021). Second, most GCNs are built upon a feedforward mechanism and suffer from the *memoryless* problem. After each layer operation, the representation learned from the current layer modifies the representation produced from the previous layers. As a result, there is no explicit memory mechanism. In other words, the *over-smoothing* issue creates difficulties in capturing high-order information, while the *memoryless* issue leads to a loss of lower-order information. Theoretically, under Definition 3.1, the GCN models with over-smoothing or memoryless issues cannot learn

the $\Delta(K)$-representer. A more detailed discussion is provided in the Appendix.

3.1. Actor-Critic Graph Neural Network

In this section, we describe our actor-critic graph neural network (Figure 1), which is designed to effectively aggregate different levels of node-neighbor information to obtain a powerful graph embedding. In this dual neural network structure, the actor graph neural network aims to capture the hidden representation for each order of node-neighbor information, while the critic neural network plays the role of evaluating the quality of the hidden representation learned by the actor network. To this end, we perform a fusion operation to integrate the representations from individual actor networks using the corresponding quality scores as weights. This framework resolves the over-smoothing and memoryless issues, ensuring a properly learned $\Delta(K)$-representer.

In contrast to GCNs, we adopt the simple weighted sum aggregator and abandon the nonlinear transformation. As a result, the graph convolution operation in our actor graph neural network is defined as

$$H^{(k)} = (\mathcal{L})^k XW, \tag{2}$$

where the graph Laplacian $\mathcal{L} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$, and W is the weight matrix, which can be fixed as identity matrix in every message passing round. We argue that the majority of the benefit arises from the local averaging of neighboring features. This is because unlike multi-dimensional image data, the vectorized temporal signal does not require many nonlinear layers to capture the information. Furthermore, the nonlinear feature transformation in GNNs is useful but not critical (Wu et al. 2019). By removing the nonlinear activation, our graph convolution layer achieves slower convergence in certain values for embedding vectors, thus, alleviating the over-smoothing issue. Additionally, abandoning the nonlinear feature transformation operation greatly improves computation.

It is also worth noting that in (2), we aggregate only the connected neighbors without integrating the target node itself. That is, the graph Laplacian is based on the adjacency matrix A instead of its augmented counterpart $\tilde{A}$. The fusion operation in our model, to be discussed later, essentially captures a similar effect as "self-connection" with adaptivity. This *partial self-connection* mitigates over-smoothing issues. In the Appendix, we provide a discussion on this theoretical investigation of the partial self-connection in Theorem S.1. Notably, this distinguishes our model from the closely related GNN works, for example, (Kipf and Welling 2016; Sun, Zhu, and Lin 2019; Wu et al. 2021), that aggregate extended neighbors and need to handle the self-connection explicitly without flexibility.

Further, in contrast to node classification, our interest lies in graph classification tasks. Therefore, based on the learned node representation $H^{(k)}$, we can apply a graph pooling operation to summarize the graph embedding from $H^{(k)}$. The goal of the graph pooling operation is to aggregate information across the entire graph to produce a single, fixed-size representation. Specifically, we could use a simple average graph pooling operation to obtain graph embeddings with a compact node representation: $\bar{H}^{(k)} = \frac{1}{N}\mathbf{1}_N H^{(k)}$. See the Appendix for other types of graph pooling operation, and Hamilton (2020) for a more comprehensive review.

As previously stated, our primary objective is to aggregate mixed-order information. To this end, we propose a fusion operation to completely preserve the information from various-order neighbors. Specifically, we can regard the graph embedding $\bar{H}^{(k)}$ from kth round of message passing as the output of the lth order information summarization, denoted as $\bar{H}^{(k)}$. From the perspective of meta algorithms, the embedding $\bar{H}^{(k)}$ can be regarded as the graph embedding learned from the kth actor graph neural network. Then the ultimate graph embedding can be weighted combined from the various order graph embeddings, that is,

$$\widetilde{H} = \sum_{k=1}^{K} \alpha^{(k)} \bar{H}^{(k)}, \tag{3}$$

where $\alpha^{(k)}$ is the fusion weight corresponding to the quality (or importance) of the kth order knowledge for $k = 1, \ldots, K$. This fusion operation can be understood as the ensemble of multiple single actor networks. that is, the actor networks for generating graph embedding $\bar{H}^{(k)}$. Thus, the fusion operation naturally combines the unique characteristics of different single learners with different order information. Moreover, as we discussed previously, the fusion operation intrinsically captures the so-called partial self-connection effect and thus relaxes over-smoothing issues. Theorem S.2 in the Appendix provides another view of the partial self-connection effect from the perspective of the graph spectral analysis. It shows that our model is more capable of mitigating the over-smoothing issue in comparison to GCNs.

To determine the fusion weights $\alpha^{(k)}$, we introduce a critic network, $f_{\text{critic}} : \bar{H}^{(k)} \mapsto \Delta_{\{-1,1\}}$ for a probability simplex $\Delta_{\{-1,1\}}$, which takes the graph embedding vector as input and output the binary probability logits. Generally, this critic network could be any proper parametric or nonparametric classification model including softmax (logistic) regression, random forest, feedforward neural network (multilayer perception, MLP), and so on. Without loss of generality, we choose the critic network as a class of the softmax regression in the following for illustration purposes.

The critic network plays a role in evaluating the quality of the graph embedding $\bar{H}^{(k)}$ in a bias induction way. That is

$$\alpha^{(k)} = \frac{1}{2}\log\left(\frac{1-\epsilon^{(k)}}{\epsilon^{(k)}}\right),$$

where the function $\text{logit}(x) = \log(x/(1-x))$ and the error rate $\epsilon^{(k)}$ is defined as

$$\epsilon^{(k)} = \sum_{i=1}^{n} \beta_i^{(k)} \mathbb{1}\left\{ s_i \neq \underset{\{s=-1,1\}}{\arg\max}\, \text{softmax}(\bar{H}_i^{(k)}) \right\} / \sum_{i=1}^{n} \beta_i^{(k)}. \tag{4}$$

Here $\bar{H}_i^{(k)}$ denotes the graph embedding for the ith graph sample, $\arg\max_{\{s=-1,1\}}(x)$ is the operator taking the maximum element in the two-dimensional vector x, and the coefficient $\beta_i^{(k)}$ denotes the weight of the ith graph sample. Intuitively, $\epsilon^{(k)}$ can be understood as the weighted classification error rate for the kth critic network, that is $\text{softmax}(\bar{H}^{(k)})$. For $i = 1, \ldots n$, we adjust the graph sample weights $\beta_i^{(k)}$ sequentially from

the kth to the $(k+1)$th step by following the updating rule: $\beta_i^{(k+1)} \propto \beta_i^{(k)} e^{\mathbb{1}\{s_i \neq \arg\max_{\{s=-1,1\}} \text{softmax}(\bar{H}_i^{(k)})\} \cdot \alpha^{(k)}}$. This update rule intentionally pays more attention to the misclassified graph samples with potentially insufficient representation power and increases their weights when training the next single actor network. Notably, instead of enhancing the nonlinear representation for embedding vectors, for example, $\bar{H}_i^{(k)}$ during the phase of the fusion, we might opt to perform nonlinear transformation on the graph embedding $\widetilde{H}$. This helps to relax the training difficulties well, in contrast to the existing (Sun, Zhu, and Lin 2019; Ivanov and Prokhorenkova 2021) where the nonlinear transformation is performed layer-wisely.

The embedding of the ultimate graph $\widetilde{H}$ in (3) combines information adaptively from the 1st to the Kth order node neighbors, using adaptive weights. Furthermore, the sequential updates of single actor networks maintain the same learning pattern as standard GCNs, wherein the message passing for the $(k+1)$th hop directly succeeds the message passing for the kth hop. In this manner, our actor-critic graph neural network maintains most of the desirable properties found in standard GCNs, including invariance to graph isomorphism (Xu et al. 2018a) and effective relational representation (Wu et al. 2020).

Ultimately, we establish the classification model and the rule for prediction based on the graph embedding $\widetilde{H}_i$ corresponding to the ith graph sample,

$$p(\cdot|\widetilde{H}_i) = \text{softmax}(g(\widetilde{H}_i)). \tag{5}$$

where $g(\cdot)$ is an arbitrary function to perform the linear/nonlinear (or even identity) transformation mapping. The equation returns the classification logits. To streamline the notation throughout the article, we use $p_{\widehat{\theta}}(\cdot)$ to represent a trained actor-critic graph neural network model. To have a better understanding of the proposed actor-critic neural network.

Finally, we note that training the model in (5) is a well-studied convex optimization problem. It can be performed using efficient second-order methods or stochastic gradient descent (SGD) (Bottou 2010). As long as the graph connectivity pattern remains sufficiently sparse, SGD can naturally scale to handle very large graph sizes. Furthermore, we ensure the neural network architecture's consistency by training the layers sequentially, following the order of node neighbors' message passing. This sequential training approach enables us to use the trained parameters from the previous training to initialize the current model training, which effectively reduces computational costs.

4. Interpretation Model

Although the estimation model provides strong representation power to capture the complex relationship between the outcome of interests and features, understanding the rationale behind its predictions can be quite challenging. In this section, we present a practically useful interpretation framework designed to uncover the reasoning behind the "black-box" estimation model.

To bridge the gap between estimation and interpretation, we first observe that our estimation model extracts feature information from various-order node neighbors as well as graph topology to output the hidden graph representation for predictions. This suggests that the prediction made by the estimation model, that is, $\widehat{s} = \arg\max_{\{s=-1,1\}} p_{\widehat{\theta}}(\cdot)$ in (5), is determined by the adjacency matrix A and the node feature information X. Formally, to comprehend the model mechanism and provide explanations, the problem is transformed into identifying important subgraphs, denoted as $G_{sub} \subseteq G$ with a corresponding adjacency matrix A_{sub}, along with a small subset of the node feature X_{sub} in full dimension. We first focus on the identification of influential subgraphs by assuming X_{sub} has been obtained and then discuss how to perform node feature selection simultaneously with subgraph identifications.

We adapt the principle of information gain, which was first introduced in the context of decision trees (Larose and Larose 2014), into our framework. In particular, we formulate an optimization framework for influential subgraph identification. Our goal is to maximize the information gain with respect to subgraph candidates G_{sub}:

$$\underset{G_{sub}}{\text{argmax}}\, \text{IG}\left(p_{\widehat{\theta}}, G_{sub}\right) = \eta(p_{\widehat{\theta}}) - \eta\left(p_{\widehat{\theta}} \mid G_{sub}, X_{sub}\right), \tag{6}$$

where $\eta(\cdot)$ and $\eta(\cdot|\cdot)$ denote the entropy and conditional entropy, respectively.

Essentially, information gain can quantify the change in prediction probability between the full model $p_{\widehat{\theta}}(\cdot)$ and the one constrained to the subgraph G_{sub} and the subset node feature X_{sub} Ying et al. (2019). For example, if removing edge e_{ij}, that is, the (i,j)th element in the adjacency matrix A, from the full graph G significantly decreases the prediction probability, then this edge is influential and should be included in the subgraph G_{sub}. Conversely, if the edge e_{ij} is deemed redundant for prediction by the learned estimation model, it should be excluded.

Examining the right-hand side of (6), we can easily observe that the entropy term $\eta(p_{\widehat{\theta}})$ remains constant since the parameters $\widehat{\theta}$ are fixed for an estimated model. Consequently, the objective of minimizing information gain in (6) is equivalent to maximizing the conditional entropy $\eta\left(p_{\widehat{\theta}} \mid G_{sub}, X_{sub}\right)$. Nevertheless, directly optimizing the above objective function is intractable, as there are $2^{|V|}$ candidates for the subgraph G_{sub}. To address this issue, we consider a relaxation by assuming that the subgraph is a Gilbert random graph (Reitzner, Schulte, and Thäle 2017). This way, the selection of edges from the original input graph G are conditionally independent of each other and follow a probability distribution. In detail, the edge e_{ij} is a binary variable indicating whether the edge is selected, with $e_{ij} = 1$ if selected and 0 otherwise. Therefore, the graph G_{sub} is a random graph with probability $P(G_{sub}) = \Pi_{i,j \in N} P(e_{ij})$. A straightforward instantiation of $P(e_{ij})$ is the Bernoulli distribution $e_{ij} \sim \text{Bern}(\mu_{ij})$, where μ_{ij} is the first moment. In particular, we can rewrite the parameterized objective as

$$\begin{aligned} &\underset{G_{sub}}{\text{Minimize}}\ \eta\left(p_{\widehat{\theta}} \mid G_{sub}, X_{sub}\right) \\ &\quad = \underset{G_{sub}(\mu)}{\text{Minimize}}\ \mathbb{E}_{G_{sub}(\mu)}[\eta\left(p_{\widehat{\theta}} \mid G_{sub}, X_{sub}\right)], \end{aligned} \tag{7}$$

where $G_{sub}(\mu)$ is the parametrized random subgraph. Due to the discrete nature of the subgraph $G_{sub}(\mu)$, the objective function is non-smooth, making optimization challenging and unstable. To address this issue, we further leverage a continuous approximation for the binary sampling process (Maddison, Mnih, and Teh 2016; Luo et al. 2020). Let ϵ be a uniform random variable, that

is, $\epsilon \sim \text{Unif}(0,1)$, and the real-valued parameters $\psi_{ij} \in \Psi$, and a temperature parameter $\omega \in \mathbb{R}^+$, then a sample of the binary edge e_{ij} can be approximated by a sigmoid mapping:

$$\widetilde{e}_{ij} = \text{sigmoid}\left(\frac{\log(\epsilon) - \log(1-\epsilon) + \psi_{ij}}{\omega}\right).$$

We denote $\widetilde{G}_{sub}(\Psi)$ as the continuous relaxation counterpart of the subgraph, with the (i,j)th element of the adjacency matrix being $\widetilde{e}_{ij}$. Interestingly, the temperature parameter ω can describe the relationship between $\widetilde{G}_{sub}(\Psi)$ and $G_{sub}(\mu)$. We observe that as $\omega \to 0$, the approximated edge $\widetilde{e}_{ij}$ converges to the edge e_{ij}, with the probability mass function, $\lim_{\omega\to 0} P(\widetilde{e}_{ij} = 1) = \frac{\exp(\psi_{ij})}{1+\exp(\psi_{ij})}$. Recall that the edge e_{ij} follows a Bernoulli distribution with mean μ_{ij}. If we reparameterize ψ_{ij} such that $\psi_{ij} = \log\left(\frac{\mu_{ij}}{1-\mu_{ij}}\right)$, it achieves asymptotical consistency of the approximated subgraph by following the limiting theory (Paulus et al. 2020), that is, $\lim_{\omega\to 0} \widetilde{G}_{sub}(\Psi) = G_{sub}(\mu)$. This supports the feasibility of applying continuous relaxation to the binary distribution.

Unlike the objective function in (7), which is induced by the discrete original subgraph, the objective function becomes smooth under the edge continuous approximation and can be easily optimized using gradient-based methods. In other words, the gradient of the continuous edge approximation $\widetilde{e}_{ij}$ with respect to the parameters ψ_{ij} is computable. More importantly, the sampling randomness toward the subgraph is absorbed into a uniform random variable ϵ peel-off from the parameterized binary Bernoulli distribution, which greatly relaxes the complexity of the sampling processing.

In this manner, the objective function in (7) can be reformulated as

$$\underset{\Psi}{\text{Minimize}}\ \mathbb{E}_{\epsilon\sim\text{Unif}(0,1)}[\eta\left(p_{\widehat{\theta}} \mid G_{sub}(\Psi), X_{sub}\right)].$$

However, solving the conditional entropy is still computationally expensive. To avoid this issue, we follow Kipf et al. (2018) to minimize a cross-entropy as the objective function. We should note that the conditional entropy is upper bounded by cross-entropy, which validates the possibility to minimize the cross-entropy objective. In particular, the empirical objective becomes

$$\underset{\Psi}{\text{Minimize}}\ \frac{1}{n}\sum_{i=1}^{n}\sum_{s\in\{-1,1\}} p_{\widehat{\theta}}(s_i = s|X_{sub}(i)) \log p_{\widehat{\theta}}(s_i = s|G_{sub}(\Psi), X_{sub}(i)),$$

where n is the sampling size and $p_{\widehat{\theta}}(s_i = s|G_{sub}(\Psi), X_{sub}(i))$ denotes the classification logits conditional on the subgraph $G_{sub}(\Psi)$ and the subset feature $X_{sub}(i)$ of the ith graph sample.

So far, we have implicitly assumed that the subset feature X_{sub} is known. This is not the case in practice. In the context where the subset feature X_{sub} is not given, the main challenges are: (a) identifying the subset is unknown; and (b) the fact that integrating this feature selection into the developed subgraph identification optimization framework is not trivial. Motivated by the great success of self-supervised techniques in large neural language models (Devlin et al. 2018), we propose to use a "masking" approach to convert the feature subsetting problem into an optimization problem that can be naturally combined with subgraph identification. Specifically, we define a binary vector $\mathcal{B} \in \{0,1\}^T$ which holds the same dimension as the raw node feature. For each node v_i and its raw node feature X_i, $i = 1,\ldots,N$, we multiply the raw feature with the binary vector $\mathcal{B}$ to obtain $X_i \odot \mathcal{B}$, where $\odot$ is the Hadamard product. Intuitively, the vector $\mathcal{B}$ converts the value in some dimension of the node feature to 0. This aligns with the rationale that if a particular feature is not important, the corresponding weights in the neural network weight matrix take values close to 0. In terms of the principle of information gain, this type of masking does not significantly decrease the probability of the prediction or alter the information gain.

The binary vector $\mathcal{B}$ is non-smooth; we also consider a continuous relaxation for the vector by leveraging a sigmoid mapping, so that the feature selection procedure becomes a smooth optimization problem, that is,

$$X \odot \text{sigmoid}(\widetilde{\mathcal{B}}),$$

where $\widetilde{\mathcal{B}} \in \mathbb{R}^T$ is a real-valued vector and the sigmoid$(\widetilde{\mathcal{B}})$ is applied to each row of X. Next, we remove the low values in $\widetilde{\mathcal{B}}$ through thresholding to arrive at the feature subsetting.

The subgraph identification and the feature selection can be naturally integrated into a single minimization problem:

$$\min_{\Psi,\widetilde{\mathcal{B}}} \frac{1}{n}\sum_{i=1}^{n}\sum_{s\in\{-1,1\}} p_{\widehat{\theta}}(s_i = s|X(i)\odot\text{sigmoid}(\widetilde{\mathcal{B}}))$$
$$\log p_{\widehat{\theta}}(s_i = s|G_{sub}(\Psi), X(i)\odot\text{sigmoid}(\widetilde{\mathcal{B}})),$$

which forms a unified optimization framework. This unified optimization framework allows for the simultaneous identification of influential subgraphs and important node features, leading to a more interpretable and efficient model. The resulting optimization problem can be solved using gradient-based techniques.

5. Theory

In this section, we present the main theoretical results of the estimation model. First, we demonstrate that in contrast to standard GCNs, our approach is capable of effectively representing the feature differences among various-order neighbors. Second, we study the capacity of the actor graph neural network in terms of empirical Rademacher complexity. The derived bound is tight through careful analysis of both the lower and upper bounds. Furthermore, we provide a probabilistic upper bound on the generalization error of the actor-critic graph neural network which is calibrated in the fusion algorithm.

Theorem 5.1. The MaGNet actor-critic graph neural network is capable of learning a $\Delta(K)$-representer, which means it can sufficiently and effectively capture K-order node neighbor information.

Theorem 5.1 demonstrates that our estimation model can learn various-order information. This ensures the capability of the proposed estimation model on high-order message passing, where nodes receive latent representations from their 1-order neighbors as well as further K-order neighbors at the

information aggregation step. In contrast, the existing GCNs are not capable of representing this class of operations, even when stacked over multiple layers. Please see further justification for this statement in Section B of the Appendix. To establish the bounds on generalization errors, we make the following technical assumptions.

Assumption 5.1. The feature vector of any graph is contained in a L_2-ball with radius $\widetilde{c}$. Specifically, the L_2 norm of the feature vector $\|X_i\|_2 \leq \widetilde{c}$ for all $i = 1, \ldots, N$ and some constant $\widetilde{c} > 0$.

Assumption 5.2. Any weight matrix in the estimation model satisfies that

$$\|w_{\text{MLP}}^{(l)}\|_F \leq c_2, \|w_{\text{MLP}}^{(l_0)}\|_2 \leq c_1, \|W\|_F \leq c_0,$$

with some constant $c_0, c_1, c_2 > 0$ and the Frobenius norm $\|\cdot\|_F$, where $w_{\text{MLP}}^{(l)}$ is the weight matrix in lth layer of MLP for $l = 1, \ldots, l_0 - 1$, and $w_{\text{MLP}}^{(l_0)}$ is the weight vector in the last layer of MLP.

Assumption 5.3. The maximum number of elements in the graph Laplacian matrix is bounded above by, that is, $\max_{i\in[N]} \max_{j\in[N]} |\mathcal{L}_{ij}| \leq c_{\mathcal{L}}$.

Assumption 5.4. We only consider the undirected, no loops , and no multi-edges graphs, and the number of node neighbors $|\mathcal{N}(v_i)|$ for all node $v_i \in V$ is equal to some constant $q \in \mathbb{N}^+$.

Assumption 5.5. The maximum hidden dimension across all neural network layers is h.

The above assumptions are common in the (graph) neural network literature. Assumptions 5.1–5.2 impose norm constraints on the parameters and input feature, making the model class fall into a compact metric space (Liao, Urtasun, and Zemel 2020). In general, Assumption 5.1 does not require a specific data distribution for the feature vector. It holds as long as the feature vector has a bounded L_2-norm, regardless of its distribution. For example, feature vectors following the truncated Gaussian distribution, uniform distribution, logarithmic distribution, and autoregressive distribution within the L_2-ball all satisfy Assumption 5.1. Assumption 5.3 is a standard assumption to control the intensity of the graph Laplacian in GNN literature (Hamilton 2020). Assumption 5.4 requires us to focus on homogeneous graphs (Liao, Urtasun, and Zemel 2020; Lv 2021). Assumption 5.5 is a standard assumption in bounding the width of neural network layers.

We first present our result on bounding the Rademacher complexity of the model class $\mathcal{F}_{c_0,c_1,c_2}$, which is the estimation model part before and up to the step that produces $H^{(K)}$. For i_0th graph sample, formally, we define our estimation model class $\mathcal{F}_{c_0,c_1,c_2}$ in the setting of $K = 3$ and $l_0 = 2$ without loss of generality:

$$\begin{aligned}\mathcal{F}_{c_0,c_1,c_2} := \Bigg\{ f(X(i_0)) = \sigma\Bigg(\sum_{q=1}^{d_1} w_{\text{MLP}q}^{(2)} \sigma \\ \Bigg(\sum_{t=1}^{k} w_{\text{MLP}tq}^{(1)} \frac{1}{N} \sum_{m=1}^{N} \sum_{i=1}^{N} \mathcal{L}_{mi} \sum_{v=1}^{N} \mathcal{L}_{iv} \\ \times \sum_{j\in\mathcal{N}(v)} \mathcal{L}_{vj} \langle X(i_0)_j, \mathbf{w}_t \rangle \Bigg) \Bigg), \quad i_0 \in [n], \\ \|w_{\text{MLP}}^{(1)}\|_F \leq c_2, \|w_{\text{MLP}}^{(2)}\|_2 \leq c_1, \|W\|_F \leq c_0 \Bigg\}, \quad (8)\end{aligned}$$

where $\sigma(\cdot)$ is some activation function, $w_{\text{MLP}}^{(1)}$ and $w_{\text{MLP}}^{(2)}$ is the weight matrix and vector for the first and second layer of the MLP for critic network, respectively. The $\mathbf{w}_t$ is the tth column of the weight matrix W. Note that we use this particular setting as an example of the model class for simplifying the expression. The following theoretical results hold for the general case of K and l_0.

Definition 5.1. Given the input node feature matrix $\{X(i)\}_{i=1}^n$ and the model class of the actor-critic graph neural network $\mathcal{F}_{c_0,c_1,c_2}$, the empirical Rademacher complexity of $\mathcal{F}_{c_0,c_1,c_2}$ is defined as

$$\begin{aligned}&\widehat{\mathcal{R}}(\mathcal{F}_{c_0,c_1,c_2}) \\ &:= \mathbb{E}_\epsilon \left[\frac{1}{n} \sup_{f\in\mathcal{F}_{c_0,c_1,c_2}} \left| \sum_{j=1}^{n} \epsilon_j f\left(X(j)\right) \right| X(1), X(2), \ldots, X(n) \right],\end{aligned}$$

where $\{\epsilon_i\}_{i=1}^n$ is an iid family of Rademacher variables, independent of $\{X(i)\}_{i=1}^n$.

Theorem 5.2. Under Assumptions 5.1–5.5, the empirical Rademacher complexity is bounded by

$$\widehat{\mathcal{R}}(\mathcal{F}_{c_0,c_1,c_2}) \leq \frac{3(L_0)^{l_0} c_0 c_1 c_2 c_{\mathcal{L}}^{K-1} \widetilde{c}(K+l_0) h^{1.5} q^{K+0.5}}{2\sqrt{n}} |\lambda_{\max}(\mathcal{L})|; \quad \text{Upper Bound}$$

$$\widehat{\mathcal{R}}(\mathcal{F}_{c_0,c_1,c_2}) \geq \frac{(L_0)^{l_0} c_0 c_1 c_2 (\min_{m,i\in[N]} \mathcal{L}_{mi})^{K-1} \widetilde{c}(K+l_0) h^{1.5} q^{K}}{5\sqrt{n}} |\lambda_{\min}(\mathcal{L})|; \quad \text{Lower Bound}$$

where L_0 is the Lipschitz constant for activation function $\sigma(\cdot)$ in the critic network, and $\lambda_{\min}(\mathcal{L})$ and $\lambda_{\max}(\mathcal{L})$ are the finite minimum and maximum absolute eigenvalue of graph Laplacian $\mathcal{L}$.

Theorem 5.2 demonstrates that our derived upper bound is tight up to some constants when comparing it to the lower bound. Theorem 5.2 indicates that the upper bound of $\widehat{\mathcal{R}}(\mathcal{F}_{c_0,c_1,c_2})$ depends on the number of graph instances, the node

degree of the graph, and the graph convolution filter, and also the maximum width of the neural networks. Interestingly, the above bound is independent of the maximum number of nodes, N, for traditional regular graphs. Note that while Esser, Chennuru Vankadara, and Ghoshdastidar (2021) also examine the relation between the graph information and the feature information, their bounds are not directly comparable to our theoretical results. Lv (2021) also establishes the Rademacher complexity bound; however, the focus is only on node classification tasks and graph neural networks with one hidden layer.

Applying our results in empirical Rademacher complexity $\widehat{\mathcal{R}}(\mathcal{F}_{c_0,c_1,c_2})$ to generalization analysis, we now state the fundamental result of the generalization bound of the estimation model. We denote $conv(\mathcal{F}_{c_0,c_1,c_2})$ as the closed convex hull of $\mathcal{F}_{c_0,c_1,c_2}$. That is, $conv(\mathcal{F}_{c_0,c_1,c_2})$ consists of all functions that are pointwise limits of convex combinations of functions from $\mathcal{F}_{c_0,c_1,c_2}$:

$$\operatorname{conv}(\mathcal{F}_{c_0,c_1,c_2}) := \Big\{ f : \forall x, f(x) = \lim_{K\to\infty} f_K(x), f_K = \sum_{k=1}^{K} w_k f_k, \sum_{k=1}^{K} w_k = 1, f_k \in \mathcal{F}_{c_0,c_1,c_2}, K \geq 1 \Big\}.$$

Obviously, we can observe that the combination in (3) belongs to $conv(\mathcal{F}_{c_0,c_1,c_2})$. Next, we present the probabilistic generalization error for the estimation model.

Theorem 5.3. Under Assumptions 5.1–5.5, given $\widehat{s}$ as the predicted label from an K-layers actor-critic graph neural network with true label s_0, then the probabilistic upper bound of the generalization error

$$P \quad (\widehat{s}s_0 \leq 0) \leq \mathcal{O}\Bigg(\prod_{k=1}^{K} \Bigg\{ \underbrace{\sqrt{\epsilon^{(k)}\left(1-\epsilon^{(k)}\right)} + \left(\frac{\log\log_2\left(2\left(\log\prod_{k=1}^{K}\sqrt{\frac{1-\epsilon^{(k)}}{\epsilon^{(k)}}} \vee 1\right)\right)}{n} \right)^{0.5}}_{\text{fusion estimation bias}} + \underbrace{\sqrt{\frac{1}{2n}\log\frac{2}{\delta}}}_{\text{intrinsic uncertainty}} + \underbrace{\frac{(L_0)^{l_0} c_0 c_1 c_2 c_{\mathcal{L}}^{K-1} \widetilde{c}(K+l_0) h^{1.5} q^{K+0.5}}{\sqrt{n}} |\lambda_{\max}(\mathcal{L})| \left(\log\prod_{k=1}^{K}\sqrt{\frac{1-\epsilon^{(k)}}{\epsilon^{(k)}}} \vee 1 \right)}_{\text{local complexity}} \Bigg\} \Bigg),$$

with probability at least $1-\delta$ for $\delta \in [0,1)$, where $\vee$ is a maximum operator.

Theorem 5.3 demonstrates that the generalization error of the proposed estimation model is bounded in terms of the error rate at the kth iteration defined in (4), that is, $\epsilon^{(k)} \geq 1/2$ for any $k = 1,\ldots,K$. In comparison to the generalization bound on vanilla GNNs (Scarselli, Tsoi, and Hagenbuchner 2018; Garg, Jegelka, and Jaakkola 2020), our bound is independent of the number of hidden units and the maximum number of nodes N in any input graph. For a regular graph with $q = \mathcal{O}(1)$ (Bollobás 1998), we conclude that $\lambda_{\max}(\mathcal{L}) = 1$, which yields a generalization error bound of order $\mathcal{O}(1/\sqrt{n})$ that is fully independent of the number of nodes N.

6. Simulation Studies

In this section, we present a comprehensive evaluation of MaGNet using synthetic datasets. To generate graphs, we allow the number of nodes N in the graph to vary with different graph sample sizes n. Each node has a p-dimensional feature in estimation or interpretation tasks. We distinguish two categories of nodes, specifically, important nodes and non-important nodes, and we generate their features by applying two separate processes, resulting in two different settings. To establish graph structure, we calculate the correlation across the varying node features to obtain the adjacency matrix. In all the experiments, we use a binary outcome of interest. In what follows, we illustrate the data-generating process for each setting.

Setting 1: For the important nodes, features are generated by following a multivariate Gaussian distribution, MVN$(0, 0.1 \cdot I)$, where I is an identity matrix. On the other hand, for the non-important nodes, the instance features are sampled from a uniform distribution, Unif$(0,1)$. The difference in feature generation mechanisms creates a distributional gap influencing the classification target outcome. It is important to note that the target outcome or classification rule is based solely on the features of important nodes and remains independent of those of non-important nodes. This generation process allows a good classifier to be able to separate important nodes from non-important ones. In this setting, we form a linear classification rule $\frac{\mathbf{e}^\top X_{V_0}\mathbf{e}}{|V_0|} + N(0,0.1) > 0$, where $\mathbf{e}$ is the column vector whose entries are all 1's. Here, X_{V_0} is the node feature matrix associated with the important node set V_0. It has dimension $|V_0| \times p$, where $|\cdot|$ is the cardinality operator.

Setting 2: For the important nodes, features are generated following a Gaussian process in order to introduce a dependency between the temporal features. The mean function $m(x_t)$ for $t = 1,\ldots,p$ and $x_t \sim$ Unif$(0,1)$. The kernel covariance function of the Gaussian process is $k(x_t, x_{t'}) = \sigma^2 \exp\left(-\frac{1}{l^2}|x_t - x_{t'}|^2\right)$, where $l = 1$ and $\sigma = 1$. In contrast, the instance features in the non-important nodes are sampled from a Gaussian process with the same mean function, but σ is set to 2.5 in the kernel covariance function. In this setting, we use a nonlinear and complex classification rule as follows:

$$\sin(\mathbf{x}\mathbf{e}_1)\cdot\cos(\mathbf{x}\mathbf{e}_2) + \mathbf{x}^{\circ 3}\mathbf{e}_3 + N(0,0.1) > 0,$$

Table 1. The results of classification accuracy over 50 repeated experiments in Setting 1.

Sample size	Important nodes	Nodes	MaGNet	PNGAT	GTN	GPS	MSGCN	APPNP
100	10	30	**0.761**	0.715	0.745	0.742	0.719	0.708
		50	**0.745**	0.701	0.738	0.734	0.710	0.696
		75	**0.740**	0.692	0.717	0.719	0.702	0.678
	20	30	**0.772**	0.725	0.754	0.760	0.736	0.719
		50	**0.764**	0.715	0.740	0.741	0.728	0.709
		75	**0.752**	0.708	0.728	0.722	0.719	0.702
250	10	30	**0.779**	0.744	0.768	0.760	0.739	0.732
		50	**0.774**	0.734	0.754	0.747	0.731	0.726
		75	**0.763**	0.720	0.741	0.732	0.723	0.715
	20	30	0.785	0.748	0.769	**0.787**	0.773	0.744
		50	**0.781**	0.736	0.757	0.761	0.755	0.738
		75	**0.769**	0.732	0.750	0.747	0.734	0.722

The bold values identify the best method in each setting.

Table 2. The results of classification accuracy over 50 repeated experiments in Setting 2.

Sample size	Important nodes	Nodes	MaGNet	PNGAT	GTN	GPS	MSGCN	APPNP
100	10	30	**0.753**	0.718	0.740	0.728	0.697	0.746
		50	**0.742**	0.710	0.729	0.718	0.684	0.728
		75	**0.736**	0.697	0.715	0.710	0.672	0.709
	20	30	**0.768**	0.740	0.749	0.762	0.708	0.728
		50	0.755	0.729	0.744	**0.760**	0.701	0.711
		75	**0.748**	0.710	0.717	0.719	0.693	0.704
250	10	30	0.776	0.735	**0.778**	0.766	0.728	0.735
		50	**0.771**	0.733	0.759	0.758	0.720	0.727
		75	**0.765**	0.731	0.740	0.747	0.711	0.724
	20	30	**0.786**	0.754	0.763	0.770	0.767	0.761
		50	**0.779**	0.750	0.756	0.759	0.752	0.753
		75	**0.774**	0.741	0.752	0.748	0.734	0.739

The bold values identify the best method in each setting.

where $\circ$ is Hadamard power, and the row vector $\mathbf{x} = \frac{\mathbf{e}^{\mathrm{T}} X_{|V_0|\times p}}{|V_0|}$. Moreover, $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$ are column vectors with dimension p. In particular, $\mathbf{e}_1 = [\underbrace{1, 1, \ldots, 1}_{\lfloor p/3 \rfloor}, 0, \ldots, 0]$, $\mathbf{e}_2 = [0, \ldots, 0, \underbrace{1, 1, \ldots, 1}_{\lfloor p/3 \rfloor}, 0, \ldots, 0]$, $\mathbf{e}_3 = [0, \ldots, 0, \underbrace{1, 1, \ldots, 1}_{\lfloor p/3 \rfloor}]$.

6.1. Evaluation of the Estimation Model

In this section, we evaluate the classification accuracy of the MaGNet estimation model by comparing it against several state-of-the-art GNN approaches, including GPS Graph Transformer Network (GPS, Rampášek et al. 2022), Graph Transformer Network (GTN, Yun et al. 2019), Mean-Subtraction-Norm Graph Convolutional Network (MSGCN, Yang et al. 2020), PairNorm Graph Attention Network (PNGAT, Zhao and Akoglu 2019), and Approximate Personalized Propagation of Neural Predictions (APPNP, Gasteiger, Bojchevski, and Günnemann 2018). The results are provided in Tables 1 and 2.

As shown in Tables 1 and 2, the MaGNet estimation model provides the best classification results among all the competing methods in general. This superior performance is consistent across varying sample sizes, node quantities, and important node sizes, indicating MaGNet's robust performance in graph classification tasks. This is mainly due to MaGNet's ability to effectively integrate both local and global information. The advantage of our model in solving the memoryless and over-smoothing issues results in effective and powerful representations for graph-structured data.

6.2. Evaluation of the Interpretation Model

In this section, we introduce different types of interpretation tasks and the corresponding results of the interpretation model. In particular, we consider three types of model interpretation tasks: node-wise, edge-wise, and feature-wise reasoning. We note that each of them is aligned with the functionalities of the proposed MaGNet interpretation model. In the following interpretation tasks, we consider a correlated and temporal data-generating process as in the simulation setting 2 in order to mimic the scenario with time-varying features in neural activity experiments.

In the node-wise interpretation tasks, the ultimate aim is to retain important nodes after node-wise reasoning. This is particularly crucial in practice for achieving a parsimonious model that simultaneously maintains interpretability and performance. In the edge-wise interpretation tasks, we initially define the notions of important and redundant edges (REs). An important edge refers to an edge connecting two important nodes. In contrast, all other edges not satisfying this condition are considered non-important edges. In the task, we seek to minimize the redundant edges in the trained MaGNet estimation model. To evaluate the interpretation model performance, we define two metrics: the absolute metric (AM) and the relative metric (RM), as follows:

$$\mathrm{AM} := \frac{\#\text{ of existing RE after reasoning}}{\#\text{ of all possible RE}},$$

Table 3. The node-wise interpretation performance over 50 repeated experiments.

Sample size	Important nodes	All nodes	MaGNet	IntGradients
100	10	30	**0.763**	0.708
		50	**0.745**	0.684
	20	30	**0.817**	0.754
		50	**0.795**	0.752
250	10	30	**0.775**	0.723
		50	**0.758**	0.687
	20	30	**0.826**	0.767
		50	**0.814**	0.759

The bold values identify the best method in each setting.

Table 4. The edge-interpretation performance over two metrics with 50 repeated experiments, where higher RM is better, and lower AM is better.

Metric	Important nodes	All nodes	MaGNet	IntGradients
RM	10	30	**0.806**	0.768
		50	**0.791**	0.754
	20	30	**0.827**	0.786
		50	**0.812**	0.771
AM	10	30	**0.090**	0.132
		50	**0.128**	0.161
	20	30	**0.056**	0.101
		50	**0.084**	0.131

The bold values identify the best method in each setting.

and

$$\text{RM} := \frac{\#\text{ of existing RE before reasoning} - \#\text{ of existing RE after reasoning}}{\#\text{ of existing RE before reasoning}}.$$

In the feature-wise interpretation tasks, we aim to detect influential features for graph nodes. In this task, we choose a submatrix with dimension $|V_0| \times p_0$ from the feature matrix X for some $p_0 < p$. Then, we use this submatrix to form a classification rule as discussed above.

In the following, we present the results of the three types of interpretation task experiments. We compare our method to a state-of-the-art approach, Integrated Gradients (IntGradients, Sundararajan, Taly, and Yan 2017), which is a gradient-based model interpretation approach designed for deep neural networks. For the implementation of IntGradients, we adapt the module of IntGradients to graph neural network settings and apply it to our trained MaGNet estimation model. Note that for the feature-wise interpretation tasks, we only evaluate the model performance of the MaGNet interpretation method, because IntGradients is not able to perform such type of task.

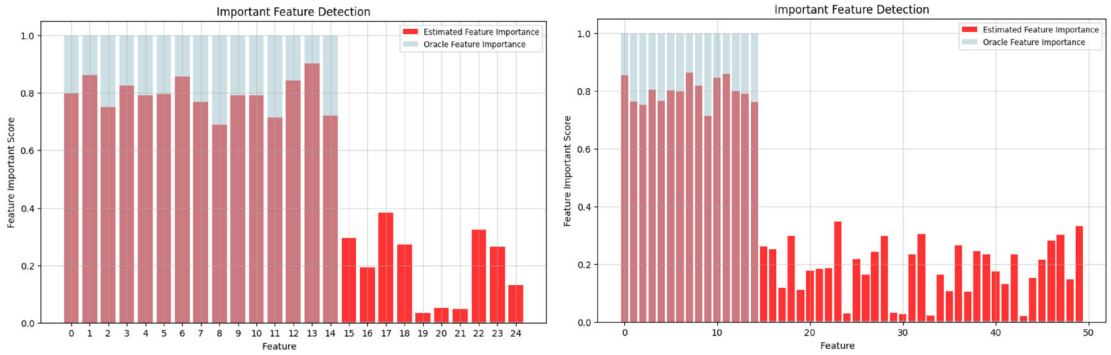
As shown in Table 3, the recovery rate of important nodes using the MaGNet interpretation model consistently outperforms the competing method across various settings. This is mainly due to the strength of our method in terms of leveraging the information gain to directly assess the reduction of uncertainty for the node subgraph. This technique incorporates statistical uncertainty as a measurement criterion instead of applying a fully deterministic gradient-based method. Another advantage of our method is the reparameterization strategy for continuously approximating discrete variables. This makes the proposed interpretation framework more computationally stable than the competing methods. Furthermore, the MaGNet interpretation model is particularly designed for the graph neural network and inherits the properties of the trained MaGNet estimation model. As a result, it is able to leverage local and global information to do the interpretation. The results of the edge-wise interpretation tasks are summarized in Table 4. It shows that our proposed method has a high edge reduction rate, showcasing its effectiveness in pruning the non-important edges. Importantly, the method exhibits the ability to retain significant edges, suggesting an inherent capability in discriminating between important and non-important edges and, thus, preserving the influential subgraph structure. This performance is consistently validated across different settings, demonstrating the proposed method's robustness and adaptability to varying graph sizes. Further, these results demonstrate that our interpretation model can lead to a more parsimonious and interpretable results in practice. The results of the feature-wise interpretation task are reported in Figure 2. The interpretation model tends to assign high scores to the top influential feature. It indicates that our model achieves great performance in temporal feature reasoning.

7. Application to Local Field Potential Activity Data from the Rat Brain

In this section, we apply the proposed method to neural activity data recorded from an array of electrodes implanted inside the brain. The brain region of interest is the hippocampus, a region near the middle of the rat brain known to be important for the temporal organization of our memories and behaviors. Although it is well established that the hippocampus plays a key role in this function across mammals, the underlying neuronal mechanisms remain unclear. To shed light on these underlying mechanisms, we previously recorded neural activity in the hippocampus of rats performing a complex sequence memory task (Allen et al. 2016) (as such high-precision data are currently not available in humans). Using that dataset, our objective here is to apply the proposed method to identify key functional relationships in the local field potential (LFP) activity simultaneously recorded across electrodes during task performance, as this information could provide novel insights into potential functional relationships within that region.

The LFP neural activity data were collected from the CA1 region of the hippocampus while rats performed an odor sequence memory task (Figure 3). In this task, rats received repeated presentations of odor sequences (e.g., ABCDE) at a single odor port and were required to identify each item as either "in sequence" (InSeq; e.g., AB<u>C</u>…) or "out of sequence"(OutSeq; e.g., AB<u>D</u>…). Importantly, the recordings were performed from surgically implanted electrodes (tetrodes), organized into two bundles, which spanned much of the proximo-distal axis of dorsal CA1. This experimental design thus provides a unique opportunity to directly examine the anatomical distribution of information processing along that axis.

In recent work, Shahbaba et al. (2022) showed that information about trial content, such as the identity of the odor presented and whether it was presented in or out of sequence, could be accurately decoded from the ensemble spiking activity. However, that study did not determine whether task-relevant information was also contained in the local field potential activ-



(a) The dimension of temporal feature $p = 25$. (b) The dimension of temporal feature $p = 50$.

Figure 2. The feature-interpretation performance over 50 repeated experiments.

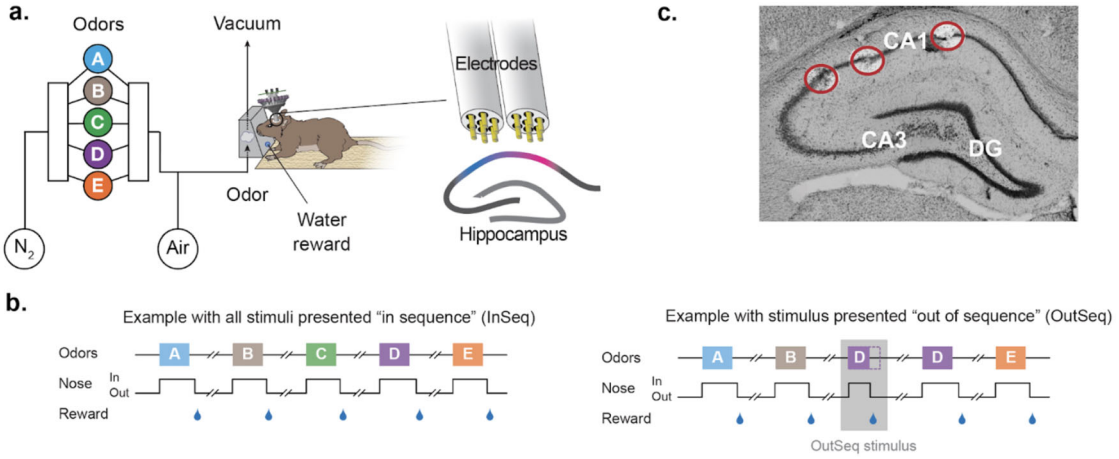


Figure 3. **(a.)** The task involves repeated presentations of sequences of odors and requires rats to determine whether each odor was presented "in sequence" (InSeq; e.g., ABC…) or "out of sequence" (OutSeq; e.g., ABD…). Using an automated delivery system (left), all odors were presented in the same odor port (median interval between odors ∼5 s). Recordings were performed from electrodes organized into two bundles (right), which spanned much of the proximo-distal axis of dorsal CA1. **(b.)** In each session, the same sequence was presented multiple times, with approximately half the presentations including all InSeq trials (left) and the other half including one OutSeq trial (right). Each odor presentation was initiated by a nosepoke and rats were required to correctly identify each odor as either InSeq (by holding their nosepoke response until a tone signaled the end of the odor at 1.2 sec) or OutSeq (by withdrawing their nose before the signal; <1.2 sec) to receive a water reward. Incorrect responses resulted in the termination of the sequence. **(c.)** Location of three electrode tips (red circles). The leftmost and rightmost electrodes approximate the extent of the CA1 transverse axis recorded in each animal.

ity. A fundamentally different data type from the discrete neural spiking activity, the LFP's continuous signal is more challenging to decode. To our knowledge, there are only two reports that exclusively use LFP to successfully decode spatial information in the hippocampus, which requires high-density recordings (Agarwal et al. 2014; Taxidis et al. 2015), and none showing decoding of nonspatial information from hippocampal LFP alone. To address this gap in knowledge, here we examined whether the content of odor trials can be decoded from hippocampal LFP activity and, if so, whether the dynamics vary over space (electrodes) and time.

For this analysis, we have focused on decoding the two main trial types (InSeq and OutSeq) using LFP activity from the 0–500 ms period (0 = odor onset), a time period in which there are no overt differences in the behavior of the animals between InSeq and OutSeq trials. We considered each rat's data an independent dataset and performed the classification evaluation task separately. For each rat's data, we randomly selected about 70 graph instances as the training set and the other 30 graph samples as the testing set. Figure 4 shows that the MaGNet estimation model achieves the best performance for all the rats.

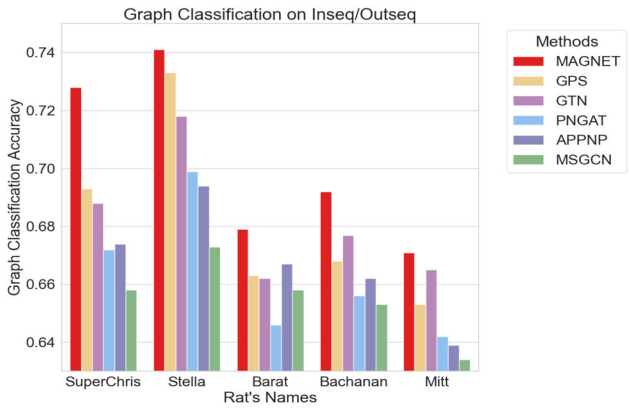


Figure 4. Barplot of estimation accuracy for the MaGNet estimation model and alternative competing approaches on decoding the two main trial types.

This is because our method's integration of both low-order and high-order information effectively uses more information into the latent representation. In addition, due to the proposed actor-critic structure, our method is less likely to suffer the *over-smoothing* and *memoryless* issue.

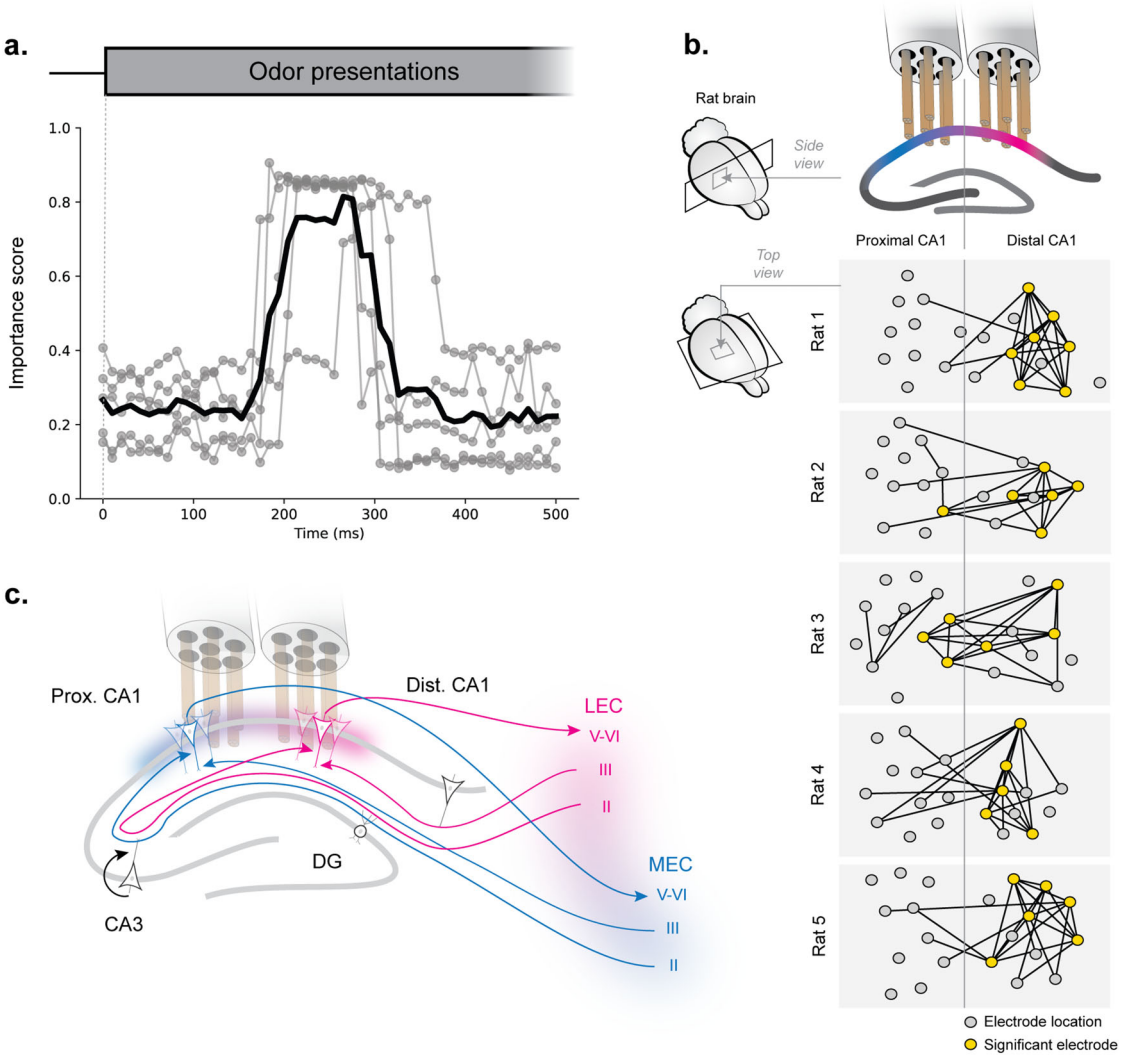


Figure 5. **(a.)** Significant decoding of InSeq and OutSeq trials based on LFP activity during the first 500ms of odor trials. Scores peak during the 185–320 ms period, prior to the behavioral response. Grey traces indicate individual subject decodings, the black line indicates the mean across subjects. **(b.)** Informative electrode nodes in the distal region of CA1. Schematic showing side view of electrode bundles implanted across the CA1 proximal-distal axis (Top). Schematic showing a top view of the anatomical distribution of electrodes across subjects based on electrode tract reconstruction (bottom). Yellow indicates significant nodes (electrodes). **(c.)** The clustering of informative nodes in distal CA1 is consistent with known anatomical differences in input connections. Odor information enters the hippocampus primarily through the LEC, which more strongly projects to the distal segment of CA1. In contrast, the MEC more strongly projects to proximal CA1. Approximate location of the implanted electrode bundles is shown.

We have also investigated the temporal dynamics of this decoding during trial periods by applying our MaGNet interpretation model. Specifically, we examined the most informative time bins (Allen et al. 2020) for the InSeq/OutSeq classification in the first 500 ms of trials in Figure 5(a). We found that the most informative time bins occurred between ∼180 ms and ∼320 ms after the rats poked into the port. This timeline is consistent with reports of hippocampal neurons responding to odor information in as little as 100 ms (Allen et al. 2020) and with the expected timeline of InSeq/OutSeq identification within trials. With the neuropsychology's experimental knowledge, this implies that the MaGNet interpretation model is able to successfully identify the influential neural dynamics.

In addition, we have found that most informative electrodes clustered in the distal region of CA1 in Figure 5(b). In fact, across all five rats, the majority (86.7%) of significant electrodes were in distal CA1, and more than half of all electrodes in distal CA1 reached significance. This distribution of significant nodes suggests distal CA1 plays a more important role in representing InSeq/OutSeq information than proximal CA1, a pattern consistent with known differences in their anatomical connections (Figure 5(c)). Odor information enters the hippocampus primarily through the LEC (lateral entorhinal cortex), which more strongly projects to the distal segment of CA1. In contrast, the MEC (medial entorhinal cortex) more strongly projects to proximal CA1. However, the observation that some significant nodes also extended into proximal CA1 suggests that functional interactions among the two segments of CA1 are critical for task performance. In summary, we apply the MaGNet framework to LFP activity data recorded from the hippocampus of rats as they performed a challenging nonspatial sequence memory task. According to the analysis, we can decode the trial type (whether the odor was presented in or out of sequence) as well as identify the most informative trial periods and electrodes. Therefore, not only did the model provide the first direct evidence of decoding nonspatial trial content from hippocampal LFP activity alone, it

also provided a high degree of specificity about how this information was distributed over space and time. This neuroscience result is consistent with a growing literature on the influence of anatomical gradients on information processing within brain regions (Knierim, Neunuebel, and Deshmukh 2014; Witter et al. 2017), specifically with evidence that inputs carrying nonspatial information more strongly project to distal CA1 than proximal CA1 (Agster and Burwell 2009).

8. Discussion

In this article, we have proposed a novel graph neural network framework, MaGNet, which is able to effectively integrate both low-order and high-order information to allow powerful latent representation. Furthermore, MaGNet includes a practically useful interpretation component, which offers a tractable framework for identifying influential subgraphs, as well as important nodes, edges, and node features. In addition, we have also established rigorous theoretical foundations to assess the efficacy, statistical complexity, and generalizability of the MaGNet estimation model. These theoretical results ensure that the proposed estimation model is reliable and effective, contributing to the practical utility of MaGNet in various applications, especially in neuroscience.

One of the potential directions for exploration is to extend the current framework to accommodate different types of tasks. For example, rather than solely focusing on graph classification, the framework can be extended for node classifications, link prediction, and beyond. Furthermore, conducting rigorous theoretical investigations into the interpretation model—such as studying how the introduced approximations and relaxations affect selection errors—is both crucial and promising. Additionally, future research could explore studying the expressivity of the proposed method using the Weisfeiler-Lehman graph isomorphism test. Another potential research direction is to investigate generalization error in non-regular settings to provide a more comprehensive theoretical understanding of the current model. Exploring these directions will help expand the utility and effectiveness of our framework for a wide range of applications.

Moreover, there is a pressing need to expand the current framework to support dynamic settings. While modeling time-varying changes and dynamic systems holds central importance in numerous real-world applications, the current MaGNet framework (along with the majority of GNN models) is primarily tailored for static graph data. These models are capable of incorporating structural information into the learning process, but they fall short in capturing the evolution of dynamic graphs. Typically, dynamics in a graph refer to node attribute modifications or edge-structure changes, including the additions and deletions of nodes or edges. As a possible expansion of the existing MaGNet framework, we will explore the incorporation of node and edge activation functions to signify and capture the presence of the nodes and edges within each timestamp. This will enable subsequent utilization of attention mechanisms such as self-attention and neighborhood attention, which have shown efficacy in foundational models (Bommasani et al. 2021), in order to account for historical time-evolved information from preceding timestamps.

Supplementary Materials

The supplementary materials provide the proof of main theorems and additional technical results. The corresponding code for the algorithm, along with instructions to access the real dataset, is available online at the following GitHub repository: *https://anonymous.4open.science/r/NeuralDecoding-AC33*.

Acknowledgments

The authors thank the Editor, Associate Editor, and anonymous reviewers for their insightful suggestions and helpful feedback which improved the article significantly.

Disclosure Statement

The authors declare no financial interest that has arisen from the direct applications of this research.

Funding

This work was supported by NIH (awards NIH funding), NSF (awards DMS-2210640, DMS-1763272, CAREER IOS-1150292, BCS-1439267, and NCS-FR-2319618), the Simons Foundation (award 594598), and the Whitehall Foundation (award 2010-05-84).

ORCID

Keiland W. Cooper http://orcid.org/0000-0002-0358-9645

References

Abu-El-Haija, S., Perozzi, B., Kapoor, A., Alipourfard, N., Lerman, K., Harutyunyan, H., Ver Steeg, G., and Galstyan, A. (2019), "Mixhop: Higher-Order Graph Convolutional Architectures via Sparsified Neighborhood Mixing," in *International Conference on Machine Learning*, PMLR, pp. 21–29. [2,3]

Agarwal, G., Stevenson, I. H., Berényi, A., Mizuseki, K., Buzsáki, G., and Sommer, F. T. (2014), "Spatially Distributed Local Fields in the Hippocampus Encode Rat Position," *Science*, 344, 626–630. [11]

Agster, K. L., and Burwell, R. D. (2009), "Cortical Efferents of the Perirhinal, Postrhinal, and Entorhinal Cortices of the Rat," *Hippocampus*, 19, 1159–86. [13]

Allen, L. M., Lesyshyn, R. A., O'Dell, S. J., Allen, T. A., and Fortin, N. J. (2020), "The Hippocampus, Prefrontal Cortex, and Perirhinal Cortex are Critical to Incidental Order Memory," *Behavioural Brain Research*, 379, 112215. [12]

Allen, T. A., Salz, D. M., McKenzie, S., and Fortin, N. J. (2016), "Nonspatial Sequence Coding in CA1 Neurons," *Journal of Neuroscience*, 36, 1547–1563. [10]

Bodnar, C., Di Giovanni, F., Chamberlain, B., Lio, P., and Bronstein, M. (2022), "Neural Sheaf Diffusion: A Topological Perspective on Heterophily and Oversmoothing in GNNs," in *Advances in Neural Information Processing Systems* (Vol. 35), pp. 18527–18541. [1,2]

Bollobás, B. (1998), *Modern Graph Theory* (Vol. 184), New York: Springer. [8]

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., et al. (2021), "On the Opportunities and Risks of Foundation Models," arXiv preprint arXiv:2108.07258. [13]

Bottou, L. (2010), "Large-Scale Machine Learning with Stochastic Gradient Descent," in *Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22–27, 2010 Keynote, Invited and Contributed Papers*, pp. 177–186, Springer. [5]

Dabkowski, P., and Gal, Y. (2017), "Real Time Image Saliency for Black Box Classifiers," in *Advances in Neural Information Processing Systems* (Vol. 30). [2]

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018), "Bert: Pre-Training of Deep Bidirectional Transformers for Language Understanding," arXiv preprint arXiv:1810.04805. [6]

Esser, P., Chennuru Vankadara, L., and Ghoshdastidar, D. (2021), "Learning Theory Can (Sometimes) Explain Generalisation in Graph Neural Networks," in *Advances in Neural Information Processing Systems* (Vol. 34), pp. 27043–27056. [8]

Garg, V., Jegelka, S., and Jaakkola, T. (2020), "Generalization and Representational Limits of Graph Neural Networks," in *International Conference on Machine Learning*, pp. 3419–3430, PMLR. [8]

Gasteiger, J., Bojchevski, A., and Günnemann, S. (2018), "Predict then Propagate: Graph Neural Networks meet Personalized PageRank," in *International Conference on Learning Representations*. [9]

Graham, D., Wang, J., and Ravanbakhsh, S. (2019), "Equivariant Entity-Relationship Networks," arXiv preprint arXiv:1903.09033. [2]

Hamilton, W. L. (2020), *Graph Representation Learning*, San Rafael, CA: Morgan & Claypool Publishers. [1,4,7]

Ivanov, S., and Prokhorenkova, L. (2021), "Boost then Convolve: Gradient Boosting Meets Graph Neural Networks," in *International Conference on Learning Representations*. [5]

Kipf, T., Fetaya, E., Wang, K.-C., Welling, M., and Zemel, R. (2018), "Neural Relational Inference for Interacting Systems," in *International Conference on Machine Learning*, p. 2688, PMLR. [6]

Kipf, T. N., and Welling, M. (2016), "Semi-Supervised Classification with Graph Convolutional Networks," arXiv preprint arXiv:1609.02907. [1,2,4]

Knierim, J. J., Neunuebel, J. P., and Deshmukh, S. S. (2014), "Functional Correlates of the Lateral and Medial Entorhinal Cortex: Objects, Path Integration and Local–Global Reference Frames," *Philosophical Transactions of the Royal Society B: Biological Sciences*, 369, 20130369. [13]

Larose, D. T., and Larose, C. D. (2014), *Discovering Knowledge in Data: An Introduction to Data Mining* (Vol. 4), Hoboken, NJ: Wiley. [5]

Li, G., Müller, M., Ghanem, B., and Koltun, V. (2021), "Training Graph Neural Networks with 1000 Layers," in *International Conference on Machine Learning*, PMLR, pp. 6437–6449. [3]

Li, Q., Han, Z., and Wu, X.-M. (2018), "Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning," in *The AAAI Conference on Artificial Intelligence* (Vol. 32). [1]

Liao, R., Urtasun, R., and Zemel, R. (2020), "A Pac-Bayesian Approach to Generalization Bounds for Graph Neural Networks," arXiv preprint arXiv:2012.07690. [7]

Liu, J., Hooi, B., Kawaguchi, K., and Xiao, X. (2022), "MGNNI: Multiscale Graph Neural Networks with Implicit Layers," in *Advances in Neural Information Processing Systems*. [1]

Luo, D., Cheng, W., Xu, D., Yu, W., Zong, B., Chen, H., and Zhang, X. (2020), "Parameterized Explainer for Graph Neural Network," in *Advances in Neural Iinformation Processing Systems* (Vol. 33), pp. 19620–19631. [5]

Lv, S. (2021), "Generalization Bounds for Graph Convolutional Neural Networks via Rademacher Complexity," arXiv preprint arXiv:2102.10234. [7,8]

Maddison, C. J., Mnih, A., and Teh, Y. W. (2016), "The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables," arXiv preprint arXiv:1611.00712. [5]

Oono, K., and Suzuki, T. (2020), "Optimization and Generalization Analysis of Transduction through Gradient Boosting and Application to Multi-Scale Graph Neural Networks," in *Advances in Neural Information Processing Systems* (Vol. 33), pp. 18917–18930. [1]

Paulus, M., Choi, D., Tarlow, D., Krause, A., and Maddison, C. J. (2020), "Gradient Estimation with Stochastic Softmax Tricks," in *Advances in Neural Information Processing Systems* (Vol. 33), pp. 5691–5704. [6]

Rampášek, L., Galkin, M., Dwivedi, V. P., Luu, A. T., Wolf, G., and Beaini, D. (2022), "Recipe for a General, Powerful, Scalable Graph Transformer," *Advances in Neural Information Processing Systems*, 35, 14501–14515. [9]

Reitzner, M., Schulte, M., and Thäle, C. (2017), "Limit Theory for the Gilbert Graph," *Advances in Applied Mathematics*, 88, 26–61. [5]

Scarselli, F., Tsoi, A. C., and Hagenbuchner, M. (2018), "The Vapnik–Chervonenkis Dimension of Graph and Recursive Neural Networks," *Neural Networks*, 108, 248–259. [8]

Shahbaba, B., Li, L., Agostinelli, F., Saraf, M., Cooper, K. W., Haghverdian, D., Elias, G. A., Baldi, P., and Fortin, N. J. (2022), "Hippocampal Ensembles Represent Sequential Relationships Among An Extended Sequence of Nonspatial Events," *Nature Communications*, 13, 787. [10]

Sun, K., Zhu, Z., and Lin, Z. (2019), "Adagcn: Adaboosting Graph Convolutional Networks into Deep Models," arXiv preprint arXiv:1908.05081. [1,4,5]

Sundararajan, M., Taly, A., and Yan, Q. (2017), "Axiomatic Attribution for Deep Networks," in *International Conference on Machine Learning*, PMLR, pp. 3319–3328. [10]

Taxidis, J., Anastassiou, C. A., Diba, K., and Koch, C. (2015), "Local Field Potentials Encode Place Cell Ensemble Activation During Hippocampal Sharp Wave Ripples," *Neuron*, 87, 590–604. [11]

Veličković, P. (2023), "Everything is Connected: Graph Neural Networks," *Current Opinion in Structural Biology*, 79, 102538. [1]

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. (2017), "Graph Attention Networks," arXiv preprint arXiv:1710.10903. [1]

Witter, M. P., Doan, T. P., Jacobsen, B., Nilssen, E. S., and Ohara, S. (2017), "Architecture of the Entorhinal Cortex A Review of Entorhinal Anatomy in Rodents with Some Comparative Notes," *Frontiers in Systems Neuroscience*, 11, 46. [13]

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. (2019), "Simplifying Graph Convolutional Networks," in *International Conference on Machine Learning*, pp. 6861–6871, PMLR. [4]

Wu, J., Wang, X., Feng, F., He, X., Chen, L., Lian, J., and Xie, X. (2021), "Self-Supervised Graph Learning for Recommendation," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 726–735. [4]

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Philip, S. Y. (2020), "A Comprehensive Survey on Graph Neural Networks," *IEEE transactions on Neural Networks and Learning Systems*, 32, 4–24. [5]

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2018a), "How Powerful Are Graph Neural Networks?" arXiv preprint arXiv:1810.00826. [5]

Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K.-i., and Jegelka, S. (2018b), "Representation Learning on Graphs with Jumping Knowledge Networks," in *International Conference on Machine Learning*, pp. 5453–5462, PMLR. [1]

Yang, C., Wang, R., Yao, S., Liu, S., and Abdelzaher, T. (2020), "Revisiting Over-Smoothing in Deep GCNs," arXiv preprint arXiv:2003.13663. [1,9]

Ying, Z., Bourgeois, D., You, J., Zitnik, M., and Leskovec, J. (2019), "Gnnexplainer: Generating Explanations for Graph Neural Networks," in *Advances in Neural Information Processing Systems* (Vol. 32). [5]

Yun, S., Jeong, M., Kim, R., Kang, J., and Kim, H. J. (2019), "Graph Transformer Networks," in *Advances in Neural Information Processing Systems* (Vol. 32). [1,9]

Zhang, W., Sheng, Z., Yin, Z., Jiang, Y., Xia, Y., Gao, J., Yang, Z., and Cui, B. (2022), "Model Degradation Hinders Deep Graph Neural Networks," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2493–2503. [1]

Zhang, Z., Cui, P., and Zhu, W. (2020), "Deep Learning on Graphs: A Survey," *IEEE Transactions on Knowledge and Data Engineering*, 34, 249–270. [3]

Zhao, L., and Akoglu, L. (2019), "Pairnorm: Tackling Oversmoothing in GNNs," arXiv preprint arXiv:1909.12223. [1,9]