Towards Scalable (All-Pair) Message Passing for Node Classification beyond Explicit Topology

Anonymous Author(s)
Affiliation
Address
email

Abstract

Graph neural networks have been extensively studied for learning with inter-connected data. Despite this, recent evidence has revealed GNNs' deficiencies related to over-squashing, heterophily, handling long-range dependencies, edge incompleteness and particularly, the absence of graphs altogether. While a plausible solution is to learn new topology for message passing, issues concerning quadratic complexity hinder simultaneous guarantees for scalability and precision in large networks. In this paper, we introduce a novel all-pair message passing scheme for efficiently propagating layer-wise signals between arbitrary nodes. Specifically, the efficient computation per layer is enabled by a kernerlized Gumbel-Softmax operator that reduces the algorithmic complexity to linearity w.r.t. node numbers for learning latent structures from large, potentially fully-connected graphs in a differentiable manner. We also provide accompanying theory as justification for our design. Extensive experiments demonstrate the promising efficacy of the method in various tasks including node classification on different sizes of graphs (1K$\sim$1M) and graph-enhanced applications where input topology is missing. The full version of codes will be made public upon publication.

1 Introduction

Relational structure inter-connecting instance nodes as a graph is ubiquitous from social domains (e.g., citation networks) to natural science (protein-protein interaction), where graph neural networks (GNNs) [32, 19, 14, 36] have shown promising power for leveraging such data dependence as geometric priors. However, there arises increasing evidence challenging the core GNN hypothesis that propagating information along observed graph structures will necessarily produce better node-level representations for prediction on each individual instance node. Conflicts with this premise lead to commonly identified deficiencies with GNN message-passing rules w.r.t. heterophily [45], over-squashing [2], long-range dependencies [8], and graph incompleteness [11], etc.

Moreover, in graph-enhanced applications, e.g., text classification [42], vision navigation [12], physics simulation [30], etc., graph structures are often unavailable though individual instances are strongly inter-correlated. A common practice is to artificially construct a graph via some predefined rules (e.g., k-NN), which is agnostic to downstream tasks, presumably causing misspecification of GNNs' inductive bias on input geometry.

Natural solutions resort to organically combining learning optimal graph topology with message passing. However, one critical difficulty is the *scalability* issue with $O(N^2)$ (where N denotes #nodes) computational complexity, which is prohibitive for large networks (with $10K \sim 1M$ nodes). Some existing approaches harness neighbor sampling [43], anchor-based adjacency surrogates [4] and hashing schemes [40] to reduce the overhead; however, these strategies may sacrifice model precision and still struggle to handle graphs with million-level nodes. Another obstacle lies in the

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

Table 1: Comparison of popular graph structure learning approaches for node-level tasks where in particular, the graph connects all instance nodes and one's target is for prediction on each individual node. For *parameterization*, 'Function' means learning through functional mapping and 'Adjacency' means directly optimizing graph adjacency. For *expressivty*, 'Fixed' means learning one graph shared by all propagation layers and 'Layer-wise' means learning graph structures per layers. The *largest demo* means the largest # nodes of datasets used. † m denotes # anchors (i.e., a subset of nodes).

Models	Parameterization	Expressivity	Input Graphs	Inductive	Complexity	Largest Demo
LDS-GNN [11]	Adjacency	Fixed	Required	No	$O(N^2)$	0.01M
ProGNN [18]	Adjacency	Fixed	Required	No	$O(N^2)$	0.02M
VGCN [10]	Adjacency	Fixed	Required	No	$O(N^2)$	0.02M
BGCN [43]	Adjacency	Fixed	Required	No	$O(N^2)$	0.02M
GLCN [17]	Function	Fixed	Not necessary	Yes	$O(N^2)$	0.02M
IDGL [4]	Function	Fixed	Required	Yes	$O(N^2)$ or $O(Nm)^{\dagger}$	0.1M
ALIGNS (Ours)	Function	Layer-wise	Not necessary	Yes	$O(N)$ or $O(E)$	2M

increased degrees of freedom due to at least an $N \times N$ all-pair similarity matrix, which may result in large combinatorial search space and vulnerability to over-fitting.

In this work, we introduce a novel all-pair message passing scheme that can scale to large systems without compromising performance. We develop a kernelized Gumbel-Softmax operator that seamlessly synthesizes *random feature map* [26] and approximated sampling strategy [16], for distilling latent structures among all the instance nodes and yielding moderate gradients through differentiable optimization. Though such a combination of two operations involving randomness could potentially result in mutual distortion, we theoretically prove that the new operator can still guarantee a well-posed approximation for concrete variables (discrete structures) with the error bounded by feature dimensions. Furthermore, such a design can reduce the algorithmic complexity of learning new topology per layer to $O(N)$ by avoiding explicit computation for the cumbersome all-pair similarity.

The proposed module opens the door for a new class of GNNs capable of efficiently propagating messages between arbitrary node pairs in flexible layer-specific latent graphs. And to accommodate input graphs (if any), we devise two simple techniques: a relational bias and a edge-level regularization loss, as guidance for learning new structures. We evaluate our approach on diverse node classification tasks ranging from citation networks to images/texts. The results show its promising power for tackling heterophily, long-range dependencies, large-scale graphs, graph incompleteness and learning in the absence of input graphs. **The contributions of this paper are summarized as follows:**

1) We develop a kernelized Gumbel-Softmax operator which is proven to serve as a well-posed approximation for concrete variables, particularly the discrete latent structure among data points. The new module can reduce the algorithmic complexity for learning new message-passing topology from quadratic to linear w.r.t. instance node numbers (e.g., 1K$\sim$1M) of a dataset.

2) We further propose a novel class of GNNs with layer-wise message passing as operated over latent graphs potentially connecting all nodes. The latter are optimized with the network parameters in an end-to-end differentiable fashion through a new objective that essentially pursues sampling optimal topology from a posterior conditioned on input features and downstream labels.

3) We demonstrate the model's efficacy by extensive experiments over 10 datasets where significant improvement over strong GNN models and SOTA structure learning methods is shown. Besides, it successfully scales to large graph datasets with up to 2M nodes where prior arts failed, and reduces the time/space consumption of the competitors by up to 93.1%/80.6% on moderate sized datasets.

2 Related Works and Comparison

Graph Neural Networks. Building expressive GNNs is a fundamental problem in learning over graph data. With Graph Attention Networks (GAT) [36] as an early attempt, there are many follow-up works, e.g., [21, 39], considering weighting the edges in input graph for enhancing the expressiveness. Other studies, e.g., [27, 44] focus on sparsifying input structures to promote robust representations. However, these works cannot learn new edges out of the scope of input geometry, which may limit the model's receptive fields within local neighbors and neglect global information.

Graph Structure Learning. Going beyond observed topology, graph structure learning targets learning a new graph for message passing among all the instances [47]. One line of work is similarity-

driven where the confidence of edges are reflected by some similarity functions between node pairs, e.g., Gaussian kernels [40], cosine similarity [4], attention networks [17], non-linear MLP [7] etc. Another line of work optimizes the adjacency matrix. Due to the increased optimization difficulties, some sophisticated training methods are introduced for learning, such as bi-level optimization [11], variational approaches [10, 34], Bayesian inference [43] and projected gradient descent [18]. To push further the limits of structure learning, this paper proposes a new model ALIGNS (for A LImber Graph Network at Scale) whose merits are highlighted through a high-level comparison in Table 1.

Node-Level v.s. Graph-Level Prediction. We emphasize upfront that our focus is on *node-level* prediction tasks involving a single large graph such that scalability is paramount, especially if we are to consider arbitrary relationships across *all* nodes (each node is an instance with label and one can treat all the nodes non-i.i.d. generated due to the inter-dependence) for structure-learning purposes. Critically though, this scenario is quite distinct from *graph-level* classification tasks whereby each i.i.d. instance is itself a small graph and fully connecting nodes *within* each graph is computationally inexpensive. While this latter scenario has been explored in the context of graph structure learning [38, 9], existing efforts do not scale to the large graphs endemic to node-level prediction.

3 ALIGNS: A Limber Graph Network at Scale

Let $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ denote a graph with $\mathcal{N}$ a node set ($|\mathcal{N}| = N$) and $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$ an edge set ($|\mathcal{E}| = E$). Each node $u \in \mathcal{N}$ is assigned with node features $\mathbf{x}_u \in \mathbb{R}^D$ and a label y_u. We define an adjacency matrix $\mathbf{A} = \{a_{uv}\} \in \{0,1\}^{N \times N}$ where $a_{uv} = 1$ if edge $(u, v) \in \mathcal{E}$ and $a_{uv} = 0$ otherwise. Without loss of generality, $\mathcal{E}$ could be an empty set in case of no input structure. There are two common settings: transductive learning, where testing nodes are within the graph used for training, and inductive learning which handles new unseen nodes out of the training graph. The target is to learn a function for node-level prediction, i.e., estimate labels for unlabeled or new nodes in the graph.

General Model and Key Challenges. We start with the observation that the input structures may not be the ideal one for propagating signals among nodes and instead there exist certain latent structures that could facilitate learning better node representations. We thus consider the updating rule

$$\tilde{\mathbf{A}}^{(l)} = g(\mathbf{A}, \mathbf{Z}^{(l)}; \omega), \qquad \mathbf{Z}^{(l+1)} = h(\tilde{\mathbf{A}}^{(l)}, \mathbf{A}, \mathbf{Z}^{(l)}; \theta), \tag{1}$$

where $\mathbf{Z}^{(l)} = \{\mathbf{z}_u^{(l)}\}_{u \in \mathcal{N}}$ and $\tilde{\mathbf{A}}^{(l)} = \{\tilde{a}_{uv}^{(l)}\}_{u,v \in \mathcal{N}}$ denotes the node representations and the estimated latent graph of the l-th layer, respectively, and g, h are both differentiable functions aiming at 1) structure estimation for a layer-specific latent graph $\tilde{\mathbf{A}}^{(l)}$ based on node representations and 2) feature propagation for updating node representations, respectively. The model defined by Eqn. 1 potentially enables message passing between any node pair in each layer, which, however, poses two *challenges*: 1) how to reduce the prohibitive algorithmic complexity for learning new graphs in each layer; 2) how to enable end-to-end differentiable optimization for learning discrete structures. Notice that the first technical challenge is particularly non-trivial in our case, i.e., node-level prediction tasks, since the latent graphs could potentially connect *all the instance nodes* (e.g., from thousands to millions, depending on dataset sizes), which is fairly hard to simultaneously guarantee precision and scalability.

3.1 Efficient Learning Discrete Structures

We describe our new message-passing scheme with an efficient kernelized Gumbel-Softmax operator to resolve the aforementioned challenges. We assume $\mathbf{z}_u^{(0)} = \mathbf{x}_u$ as the initial node representation.

Kernelized Message Passing. We define a full-graph attentive network that estimates latent interactions among instance nodes and enables corresponding densely-connected message passing:

$$\tilde{a}_{uv}^{(l)} = \frac{\exp((W_Q^{(l)}\mathbf{z}_u^{(l)})^\top (W_K^{(l)}\mathbf{z}_v^{(l)}))}{\sum_{w=1}^{N} \exp((W_Q^{(l)}\mathbf{z}_u^{(l)})^\top (W_K^{(l)}\mathbf{z}_w^{(l)}))}, \quad \mathbf{z}_u^{(l+1)} = \sum_{v=1}^{N} \tilde{a}_{uv}^{(l)} \cdot (W_V^{(l)}\mathbf{z}_v^{(l)}), \tag{2}$$

where $W_Q^{(l)}$, $W_K^{(l)}$ and $W_V^{(l)}$ are learnable parameters in l-th layer. We omit non-linearity activation (after aggregation) for brevity. Eqn. 2 requires prohibitive $\mathcal{O}(N^2)$ complexity. Also, given large N, the normalization in the denominator would shrink attention weights to zero and lead to gradient vanishing. We call this problem as *over-normalizing*.

To accelerate the full-graph model, we observe that the *dot-then-exponentiate* operation in Eqn. 2 can be converted into a pairwise similarity function:

$$\mathbf{z}_u^{(l+1)} = \sum_{v=1}^{N} \frac{\kappa(W_Q^{(l)}\mathbf{z}_u^{(l)}, W_K^{(l)}\mathbf{z}_v^{(l)})}{\sum_{w=1}^{N}\kappa(W_Q^{(l)}\mathbf{z}_u^{(l)}, W_K^{(l)}\mathbf{z}_w^{(l)})} \cdot (W_V^{(l)}\mathbf{z}_v^{(l)}), \tag{3}$$

where $\kappa(\cdot,\cdot) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a positive-definite kernel measuring the pairwise similarity. The kernel function can be further approximated by random features (RF) [26]which serves as an unbiased estimation via $\kappa(\mathbf{a},\mathbf{b}) = \langle \Phi(\mathbf{a}), \Phi(\mathbf{b})\rangle_{\mathcal{V}} \approx \phi(\mathbf{a})^\top \phi(\mathbf{b})$, where the first equation is by Mercer's theorem with $\Phi : \mathbb{R}^d \to \mathcal{V}$ a basis function and $\mathcal{V}$ a high-dimensional vector space, and $\phi(\cdot) : \mathbb{R}^d \to \mathbb{R}^m$ is a low-dimensional feature map with random transformation. There are many potential choices for ϕ, e.g., Positive Random Features (PRF) [6]

$$\phi(\mathbf{x}) = \frac{\exp\left(\frac{-\|\mathbf{x}\|_2^2}{2}\right)}{\sqrt{m}}[\exp(\mathbf{w}_1^\top \mathbf{x}), \cdots, \exp(\mathbf{w}_m^\top \mathbf{x})], \tag{4}$$

where $\mathbf{w}_k \sim \mathcal{N}(0, I_d)$ is i.i.d. sampled random transformation. The RF converts dot-then-exponentiate operation into inner-product in vector space, which enables us to re-write Eqn. 3 (assuming $\mathbf{q}_u = W_Q^{(l)}\mathbf{z}_u^{(l)}$, $\mathbf{k}_u = W_K^{(l)}\mathbf{z}_u^{(l)}$ and $\mathbf{v}_u = W_V^{(l)}\mathbf{z}_u^{(l)}$ for simplicity):

$$\mathbf{z}_u^{(l+1)} = \sum_{v=1}^{N} \frac{\phi(\mathbf{q}_u)^\top \phi(\mathbf{k}_v)}{\sum_{w=1}^{N}\phi(\mathbf{q}_u)^\top \phi(\mathbf{k}_w)} \cdot \mathbf{v}_v = \frac{\phi(\mathbf{q}_u)^\top \sum_{v=1}^{N}\phi(\mathbf{k}_v)\cdot \mathbf{v}_v}{\phi(\mathbf{q}_u)^\top \sum_{w=1}^{N}\phi(\mathbf{k}_w)}. \tag{5}$$

The key advantage of Eqn. 5 is that the two summations are shared by each u, so that one only needs to compute them once and re-used for others. Such a property enables $\mathcal{O}(N)$ computational complexity for full-graph message passing, which paves the way for learning graph structures among large-scale instances. Moreover, one can notice that Eqn. 5 avoids computing the $N \times N$ similarity matrix, i.e., $\{\tilde{a}_{uv}^{(l)}\}_{N\times N}$, required by Eqn. 2, thus also reducing the learning difficulties.

Nevertheless, Eqn. 5 still suffers what we mentioned the over-normalizing issue. The crux is that the message passing is operated on a weighted fully-connected graph where, in fact, only partial edges are important. Also, such a deterministic way of feature aggregation over all the instances may increase the risk for over-fitting, especially when N is large. We next resolve the issues by distilling a sparse structure from the fully-connected graph.

Differentiable Stochastic Structure Learning. The difficultly lies in how to enable differentiable optimization for discrete graph structures. The weight $\tilde{a}_{uv}^{(l)}$ given by Eqn. 2 could be used to define a categorical distribution for generating latent edge $e_{uv} \sim \text{Cat}(\boldsymbol{\pi}_u^{(l)})$ where $\boldsymbol{\pi}_u^{(l)} = \{\pi_{uv}^{(l)}\}_{v=1}^N$ and $\pi_{uv}^{(l)} = p(v|u) = \tilde{a}_{uv}^{(l)}$. Then in principle, we can sample over the categorical distribution multiple times for each node to obtain its neighbors. However, the sampling process would introduce discontinuity and hinders back-propagation. Fortunately, we notice that the Eqn. 3 can be modified to incorporate the reparametrization trick [16] to allow differentiable learning:

$$\mathbf{z}_u^{(l+1)} = \sum_{v=1}^{N} \frac{\exp((\mathbf{q}_u^\top \mathbf{k}_u + g_v)/\tau)}{\sum_{w=1}^{N}\exp((\mathbf{q}_u^\top \mathbf{k}_w + g_w)/\tau)} \cdot \mathbf{v}_u = \sum_{v=1}^{N} \frac{\kappa(\mathbf{q}_u/\sqrt{\tau}, \mathbf{k}_v/\sqrt{\tau})e^{g_v/\tau}}{\sum_{w=1}^{N}\kappa(\mathbf{q}_u/\sqrt{\tau}, \mathbf{k}_w/\sqrt{\tau})e^{g_w/\tau}} \cdot \mathbf{v}_v, \tag{6}$$

where g_u is i.i.d. sampled from Gumbel distribution and τ is a temperature coefficient. Eqn. 6 is a continuous relaxation of sampling over $\text{Cat}(\boldsymbol{\pi}_u^{(l)})$ and τ controls the closeness to hard discrete samples [22]. Following similar reasoning as Eqn. 3 and 5, we can yield

$$\mathbf{z}_u^{(l+1)} \approx \sum_{v=1}^{N} \frac{\phi(\mathbf{q}_u/\sqrt{\tau})^\top \phi(\mathbf{k}_v/\sqrt{\tau})e^{g_v/\tau}}{\sum_{w=1}^{N}\phi(\mathbf{q}_u/\sqrt{\tau})^\top \phi(\mathbf{k}_w/\sqrt{\tau})e^{g_w/\tau}} \cdot \mathbf{v}_v = \frac{\phi(\mathbf{q}_u/\sqrt{\tau})^\top \sum_{v=1}^{N} e^{g_v/\tau}\phi(\mathbf{k}_v/\sqrt{\tau})\cdot \mathbf{v}_v}{\phi(\mathbf{q}_u/\sqrt{\tau})^\top \sum_{w=1}^{N} e^{g_w/\tau}\phi(\mathbf{k}_w/\sqrt{\tau})}. \tag{7}$$

Eqn. 7 achieves message passing over a sampled latent graph (where we only sample once for each node) and still guarantees linear complexity as Eqn. 5. In practice, we can sample K times (e.g., $K = 5$) for each node and take an average of the aggregated results (see Alg. 1).

3.2 Well-posedness of the Kernelized Gumbel-Softmax Operator

One reasonable concern for Eqn. 7 is whether the RF approximation for kernel functions maintains the well-posedness of Gumbel approximation for the target discrete variables. As a justification for the new message-passing function, we next answer two theoretical questions: 1) How is the approximation capability of RF for the original dot-then-exponentiate operation with Gumbel variables in Eqn. 6? 2) Does Eqn. 7 still guarantee a continuous relaxation of the categorical distributions? We formulate the results as follows and defer proofs to Appendix A.

Theorem 1 (Approximation Error for Softmax-Kernel)**.** *Assume $\|\mathbf{q}_u\|_2$ and $\|\mathbf{k}_v\|_2$ are bounded by r, then with probability at least $1-\epsilon$, the gap $\Delta = \left|\phi(\mathbf{q}_u/\sqrt{\tau})^\top \phi(\mathbf{k}_v/\sqrt{\tau}) - \kappa(\mathbf{q}_u/\sqrt{\tau}, \mathbf{k}_v/\sqrt{\tau})\right|$), where ϕ is defined by Eqn. 4, will be bounded by $\mathcal{O}\left(\sqrt{\frac{\exp(6r/\tau)}{m\epsilon}}\right)$.*

We can see that the error bound of RF for approximating original softmax-kernel function depends on both the dimension of feature map ϕ and temperature τ. Notably, the error bound is independent of node number N, which implies that the approximation ability is insensitive to dataset sizes.

The second question is non-trivial since Eqn. 7 involves randomness of Gumbel variables and random transformation in ϕ, which *cannot* be decoupled apart. We define $c_{uv} = \frac{\phi(\mathbf{q}_u/\sqrt{\tau})^\top \phi(\mathbf{k}_v/\sqrt{\tau}) e^{g_v/\tau}}{\sum_{w=1}^N \phi(\mathbf{q}_u/\sqrt{\tau})^\top \phi(\mathbf{k}_w/\sqrt{\tau}) e^{g_w/\tau}}$ as the result from the kernelized Gumbel-Softmax and $\mathbf{c}_u = \{c_{uv}\}_{v=1}^N$ denotes the sampled edge vector for node u. We can arrive at the result as follows.

Theorem 2 (Property of Kernelized Gumbel-Softmax Random Variables)**.** *Suppose m is sufficiently large, we have the convergence property for the kernelized Gumbel-Softmax operator*

$$\lim_{\tau\to 0} \mathbb{P}(c_{uv} > c_{uv'}, \forall v' \neq v) = \frac{\exp(\mathbf{q}_u^\top \mathbf{k}_v)}{\sum_{w=1}^N \exp(\mathbf{q}_u^\top \mathbf{k}_w)}, \quad \lim_{\tau\to 0} \mathbb{P}(c_{uv} = 1) = \frac{\exp(\mathbf{q}_u^\top \mathbf{k}_v)}{\sum_{w=1}^N \exp(\mathbf{q}_u^\top \mathbf{k}_w)}.$$

The theorem shows that when i) the dimension of feature map is large enough and ii) the temperature goes to zero, the distribution from which latent structures are sampled would converge to the original categorical distribution.

Remark. The two theorems imply a trade-off between RF approximation and Gumbel-Softmax approximation w.r.t. the choice of τ. A large τ would help to reduce the burden on kernel dimension m, and namely, small τ would require a very large m to guarantee enough RF approximation precision. On the other hand, if τ is too large, the weight on each edge will converge to $\frac{1}{N}$, i.e., the model nearly degrades to mean pooling, while a small τ would endow the kernelized Gumbel-Softmax with better approximation to the categorical distribution. Empirical studies are presented in Appendix D.2.

3.3 Input Structures as Relational Bias

Eqn. 7 does not leverage any information from observed geometry which, however, is often recognized important for modeling physically-structured data [3]. We therefore accommodate input topology (if any) as relational bias via modifying the attention weight as $\tilde{a}_{uv}^{(l)} \leftarrow \tilde{a}_{uv}^{(l)} + \mathbb{I}[a_{uv}=1]\sigma(b^{(l)})$, where $b^{(l)}$ is a learnable scalar as relational bias for any adjacent node pairs (u, v) and σ is a certain (bounded) activation function like sigmoid. The relational bias aims at assigning adjacent nodes in $\mathcal{G}$ with proper weights, and the node representations could be accordingly updated by

$$\mathbf{z}_u^{(l+1)} \leftarrow \mathbf{z}_u^{(l+1)} + \sum_{v, a_{uv}=1} \sigma(b^{(l)}) \cdot \mathbf{v}_v. \quad (8)$$

Algorithm 1: Efficient Message Passing with Stochastic Learning of Discrete Structures

Input: Node features $\mathbf{X}$, observed adjacency $\mathbf{A}$.
$\mathbf{Z}^{(0)} \leftarrow \mathbf{X}$;
for $l = 0, 1 \ldots, L-1$ **do**
  for $k = 1, 2, \ldots, K$ **do**
    $\mathbf{q}_u \leftarrow W_Q^{(l)} \mathbf{z}_u^{(l)}, \mathbf{k}_u \leftarrow W_K^{(l)} \mathbf{z}_u^{(l)}$;
    $\mathbf{U}_k^{(l)} \leftarrow \sum_{v=1}^N e^{g_v/\tau} \phi(\mathbf{k}_v/\sqrt{\tau}) \cdot (W_V^{(l)} \mathbf{z}_u^{(l)})^\top$;
    $\mathbf{V}_k^{(l)} \leftarrow \sum_{w=1}^N e^{g_w/\tau} \phi(\mathbf{k}_w/\sqrt{\tau})$;
    $\mathbf{z}_{u,k}^{(l)} \leftarrow \frac{\phi(\mathbf{q}_u/\sqrt{\tau})^\top \mathbf{U}_k^{(l)}}{\phi(\mathbf{q}_u/\sqrt{\tau})^\top \mathbf{V}_k^{(l)}}, \ \forall u$;
  $\mathbf{Z}_k^{(l)} \leftarrow \{\mathbf{z}_{u,k}^{(l)}\}_{u\in\mathcal{N}}, \ \mathbf{Z}^{(l)} \leftarrow \frac{1}{K}\sum_{k=1}^K \mathbf{Z}_k^{(l)}$;
  $\mathbf{z}_u^{(l+1)} \leftarrow \mathbf{z}_u^{(l)} + \sum_{v, a_{uv}=1} \sigma(b^{(l)}) \cdot \mathbf{v}_v, \ \forall u$;
Output: Predicted labels $\hat{\mathbf{Y}} = \mathrm{MLP}(\{\mathbf{Z}^{(l)}\}_{l=0}^L)$.

We summarize the feed-forward computation of ALIGNS in Alg. 1. Eqn. 8 increases the algorithmic complexity for message passing to $\mathcal{O}(N+E)$, albeit within the same order-of-magnitude as common GNNs operating on input graphs. Also, one can consider higher-order adjacency as relational bias for better expressiveness at some expense of efficiency, as similarly done by [1].

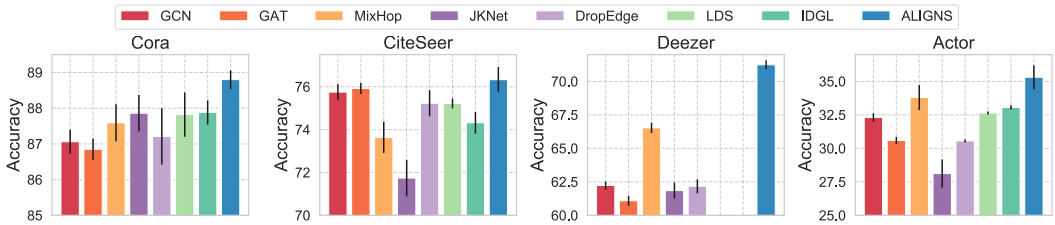


Figure 1: Experimental results for node classification in transductive setting on four common datasets. The missing results on `Deezer` is caused by out-of-memory (OOM).

3.4 Learning Objective

Given node features $\mathbf{X} = \{\mathbf{x}_u\}$ and training labels $\mathbf{Y} = \{y_u\}$, we are to maximize the data log-likelihood which yields a supervised loss (with C classes)

$$\mathcal{L}_s(\mathbf{Y}, \hat{\mathbf{Y}}) = -\frac{1}{N}\sum_{v=1}^{N}\sum_{c=1}^{C}\mathbb{I}[y_u = c]\log \hat{y}_{u,c}, \tag{9}$$

where $\mathbb{I}[\cdot]$ is an indicator function. However, it may not suffice to generalize well. The reason is that the graph topology learning increases the degrees of freedom and the number of training labels is not comparable to that. Therefore, we additionally introduce an edge-level regularization:

$$\mathcal{L}_e(\mathbf{A}, \tilde{\mathbf{A}}) = -\frac{1}{NL}\sum_{l=1}^{L}\sum_{(u,v)\in\mathcal{E}}\frac{1}{d_u}\log \pi_{uv}^{(l)}, \tag{10}$$

where d_u denotes the in-degree of node u and $\pi_{uv}^{(l)}$ is the predicted probability for edge (u, v) at l-th layer. Eqn. 10 is essentially a maximum likelihood estimation (MLE) for edges in $\mathcal{E}$, whose data distribution is defined as

$$p_0(v|u) = \begin{cases} \frac{1}{d_u}, & a_{uv} = 1 \\ 0, & otherwise. \end{cases} \tag{11}$$

We next show how to efficiently obtain $\pi_{uv}^{(l)}$. Although the feed-forward ALIGNS computation defined by Eqn. 7 does not explicitly produce the value for each $\pi_{uv}^{(l)}$, we can query their values by

$$\pi_{uv}^{(l)} = \frac{\phi(W_Q^{(l)}\mathbf{z}_u^{(l)})^\top \phi(W_K^{(l)}\mathbf{z}_v^{(l)})}{\phi(W_Q^{(l)}\mathbf{z}_u^{(l)})^\top \sum_{w=1}^{N}\phi(W_K^{(l)}\mathbf{z}_w^{(l)})}, \tag{12}$$

where the summation term can be re-used from previous computation in Eqn. 7. Therefore, the computation for each $\pi_{uv}^{(l)}$ requires $\mathcal{O}(1)$ complexity, yielding the total complexity controlled within $\mathcal{O}(E)$ (since we only need to query the observed edges). The final objective can be the combination of two: $\mathcal{L} = \mathcal{L}_s + \lambda\mathcal{L}_e$, where λ controls how much emphasis is put on input topology.

4 Evaluation

We consider a diverse set of datasets for experiments and present detailed dataset information in Appendix C. For implementation, we set σ as sigmoid function and τ as 0.25 for all datasets. The output prediction layer is a one-layer MLP. More implementation details are presented in Appendix B. All experiments are conducted on a NVIDIA V100 with 16 GB memory.

As baseline models, we basically consider GCN [19] and GAT [36]. Besides, we compare with some advanced GNN models, including JKNet [41] and MixHop [1]. These GNN models all rely on input graphs. We further consider DropEdge [27] and two SOTA graph structure learning methods, LDS-GNN [11] and IDGL [4] for comparison. More detailed information about these models are presented in Appendix B. All the experiments are repeated five times with different initializations.

4.1 Experiments on Transductive Node Classification

We study supervised node classification in transductive setting on several common graph datasets: `Cora`, `Citeseer`, `Deezer` and `Actor`. The first two have high homophily ratios and the last two

Table 2: Experimental results on `OGB-Proteins` dataset.

Method	ROC-AUC (%)
MLP	72.04 ± 0.48
GCN	72.51 ± 0.35
ALIGNS	**77.45** ± 1.15
ALIGNS-dt	75.50 ± 0.64
ALIGNS-tp	76.18 ± 0.09

Table 3: Results on `Amazon2M` with random mini-batch partition (*batch*) and node clustering (*cluster*) for training on a single GPU.

Method	Accuracy (%)	Method	Accuracy (%)
MLP_{batch}	63.46 ± 0.10	$\text{MLP}_{cluster}$	69.35 ± 0.20
GCN_{batch}	83.90 ± 0.10	$\text{GCN}_{cluster}$	86.06 ± 0.07
ALIGNS_{batch}	**87.85** ± 0.24	$\text{ALIGNS}_{cluster}$	**88.19** ± 0.04
ALIGNS-dt_{batch}	87.02 ± 0.75	$\text{ALIGNS-dt}_{cluster}$	88.13 ± 0.04
ALIGNS-tp_{batch}	87.55 ± 0.11	$\text{ALIGNS-tp}_{cluster}$	88.10 ± 0.09

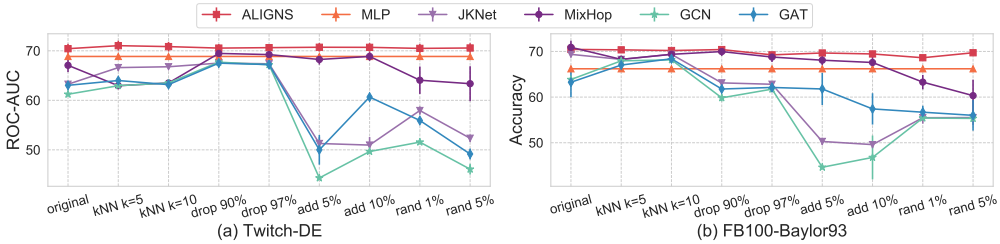


Figure 2: Experimental results for inductive node classification on original/modified testing graphs.

are identified as heterophilic graphs [46, 20]. These datasets are of small or medium sizes (with 2K∼20K nodes). We use random splits with train/valid/test ratios as 50%/25%/25% and accuracy on test nodes for evaluation. Results are plotted in Fig. 1 and ALIGNS achieves the best mean accuracy across four datasets and in particular, outperforms other models by a large margin on two heterophilic graphs. The results indicate that ALIGNS can handle both homophilious and non-homophilious graphs. Compared with two structure learning models LDS and IDGL, ALIGNS yields significantly better performance, which demonstrates the superiority of ALIGNS. Also, for `deezer`, LDS and IDGL suffers from out-of-memory (OOM). In fact, the major difficulty for `deezer` is the large dimensions of input node features (nearly 30K), which causes OOM for IDGL even with the anchor approximation. In contrast, ALIGNS manages to scale and produce desirable accuracy.

4.2 Experiments on Larger Graph Datasets

To further test the scalability of ALIGNS, we consider two large-sized networks, `OGB-Proteins` and `Amazon2M`, with over 0.1 million and 2 million of nodes, respectively. `OGB-Proteins` is a multi-task dataset with 112 output dimensions, while `Amazon2M` is extracted from the Amazon Co-Purchasing network that entails long-range dependence [13]. For `OGB-Proteins`, we use the protocol of [15] and ROC-AUC for evaluation. For `Amazon2M`, unlike [5] using random splits, we consider a more challenging setting: we rank the nodes according to their degrees, and use the top 10% for training and the rest for testing. This introduces distribution shifts and is more alike to real situations where frequent items are often used for training a model that needs to handle infrequent testing products. For validation set, we hold out 20% data from the training set. In Table 2, we present the results on `OGB-Proteins` where we only compare our model with MLP and GCN since other competitors would suffer due to OOM. ALIGNS yields much better ROC-AUC than the baselines. Table 3 reports the results on `Amazon2M`. Due to the large dataset size that is even prohibitive for GCN, we adopt mini-batch partition for training in two different ways. 1) *Batch*: randomly partition nodes into mini-batch sized 100000; 2) *Cluster*: use the METIS algorithm [5] for graph partition. For ALIGNS, we only consider structure learning among nodes in a mini-batch, which is helpful for saving space and costs only 5GB GPU memory for training on `Amazon2M`. Table 3 shows that ALIGNS outperforms baselines by a large margin, which demonstrates its practical efficacy on large-scale datasets and also the capability for addressing long-range dependence with shallow layers (we set $L = 3$).

4.3 Experiments on Inductive Node Classification

We next study inductive learning, where the model needs to handle unseen nodes with new edges during inference, over two social networks `Twitch-DE` and `Facebook-Baylor93`. For each dataset, we use the clustering algorithm to split all the nodes into five subgraphs and adopt 3/1/1 subgraphs for train/valid/test. Following [20] we use ROC-AUC and accuracy as evaluation metric for `Twitch` and `Facebook`, respectively. Fig. 2 presents the results where we study models' performance on original and modified testing graphs, including: 1) randomly drop 90% and 97% edges; 2) randomly adding 5% and 10% noised edges; 3) randomly connecting nodes with 1% and 5% probabilities; 4)

Table 4: Experimental results on semi-supervised classficiation on `Mini-ImageNet` and `20News-Groups` where we use k-NN (with different k's) for artificially constructing an input graph.

Method	`Mini-ImageNet`				`20News-Group`			
	$k=5$	$k=10$	$k=15$	$k=20$	$k=5$	$k=10$	$k=15$	$k=20$
GCN	84.86 ± 0.42	85.61 ± 0.40	85.93 ± 0.59	85.96 ± 0.66	65.98 ± 0.68	64.13 ± 0.88	62.95 ± 0.70	62.59 ± 0.62
GAT	84.70 ± 0.48	85.24 ± 0.42	85.41 ± 0.43	85.37 ± 0.51	64.06 ± 0.44	62.51 ± 0.71	61.38 ± 0.88	60.80 ± 0.59
DropEdge	83.91 ± 0.24	85.35 ± 0.44	85.25 ± 0.63	85.81 ± 0.65	64.46 ± 0.43	64.01 ± 0.42	62.46 ± 0.51	62.68 ± 0.71
IDGL	83.63 ± 0.32	84.41 ± 0.35	85.50 ± 0.24	85.66 ± 0.42	65.09 ± 1.23	63.41 ± 1.26	61.57 ± 0.52	62.21 ± 0.79
LDS	OOM	OOM	OOM	OOM	**66.15** ± 0.36	64.70 ± 1.07	63.51 ± 0.64	63.51 ± 1.75
ALIGNS	**86.77** ± 0.45	**86.74** ± 0.23	**86.87** ± 0.41	**86.64** ± 0.42	66.01 ± 1.18	**65.21** ± 1.14	**64.69** ± 1.31	**64.55** ± 0.97
ALIGNS w/o reg	86.44 ± 0.23	86.54 ± 0.40	86.67 ± 0.45	86.60 ± 0.30	65.96 ± 1.36	64.70 ± 1.07	64.64 ± 1.42	64.48 ± 1.00
ALIGNS w/o rb	86.36 ± 0.36	86.44 ± 0.21	86.24 ± 0.44	85.76 ± 0.40	64.82 ± 1.20	64.18 ± 1.45	64.43 ± 1.38	64.14 ± 1.23

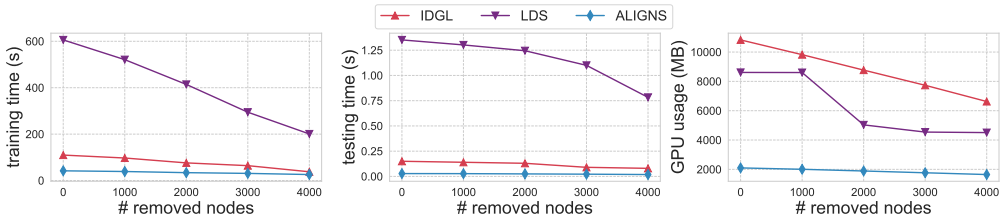


Figure 3: Comparison of training/inference time and GPU memory cost w.r.t. different instance numbers (by removing a certain portion of nodes) on `20News-Groups`.

constructing a k-NN graph with input node features ($k = 5$ or 10). The figures show that ALIGNS outperforms all the competitors in nearly all the cases. Notably, ALIGNS yields much more stable performance than other models w.r.t. different testing structures. In particular, GCN and GAT are quite vulnerable for perturbation on structures which essentially introduce distribution shift between training and testing graphs. JKNet and MixHop could alleviate the sensitivity to a certain degree, yet still suffer when replacing the input graph as randomly connected ones. With new structures learned from data, ALIGNS can perform much more stable than standard GNNs relying on input graphs.

4.4 Experiments on Graph-Enhanced Applications

We proceed to apply our model to semi-supervised image and text classification on `Mini-ImageNet` and `20News-Groups` datasets, without any given input graphs. The instances of `Mini-ImageNet` [37] are 84×84 RGB images and we randomly choose 30 classes each of which contains 600 samples for experiments. `20News-Groups` [24] consists of nearly 10K texts whose features are extracted by TF-IDF. More details for preprocessing are presented in Appendix C. Also, for each dataset, we randomly split instances into 50%/25%/25% for train/valid/test. To enable message passing (especially for baseline models), we use k-NN (over input node features) for artificially constructing a graph. Table 4 presents the comparison results under different k's. We can see that ALIGNS achieves the best performance in seven cases out of eight in total. The performance of baseline models varies significantly with different k values, and ALIGNS is much less sensitive.

4.5 Further Discussions

Comparison of Time/Space Consumption. Fig. 3 plots training/inference time and GPU memory costs of ALIGNS and two SOTA structure learning models. Compared with LDS, ALIGNS reduces the training time, inference time, memory cost by up to 93.1%, 97.9%, 75.6%, respectively; compared with IDGL (using anchor-based approximation for acceleration), ALIGNS reduces the training time, inference time, memory cost by up to 61.8%, 80.8%, 80.6%, respectively.

Ablation on Edge Loss and Relational Bias. We study the effects of edge-level regularization and relation bias as ablation study shown in Table 4. More results are deferred to Appendix D.1 and they consistently show that both components contribute to some positive effects and suggest that our edge-level loss and relation bias can both help to leverage useful information from input graphs.

Ablation on Stochastic Components. Table 2 and 3 also include two variants of ALIGNS for ablation study. 1) ALIGNS-dt: replace Gumbel-Softmax by original Softmax (with temperature 1.0) for deterministic propagation; 2) ALIGNS-tp: use original Softmax with temperature set as 0.25 (the same as ALIGNS). There is more or less performance drop when removing the Gumbel components. The possible cause stems from over-normalizing or over-fitting that are amplified in large datasets, as we discussed in Section 3.1 and the kernelized Gumbel-Softmax operator shows its effectiveness.

Impact of Temperature and Feature Map Dimension. We study the effects of τ and m in Appendix D.2 and the variation trend accords with our theoretical analysis in Section 3.2.

Visualization and Implications. We visualize the produced structures by ALIGNS on different datasets in Appendix D.3. There are some common and special patterns, like dense connection between inter-class nodes and sparse connection between inter-class nodes. This suggests that the latent graphs can propagate useful information for better distinguishing nodes with distinct classes, which helps to interpret why ALIGNS can improves prediction. We also compare the learned structures with original graphs and the results turn out to be quite different across homophilous and heterophilic graph datasets. Also we found ALIGNS tends to assign large edge weights for a small portion of dominant nodes. This can be an interesting discovery that implies that some nodes may be important anchors or serve as pivots for enhancing the whole connectivity of the dataset.

5 Why ALIGNS Improves Downstream Prediction?

There remains a natural question concerning our learning process: how effective can the learned latent topology be for downstream tasks? We next dissect the rationale from a Bayesian perspective. In fact, our model induces a predictive distribution $p(\mathbf{Y}, \tilde{\mathbf{A}}|\mathbf{X}, \mathbf{A}) = p(\tilde{\mathbf{A}}|\mathbf{X}, \mathbf{A})p(\mathbf{Y}|\tilde{\mathbf{A}}, \mathbf{X}, \mathbf{A})$ where we can treat the estimated graph $\tilde{\mathbf{A}}$ as a latent variable.[1] Specifically, $p(\tilde{\mathbf{A}}|\mathbf{X}, \mathbf{A})$ is instantiated with the structure estimation module and $p(\mathbf{Y}|\tilde{\mathbf{A}}, \mathbf{X}, \mathbf{A})$ is instantiated with the feature propagation module. In principle, ideal latent graphs should account for downstream tasks and maximize the potentials of message passing for producing informative node representations. Thus, optimal latent graphs presumably come from the posterior $p(\tilde{\mathbf{A}}|\mathbf{Y}, \mathbf{X}, \mathbf{A}) = \frac{p(\mathbf{Y}|\mathbf{X},\mathbf{A},\tilde{\mathbf{A}})p(\tilde{\mathbf{A}}|\mathbf{X},\mathbf{A})}{\int_{\mathbf{Y}} p(\mathbf{Y}|\mathbf{X},\mathbf{A},\tilde{\mathbf{A}})p(\tilde{\mathbf{A}}|\mathbf{X},\mathbf{A})d\mathbf{Y}}$ which is given by Bayes theorem. Unfortunately, such a posterior is unknown and intractable for the integration.

A Variational Perspective. An intriguing conclusion stems from another view into the learning process: we can treat the structure estimation as a variational distribution $q(\tilde{\mathbf{A}}|\mathbf{X}, \mathbf{A})$ and our learning objective in Section 3.4 can be viewed as the embodiment of a minimization problem over the predictive and variational distributions via

$$p^*, q^* = \arg\min_{p,q} \underbrace{-\mathbb{E}_q[\log p(\mathbf{Y}|\tilde{\mathbf{A}}, \mathbf{X}, \mathbf{A})]}_{\mathcal{L}_s} + \underbrace{\mathcal{D}(q(\tilde{\mathbf{A}}|\mathbf{X}, \mathbf{A})\|p_0(\tilde{\mathbf{A}}|\mathbf{X}, \mathbf{A}))}_{\mathcal{L}_e}, \tag{13}$$

where $\mathcal{D}$ denotes the Kullback-Leibler divergence. Specifically, the *predictive* term is equivalent to minimizing the supervised loss (with Gumbel-Softmax as a surrogate for sampling-based estimates over $q(\tilde{\mathbf{A}}|\mathbf{X}, \mathbf{A})$), and the KL *regularization* term is embodied with the edge-level MLE loss (Eqn. 10) (if we define the prior distribution $p_0(\tilde{\mathbf{A}}|\mathbf{X}, \mathbf{A})$ following Eqn. 11). One may notice that Eqn. 13 is essentially the Evidence Lower Bound (EBLO) for the log-likelihood $\log p(\mathbf{Y}|\mathbf{X}, \mathbf{A})$.

Proposition 1. *Assume q can exploit arbitrary distributions over $\tilde{\mathbf{A}}$. When Eqn. 13 achieves the optimum, we have 1) $\mathcal{D}(q(\tilde{\mathbf{A}}|\mathbf{X}, \mathbf{A})\|p(\tilde{\mathbf{A}}|\mathbf{Y}, \mathbf{X}, \mathbf{A})) = 0$ and 2) $\log p(\mathbf{Y}|\mathbf{X}, \mathbf{A})$ is maximized.*

The proposition indicates that our adopted learning objective intrinsically minimizes the divergence between latent graphs generated by the model and the samples from the posterior $p(\tilde{\mathbf{A}}|\mathbf{Y}, \mathbf{X}, \mathbf{A})$ that ideally helps to propagate useful adjacent information w.r.t. downstream tasks. Therefore, a well-trained network of ALIGNS on labeled data could produce effective latent topology that contributes to boosting the downstream performance.

6 Conclusion

This paper proposes a new scalable and efficient message-passing scheme that can propagate layer-wise node signals between arbitrary pairs beyond explicit topology. The key module, a kernelized Gumbel-Softmax operator, enables us to learn layer-specific latent topology with linear algorithmic complexity without compromising the precision in practice. We also provide accompanying theory as justification for the model design and interpretations that may help to understand its principles. The promising results on diverse datasets and situations verify the effectiveness, scalability, and stability for graph-structured data with distinct properties and also the absence of input graphs.

[1] We assume one latent graph to simplify the illustration though we practically learn layer-specific graphs for each layer of ALIGNS. The analysis can be trivially extended to such a case.

References

[1] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *International Conference on Machine Learning*, pages 21–29, 2019.

[2] Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations*, 2021.

[3] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *CoRR*, abs/1611.08097, 2016.

[4] Yu Chen, Lingfei Wu, and Mohammed J. Zaki. Iterative deep graph learning for graph neural networks: Better and robust node embeddings. In *Advances in Neural Information Processing Systems*, 2020.

[5] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 257–266, 2019.

[6] Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamás Sarlós, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J. Colwell, and Adrian Weller. Rethinking attention with performers. In *International Conference on Learning Representations*, 2021.

[7] Luca Cosmo, Anees Kazi, Seyed-Ahmad Ahmadi, Nassir Navab, and Michael M. Bronstein. Latent patient network learning for automatic diagnosis. *CoRR*, abs/2003.13620, 2020.

[8] Hanjun Dai, Zornitsa Kozareva, Bo Dai, Alexander J. Smola, and Le Song. Learning steady-states of iterative algorithms over graphs. In *International Conference on Machine Learning*, pages 1114–1122, 2018.

[9] Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *CoRR*, abs/2012.09699, 2020.

[10] Pantelis Elinas, Edwin V. Bonilla, and Louis C. Tiao. Variational inference for graph convolutional networks in the absence of graph data and adversarial settings. In *Advances in Neural Information Processing Systems*, 2020.

[11] Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He. Learning discrete structures for graph neural networks. In *International Conference on Machine Learning*, pages 1972–1982, 2019.

[12] Chen Gao, Jinyu Chen, Si Liu, Luting Wang, Qiong Zhang, and Qi Wu. Room-and-object aware knowledge reasoning for remote embodied referring expression. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3064–3073, 2021.

[13] Fangda Gu, Heng Chang, Wenwu Zhu, Somayeh Sojoudi, and Laurent El Ghaoui. Implicit graph neural networks. In *Advances in Neural Information Processing Systems*, 2020.

[14] William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017.

[15] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In *Advances in Neural Information Processing Systems*, 2020.

[16] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017.

[17] Bo Jiang, Ziyan Zhang, Doudou Lin, Jin Tang, and Bin Luo. Semi-supervised learning with graph learning-convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 11313–11320, 2019.

[18] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. Graph structure learning for robust graph neural networks. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 66–74, 2020.

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

[20] Derek Lim, Xiuyu Li, Felix Hohne, and Ser-Nam Lim. New benchmarks for learning on non-homophilous graphs. *CoRR*, abs/2104.01404, 2021.

[21] Dongsheng Luo, Wei Cheng, Wenchao Yu, Bo Zong, Jingchao Ni, Haifeng Chen, and Xiang Zhang. Learning to drop: Robust graph neural network via topological denoising. In *ACM International Conference on Web Search and Data Mining*, pages 779–787, 2021.

[22] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*, 2017.

[23] Julian J. McAuley, Rahul Pandey, and Jure Leskovec. Inferring networks of substitutable and complementary products. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2015.

[24] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[25] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *International Conference on Learning Representations*, 2020.

[26] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, pages 1177–1184, 2007.

[27] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*, 2020.

[28] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *J. Complex Networks*, 9(2), 2021.

[29] Benedek Rozemberczki and Rik Sarkar. Characteristic functions on graphs: Birds of a feather, from statistical descriptors to parametric models. In *ACM International Conference on Information and Knowledge Management*, pages 1325–1334, 2020.

[30] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W. Battaglia. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning*, pages 8459–8468, 2020.

[31] Victor Garcia Satorras and Joan Bruna Estrach. Few-shot learning with graph neural networks. In *International Conference on Learning Representations*, 2018.

[32] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[33] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Mag.*, 29(3):93–106, 2008.

[34] Qingyun Sun, Jianxin Li, Hao Peng, Jia Wu, Xingcheng Fu, Cheng Ji, and Philip S. Yu. Graph structure learning with variational information bottleneck. *CoRR*, abs/2112.08903, 2021.

[35] Amanda L. Traud, Peter J. Mucha, and Mason A. Porter. Social structure of facebook networks. *Physica A: Statistical Mechanics and its Applications*, 391(16):4165–4180, Aug 2012.

[36] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[37] Oriol Vinyals, Charles Blundell, Tim Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638, 2016.

[38] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph CNN for learning on point clouds. *ACM Trans. Graph.*, 38(5):146:1–146:12, 2019.

[39] Tailin Wu, Hongyu Ren, Pan Li, and Jure Leskovec. Graph information bottleneck. In *Advances in Neural Information Processing Systems*, 2020.

[40] Xuan Wu, Lingxiao Zhao, and Leman Akoglu. A quest for structure: Jointly learning the graph structure and semi-supervised classification. In *ACM International Conference on Information and Knowledge Management*, pages 87–96, 2018.

[41] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning*, pages 5449–5458, 2018.

[42] Liang Yao, Chengsheng Mao, and Yuan Luo. Graph convolutional networks for text classification. In *AAAI Conference on Artificial Intelligence*, pages 7370–7377, 2019.

[43] Yingxue Zhang, Soumyasundar Pal, Mark Coates, and Deniz Üstebay. Bayesian graph convolutional neural networks for semi-supervised classification. In *AAAI Conference on Artificial Intelligence*, pages 5829–5836, 2019.

[44] Cheng Zheng, Bo Zong, Wei Cheng, Dongjin Song, Jingchao Ni, Wenchao Yu, Haifeng Chen, and Wei Wang. Robust graph representation learning via neural sparsification. In *International Conference on Machine Learning*, pages 11458–11468, 2020.

[45] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. In *Advances in Neural Information Processing Systems*, 2020.

[46] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. In *Advances in Neural Information Processing Systems*, 2020.

[47] Yanqiao Zhu, Weizhi Xu, Jinghao Zhang, Qiang Liu, Shu Wu, and Liang Wang. Deep graph structure learning for robust representations: A survey. *CoRR*, abs/2103.03036, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix A
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Appendix C for dataset information. See the supplementary for the codes. They will be released public upon publication
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix B
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See the experiment section
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix B
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Appendix C
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Appendix

A Proof for Technical Results

A.1 Proof for Theorem 1

To prove our theorem, we first introduce the following lemma given by the Lemma 2 in [6].

Proposition 1. *Denote a softmax kernel as* $SM(\mathbf{x}, \mathbf{y}) = \exp(\mathbf{x}^\top \mathbf{y})$. *The Positive Random Features defined by Eqn. 4 for softmax-kernel estimation, i.e.,* $\widehat{SM}_m(\mathbf{x}, \mathbf{y}) = \frac{1}{m}\sum_{i=1}^{m}[\exp(\mathbf{w}_i^\top \mathbf{x} - \frac{\|\mathbf{x}\|^2}{2})\exp(\mathbf{w}_i^\top \mathbf{y} - \frac{\|\mathbf{y}\|^2}{2})]$, *has the mean and variance over* $\mathbf{w} \sim \mathcal{N}(0, I_d)$ *as*

$$\begin{aligned}\mathbb{E}_{\mathbf{w}}(\widehat{SM}_m(\mathbf{x}, \mathbf{y})) =& SM(\mathbf{x}, \mathbf{y}) = \exp(\mathbf{x}^\top \mathbf{y}),\\ \mathbb{V}_{\mathbf{w}}(\widehat{SM}_m(\mathbf{x}, \mathbf{y})) =& \frac{1}{m}\exp(\|\mathbf{x}+\mathbf{y}\|^2) SM^2(\mathbf{x}, \mathbf{y})\\ & (1 - \exp(-\|\mathbf{x}+\mathbf{y}\|^2)).\end{aligned} \tag{14}$$

The lemma shows that the Positive Random Features can achieve unbiased approximation for the softmax kernel with a quantified variance.

Back to our main theorem, suppose the L2-norms of $\mathbf{q}_u$ and $\mathbf{k}_v$ are bounded by r, we can derive the probability using the Chebyshev's inequality:

$$\mathbb{P}(\Delta \leq \sqrt{\frac{\exp(6r/\tau)}{m\epsilon}}) \geq 1 - \frac{\mathbb{V}_{\mathbf{w}}(\widehat{\mathrm{SM}}_m(\mathbf{q}_u/\sqrt{\tau}, \mathbf{k}_v/\sqrt{\tau}))}{\exp(6r/\tau)/m\epsilon} \tag{15}$$

where $\Delta = \left|\widehat{\mathrm{SM}}_m(\mathbf{q}_u/\sqrt{\tau}, \mathbf{k}_v/\sqrt{\tau}) - \mathrm{SM}(\mathbf{q}_u/\sqrt{\tau}, \mathbf{k}_v/\sqrt{\tau})\right|$ denotes the deviation of the kernel approximation. Using the result in Lemma 1, we can further obtain that the RHS of Eqn. 15 is no greater than

$$1 - \epsilon\exp(\|\frac{\mathbf{q}_u + \mathbf{k}_v}{\sqrt{\tau}}\|^2 + 2\frac{\mathbf{q}_u^\top \mathbf{k}_v}{\tau} - 6\frac{r}{\tau}). \tag{16}$$

Since $\|\frac{\mathbf{q}_u+\mathbf{k}_v}{\sqrt{\tau}}\|^2 \leq \frac{4r}{\tau}$ and $2\frac{\mathbf{q}_u^\top \mathbf{k}_v}{\tau} \leq \frac{2r}{\tau}$, we can achieve the stated result:

$$\mathbb{P}(\Delta \leq \sqrt{\frac{\exp(6r/\tau)}{m\epsilon}}) \geq 1 - \epsilon. \tag{17}$$

A.2 Proof for Theorem 2

Before entering the proof for the theorem, we first introduce two basic technical lemmas. While such results are already mentioned in previous studies [16, 22], their proofs will be useful for the subsequent reasoning. Therefore, we restate the proofs as building blocks for the following presentation.

Proposition 2. *Given real numbers* $x_i, x_j \in \mathbb{R}$ *and* u_i, u_j *i.i.d. sampled from uniform distribution within* $(0, 1)$. *With Gumbel perturbation defined as* $g(u) = -\log(-\log(u))$, *we have the probability*

$$P(x_i + g(u_i) > x_j + g(u_j)) = \frac{1}{1 + \exp\left(-(x_i - x_j)\right)}.$$

Proof. Due to $g(u) = -\log(-\log(u))$, the inequality of interests $x_i + g(u_i) > x_j + g(u_j)$ can be rearranged as

$$e^{x_i - x_j} > \frac{\log(u_i)}{\log(u_j)}. \tag{18}$$

Since $\log(u_j) < 0$, Eqn. 18 can be written as

$$u_j < u_i^{e^{x_j - x_i}}. \tag{19}$$

As u_i, u_j are i.i.d. sampled from a uniform distribution, the probability when the above formula can be calculated via:

$$\begin{aligned}\int_0^1 \int_0^{u_i^{e^{x_j - x_i}}} du_j du_i &= \int_0^1 u_i^{e^{x_j - x_i}} du_i \\ &= \frac{1}{1 + \exp(-(x_i - x_j))}.\end{aligned} \tag{20}$$

Thus, we conclude the proof with

$$P(x_i + g(u_i) > x_j + g(u_j)) = \frac{1}{1 + \exp\left(-(x_i - x_j)\right)}. \tag{21}$$

□

Proposition 3. *Let $X \sim$ Gumbel(α, τ) (i.e. $X_k = \frac{\exp((\log \alpha_k + g_k)/\tau)}{\sum_{i=1}^n \exp((\log \alpha_i + g_i)/\tau)}$) with location parameters $\alpha \in (0, \infty)^n$ and temperature $\tau \in (0, \infty)$, then:*

- $P(X_k > X_i, \forall i \neq k) = \frac{\alpha_k}{\sum_{i=1}^n \alpha_i}$,
- $P(\lim_{\tau \to 0} X_k = 1) = \frac{\alpha_k}{\sum_{i=1}^n \alpha_i}$.

Proof. This result can be similarly proved as Lemma 2. The event of interests $X_k > X_i, \forall i \neq k$ is equivalent to

$$\begin{aligned}&\log \alpha_k - \log(-\log u_k) > \log \alpha_1 - \log(-\log u_1), \\ &\log \alpha_k - \log(-\log u_k) > \log \alpha_2 - \log(-\log u_2), \\ &\qquad\qquad ... \\ &\log \alpha_k - \log(-\log u_k) > \log \alpha_n - \log(-\log u_n).\end{aligned} \tag{22}$$

Since all the above inequalities are independent given u_k, we can rearrange the first inequality as

$$u_1 < u_k^{\alpha_1/\alpha_k} \leq 1. \tag{23}$$

Since $u_1 \sim U[0, 1]$, the probability for the first inequality in Eqn. 22 being true would be $u_k^{\alpha_1/\alpha_k}$. Thus, the probability for Eqn. 22 being true can be calculated via

$$u_k^{\alpha_1/\alpha_k} u_k^{\alpha_2/\alpha_k} ... g_k^{\alpha_n/\alpha_k} = g_k^{(\alpha_1 + \alpha_2 + ... + \alpha_n)/\alpha_k} = g_k^{(1/\alpha_k) - 1}. \tag{24}$$

For simplicity, we assume $\sum_{i=1}^n \alpha_i = 1$. Then for any $g_k \in [0, 1]$, we obtain

$$\begin{aligned}P(X_k > X_i, \forall i \neq k) &= \int_0^1 g_k^{(1/\alpha_k) - 1} dg_k \\ &= \frac{\alpha_k}{\sum_{i=1}^n \alpha_i},\end{aligned} \tag{25}$$

and arrive at the result for the first bullet point. For the second bullet point, when $\tau \to 0$, we have

$$\begin{aligned}&\lim_{\tau \to 0} \frac{\exp((\log \alpha_i + g_i)/\tau)}{\exp((\log \alpha_j + g_j)/\tau)} \\ &= \lim_{\tau \to 0} \exp((\log \alpha_i + g_i - \log \alpha_j - g_j)/\tau) \\ &= \begin{cases} \infty, & \text{if } \alpha_i > \alpha_j \\ 0, & otherwise. \end{cases}\end{aligned} \tag{26}$$

Such a fact indicates that the output of a Concrete distribution with $\tau \to 0$ will be a one-hot vector ($X_{\arg\max_i \alpha_i} = 1$). This yields the conclusion that

$$P(\lim_{\tau \to 0} X_k = 1) = P(X_k > X_i, \forall i \neq k) = \frac{\alpha_k}{\sum_{i=1}^n \alpha_i}. \tag{27}$$

□

Now we turn to the proof of our theorem. We are to prove that the kernelized form in Eqn. 7 has the same property as the original Gumbel-Softmax in the limit sense (when τ goes to zero). We recall that we have defined $\mathbf{q}_u = W_Q^{(l)}\mathbf{z}_u^{(l)}$, $\mathbf{k}_u = W_K^{(l)}\mathbf{z}_u^{(l)}$ and $\mathbf{v}_u = W_V^{(l)}\mathbf{z}_u^{(l)}$ for simplicity.

First, by definition we have

$$\begin{aligned}&\phi(\frac{\mathbf{q}_u}{\sqrt{\tau}})^\top \phi(\frac{\mathbf{k}_v}{\sqrt{\tau}})e^{\frac{g_v}{\tau}}\\ =&\frac{1}{m}\exp(-\frac{||\frac{\mathbf{q}_u}{\sqrt{\tau}}||^2+||\frac{\mathbf{k}_v}{\sqrt{\tau}}||^2}{2})\sum_{i=1}^{m}\exp(\omega_i^\top(\frac{\mathbf{q}_u}{\sqrt{\tau}}+\frac{\mathbf{k}_v}{\sqrt{\tau}})+\frac{g_v}{\tau}).\end{aligned} \tag{28}$$

The property holds that for $\forall w \neq v$, we have $\lim_{\tau\to 0}\frac{\phi(\frac{\mathbf{q}_u}{\sqrt{\tau}})^\top\phi(\frac{\mathbf{k}_v}{\sqrt{\tau}})e^{\frac{g_v}{\tau}}}{\phi(\frac{\mathbf{q}_u}{\sqrt{\tau}})^\top\phi(\frac{\mathbf{k}_w}{\sqrt{\tau}})e^{\frac{g_w}{\tau}}}$ equals to ∞ or 0, i.e. the output of the kernelized Gumbel-Softmax is still a one-hot vector when $\tau \to 0$. Let

$$Y_v = \frac{\phi(\frac{\mathbf{q}_u}{\sqrt{\tau}})^\top\phi(\frac{\mathbf{k}_v}{\sqrt{\tau}})e^{\frac{g_v}{\tau}}}{\sum_{w=1}^{N}\phi(\frac{\mathbf{q}_u}{\sqrt{\tau}})^\top\phi(\frac{\mathbf{k}_w}{\sqrt{\tau}})e^{\frac{g_w}{\tau}}}. \tag{29}$$

Here Y_v is defined in the same way as c_{uv} in Section 3.2. We thus have $P(\lim_{\tau\to 0} Y_v = 1) = P(Y_v > Y_{v'}, \forall v' \neq v)$.

To compute $P(Y_v > Y_{v'}, \forall v' \neq v)$, for simplicity, let us consider the probability $P(Y_v > Y_{v'}) = P(\phi(\frac{\mathbf{q}_u}{\sqrt{\tau}})^\top\phi(\frac{\mathbf{k}_v}{\sqrt{\tau}})e^{\frac{g_v}{\tau}} > \phi(\frac{\mathbf{q}_u}{\sqrt{\tau}})^\top\phi(\frac{\mathbf{k}_{v'}}{\sqrt{\tau}})e^{\frac{g_{v'}}{\tau}})$. To keep notation clean, we define

$$\beta_v = \phi(\frac{\mathbf{q}_u}{\sqrt{\tau}})^\top\phi(\frac{\mathbf{k}_v}{\sqrt{\tau}}),\ \beta_{v'} = \phi(\frac{\mathbf{q}_u}{\sqrt{\tau}})^\top\phi(\frac{\mathbf{k}_{v'}}{\sqrt{\tau}}). \tag{30}$$

Then the above-mentioned probability can be rewritten as $P(\log\beta_v + \frac{g_v}{\tau} > \log\beta_{v'} + \frac{g_{v'}}{\tau})$, where β_v and $\beta_{v'}$ are two i.i.d. random variables.

From Lemma 1, we have $\mathrm{E}(\beta_v) = \exp(\mathbf{q}_u^\top\mathbf{k}_v/\tau) = \alpha_v^{\frac{1}{\tau}}$, $\mathrm{E}(\beta_{v'}) = \exp(\mathbf{q}_u^\top\mathbf{k}_{v'}/\tau) = \alpha_{v'}^{\frac{1}{\tau}}$, where α_v and $\alpha_{v'}$ are two constant values. Then using Lemma 2, we have

$$\begin{aligned}&P(\log\alpha_v^{1/\tau} + \frac{g_v}{\tau} > \log\alpha_{v'}^{1/\tau} + \frac{g_{v'}}{\tau})\\ =&P(\log\alpha_v + g_v > \log\alpha_{v'} + g_{v'})\\ =&\frac{1}{1+\exp(\log\alpha_{v'} - \log\alpha_v)}\\ =&\frac{\alpha_{v'}}{\alpha_v + \alpha_{v'}}.\end{aligned} \tag{31}$$

According to the Chebyshev's inequality, we have $P(|\beta_v - \alpha_v^{\frac{1}{\tau}}| \leq \epsilon_v) \geq 1 - \frac{\sigma_v^2}{\epsilon_v^2}$. Here $\sigma_v^2 = \mathbb{V}_{\mathbf{w}}(\widehat{\mathrm{SM}}_m(\frac{\mathbf{q}_u}{\sqrt{\tau}}, \frac{\mathbf{k}_j}{\sqrt{\tau}}))$, which can given by Lemma 1.

Due to the convexity of logarithmic function, we have

$$\frac{|\log\beta_v - \frac{1}{\tau}\log\alpha_v|}{|\beta_v - \alpha_v^{\frac{1}{\tau}}|} \leq \frac{1}{\alpha_v^{\frac{1}{\tau}} - \epsilon_v}, \tag{32}$$

and subsequently,

$$\begin{aligned}|\log\beta_v - \frac{1}{\tau}\log\alpha_v| &\leq \frac{|\beta_v - \alpha_v^{\frac{1}{\tau}}|}{\alpha_v^{\frac{1}{\tau}} - \epsilon_v}\\ &\leq \frac{\epsilon_v}{\alpha_v^{\frac{1}{\tau}} - \epsilon_v}.\end{aligned} \tag{33}$$

Therefore we have $P(|\log\beta_v - \frac{1}{\tau}\log\alpha_v| \leq \frac{\epsilon_v}{\alpha_v^{\frac{1}{\tau}} - \epsilon_v}) \geq P(|\beta_v - \alpha_v^{\frac{1}{\tau}}| \leq \epsilon_v)$. Based on this, we can derive the result:

$$P(|\log\beta_v - \frac{1}{\tau}\log\alpha_v| \leq \frac{\epsilon_v}{\alpha_v^{\frac{1}{\tau}} - \epsilon_v}) \geq 1 - \frac{\sigma_v^2}{\epsilon_v^2}. \tag{34}$$

Since β_v and $\beta_{v'}$ are two i.i.d. random variables, we have

$$\begin{aligned} &P(|\log\beta_v - \frac{1}{\tau}\log\alpha_v| \leq \frac{\epsilon_v}{\alpha_v^{\frac{1}{\tau}} - \epsilon_v}, \\ &|\log\beta_{v'} - \frac{1}{\tau}\log\alpha_{v'}| \leq \frac{\epsilon_{v'}}{\alpha_{v'}^{\frac{1}{\tau}} - \epsilon_{v'}}) \geq (1 - \frac{\sigma_v^2}{\epsilon_v^2})(1 - \frac{\sigma_{v'}^2}{\epsilon_{v'}^2}). \end{aligned} \tag{35}$$

For simplicity, we denote $\epsilon = \frac{\epsilon_v}{\alpha_v^{\frac{1}{\tau}} - \epsilon_v} + \frac{\epsilon_{v'}}{\alpha_{v'}^{\frac{1}{\tau}} - \epsilon_{v'}}$ and $P_\epsilon = (1 - \frac{\sigma_v^2}{\epsilon_v^2})(1 - \frac{\sigma_{v'}^2}{\epsilon_{v'}^2})$. We therefore have

$$P(|\log\beta_v - \frac{1}{\tau}\log\alpha_v| + |\log\beta_{v'} - \frac{1}{\tau}\log\alpha_{v'}| \leq \epsilon) \geq P_\epsilon. \tag{36}$$

Using the triangular inequality, we can yield

$$\begin{aligned} &|(\log\beta_v - \frac{1}{\tau}\log\alpha_v) - (\log\beta_{v'} - \frac{1}{\tau}\log\alpha_{v'})| \\ \leq &|\log\beta_v - \frac{1}{\tau}\log\alpha_v| + |\log\beta_{v'} - \frac{1}{\tau}\log\alpha_{v'}|. \end{aligned} \tag{37}$$

Combining Eqn. 36 and 37, we have

$$P(|(\log\beta_v - \frac{1}{\tau}\log\alpha_v) - (\log\beta_{v'} - \frac{1}{\tau}\log\alpha_{v'})| \leq \epsilon) \geq P_\epsilon. \tag{38}$$

Let $c = \log\beta_v - \log\beta_{v'}$, so that $\mathrm{E}(c) = \frac{1}{\tau}(\log\alpha_v - \log\alpha_{v'})$. From Eqn. 38, we can obtain

$$P(c \geq \mathrm{E}(c) - \epsilon) \geq P_\epsilon. \tag{39}$$

According to Lemma 2, the probability $P(\mathrm{E}(c) - \epsilon \geq \frac{g_{v'} - g_v}{\tau})$ can be written as

$$\begin{aligned} &P(\log\alpha_v - \log\alpha_{v'} - \tau\epsilon \geq g_{v'} - g_v) \\ &= \frac{1}{1 + \exp(\log\alpha_{v'} - \log\alpha_v + \tau\epsilon)} \\ &= \frac{1}{1 + \frac{\alpha_{v'}}{\alpha_v}e^{\tau\epsilon}}. \end{aligned} \tag{40}$$

Since $c, g_v, g_{v'}$ are generated independently, combining Eqn. 39 and 40, we can yield

$$P(c \geq \frac{g_{v'} - g_v}{\tau}) \geq \frac{P_\epsilon}{1 + \frac{\alpha_{v'}}{\alpha_v}e^{\tau\epsilon}}. \tag{41}$$

Similarly, from Eq. 38 we have $P(c \leq \mathrm{E}(c) + \epsilon) \geq P_\epsilon$ and subsequently,

$$\begin{aligned} P(\frac{g_{v'} - g_v}{\tau} \geq \mathrm{E}(c) + \epsilon) &= 1 - P(c + \epsilon \geq \frac{g_{v'} - g_v}{\tau}) \\ &= 1 - \frac{1}{1 + \frac{\alpha_{v'}}{\alpha_v}e^{-\tau\epsilon}}. \end{aligned} \tag{42}$$

Thus we have $P(c \leq \frac{g_{v'} - g_v}{\tau}) \geq P_\epsilon(1 - \frac{1}{1 + \frac{\alpha_{v'}}{\alpha_v}e^{-\tau\epsilon}})$ and also

$$P(c \geq \frac{g_{v'} - g_v}{\tau}) \leq 1 - P_\epsilon(1 - \frac{1}{1 + \frac{\alpha_{v'}}{\alpha_v}e^{-\tau\epsilon}}). \tag{43}$$

Combining Eqn. 41 and 43, we conclude that

$$\frac{P_\epsilon}{1 + \frac{\alpha_{v'}}{\alpha_v}e^{\tau\epsilon}} \leq P(c \geq \frac{g_{v'} - g_v}{\tau}) \leq 1 - P_\epsilon(1 - \frac{1}{1 + \frac{\alpha_{v'}}{\alpha_v}e^{-\tau\epsilon}}). \tag{44}$$

Based on this we consider the limitation for two sides and thus obtain

$$\lim_{P_\epsilon \to 1}\lim_{\tau \to 0} P(c \geq \frac{g_{v'} - g_v}{\tau}) = \frac{1}{1 + \frac{\alpha_{v'}}{\alpha_v}} = \frac{\alpha_v}{\alpha_v + \alpha_{v'}}. \tag{45}$$

Then with similar reasoning as Lemma 3, we have

$$\begin{aligned}&\lim_{P_\epsilon\to 1}\lim_{\tau\to 0} P(Y_v=1)\\ =&\lim_{P_\epsilon\to 1}\lim_{\tau\to 0} P(Y_v>Y_{v'},\forall v'\neq v)=\alpha_v/(\sum_{w=1}^{N}\alpha_w).\end{aligned} \tag{46}$$

Recall that

$$\begin{aligned}P_\epsilon &= (1-\frac{\sigma_v^2}{\epsilon_v^2})(1-\frac{\sigma_{v'}^2}{\epsilon_{v'}^2})\\ \sigma^2 &= \mathbb{V}_{\mathbf{w}}(\widehat{SM}_m^{+}(\mathbf{x},\mathbf{y}))\\ &= \frac{1}{m}\exp(-\frac{\|\mathbf{x}\|^2+\|\mathbf{y}\|^2}{2})\sum_{i=1}^{m}\exp(\mathbf{w}_i^\top(\mathbf{x}+\mathbf{y})),\end{aligned} \tag{47}$$

where $\mathbf{x}=\frac{\mathbf{q}_u}{\sqrt{\tau}}, \mathbf{y}=\frac{\mathbf{k}_{v,v'}}{\sqrt{\tau}}$. Therefore, P_ϵ is dependent of the precision $\epsilon_v, \epsilon_{v'}$, the random feature dimension m, and the temperature τ. If m is sufficiently large, σ would converge to zero and P_ϵ goes to 1. In such a case, Eqn. 46 holds once τ tends to zero. We thus conclude the proof.

A.3 Proof for Proposition 1

According to our definitions in Section 5, we have

$$\begin{aligned}&\mathcal{D}_{KL}(q_\phi(\tilde{\mathbf{A}}|\mathbf{X},\mathbf{A})\|p(\tilde{\mathbf{A}}|\mathbf{Y},\mathbf{X},\mathbf{A}))\\ =&\int_{A^*} q_\phi(\tilde{\mathbf{A}}|\mathbf{X},\mathbf{A})\log\frac{q_\phi(\tilde{\mathbf{A}}|\mathbf{X},\mathbf{A})}{p(\tilde{\mathbf{A}}|\mathbf{Y},\mathbf{X},\mathbf{A})}d\tilde{\mathbf{A}}\\ =&\int_{A^*} q_\phi(\tilde{\mathbf{A}}|\mathbf{X},\mathbf{A})\log\frac{q_\phi(\tilde{\mathbf{A}}|\mathbf{X},\mathbf{A})p_\theta(\mathbf{Y}|\mathbf{X},\mathbf{A})}{p(\tilde{\mathbf{A}},\mathbf{Y}|\mathbf{X},\mathbf{A})}d\tilde{\mathbf{A}}\\ =&\int_{A^*} q_\phi(\tilde{\mathbf{A}}|\mathbf{X},\mathbf{A})\log\frac{q_\phi(\tilde{\mathbf{A}}|\mathbf{X},\mathbf{A})p_\theta(\mathbf{Y}|\mathbf{X},\mathbf{A})}{p(\tilde{\mathbf{A}},\mathbf{Y}|\mathbf{X},\mathbf{A})}d\tilde{\mathbf{A}}\\ =&\int_{A^*} q_\phi(\tilde{\mathbf{A}}|\mathbf{X},\mathbf{A})\log\frac{q_\phi(\tilde{\mathbf{A}}|\mathbf{X},\mathbf{A})p_\theta(\mathbf{Y}|\mathbf{X},\mathbf{A})}{p(\mathbf{Y}|\tilde{\mathbf{A}},\mathbf{X},\mathbf{A})p(\tilde{\mathbf{A}}|\mathbf{X},\mathbf{A})}d\tilde{\mathbf{A}}\\ =&-\mathbb{E}_{q_\phi(\tilde{\mathbf{A}}|\mathbf{X},\mathbf{A})}[\log p(\mathbf{Y}|\tilde{\mathbf{A}},\mathbf{X},\mathbf{A})]+\log p_\theta(\mathbf{Y}|\mathbf{X},\mathbf{A})+\mathcal{D}_{KL}(q_\phi(\tilde{\mathbf{A}}|\mathbf{X},\mathbf{A})\|p(\tilde{\mathbf{A}}|\mathbf{X},\mathbf{A}))\\ =&-\text{ELBO}(\theta,\phi)+\log p_\theta(\mathbf{Y}|\mathbf{X},\mathbf{A})\end{aligned} \tag{48}$$

Since we assume q_ϕ can exploit arbitrary distributions over $\tilde{\mathbf{A}}$, we know that when the ELBO is optimized to the optimum, $\mathcal{D}_{KL}(q_\phi(\tilde{\mathbf{A}}|\mathbf{X},\mathbf{A})\|p(\tilde{\mathbf{A}}|\mathbf{Y},\mathbf{X},\mathbf{A}))=0$ holds. Otherwise, there exists $\phi^*\neq\phi$ such that $\text{ELBO}(\theta,\phi^*)>\text{ELBO}(\theta,\phi)$. Pushing further, when the optimum is achieved, $\log p_\theta(\mathbf{Y}|\mathbf{X},\mathbf{A})$ would equal to ELBO and namely is maximized.

B Implementation Details

We present implementation details in our experiments for reproducibility. We implement our model as well as the baselines with Python 3.8, Pytorch 1.8 and Pytorch Geometric 2.0.

For baseline models MLP, GCN, GAT, MixHop and JKnet, we use the implementation provided by [20][2]. For DropEdge and two structure learning baseline models (LDS and IDGL), we also refer to their implementation provided by the original papers [27, 11, 4]. Concretely, we use GCN as the backbone for them.

B.1 Details for Transductive Experiments

Architectures. For experiments on the datasets `Cora`, `Citeseer`, `Deezer` and `Actor`, the baseline models (GCN, GAT, MixHop, JKNet) are implemented with the following settings:

[2]https://github.com/CUAI/Non-Homophily-Benchmarks.

- Two GNN layers with hidden size 32 by default (unless otherwise mentioned). GAT uses 8 attention heads followed by its original setting.
- The activation function is ReLU (except GAT using ELU).

The architecture for our ALIGNS is:

- Two message-passing layers with hidden size 32, and we use 4 attention heads for each layer.
- The activation function is ELU, and we specify σ as sigmoid function to bound the relational bias.

For experiments on the two large datasets (OGB-Proteins and Amazon2M), we implement ALIGNS with 3 layers with hidden size 64 for better model capacity. We consider the same layer number and hidden size for competitors MLP and GCN.

Training Details. In each epoch, we feed the whole data into the model (except on `Amazon2M` for which we use mini-batch partition as we mentioned in Section 4.1), calculate the loss and conduct gradient descent accordingly. Concretely, we use BCE-Loss for two-class classification and NLL-Loss for multi-class classification, the Adam optimizer is used for gradient-based optimization. The training procedure will repeat the above process until a given budget of 500 epochs. Finally, we report the test accuracy achieved by the epoch that gives the highest accuracy on validation dataset.

Hyperparameters. For each model, we use grid search on validation set for hyper-parameter setting. The learning rate is searched within $\{0.01, 0.001, 0.0001, 0.00001\}$, weight decay within $\{0.05, 0.005, 0.0005, 0.00005\}$, and dropout probability within $\{0.0, 0.5\}$ The hyperparameters for the models are set as follows (we use the same notation as the original papers).

- For GCN and GAT, the learning rate is 0.01, and weight decay is set to 0.05. No dropout is used.
- For MixHop, the hyperparameters are the same as above, except that we further use grid search for hidden channels within {8, 16, 32, 64, 128}. We adopt 2 hops for all the four datasets.
- For JKNet, GCN is used as the backbone. Learning rate is set to 0.01 for `Deezer` and 0.001 for all the other three datasets. We concatenate the features in the final stage for Deezer, while we use max-pooling for the three other datasets. The hidden size is set as default, except for Deezer as 64.
- For DropEdge, the hidden size is chosen from {32, 64, 96, 128, 160}, the learning rate is within {0.01, 0.001, 0.0001}, and the dropedge rate is chosen from {0.3, 0.4, 0.5}.
- For LDS, the sampling time $S = 16$, the patience window size $\rho = 6$, the hidden size $\in$ {8, 16, 32, 64}, the inner learning rate $\gamma \in$ {1e-4, 1e-3, 1e-2, 1e-1}, and the number of updates used to compute the truncated hypergradient $\tau \in$ {5, 10, 15}.
- For IDGL, we use its original version without anchor approximation on `Cora`, `Citeseer` and `Actor`. For `Deezer`, even using anchor approximation, it would also suffer from out-of-memory. Besides, we set: $\epsilon = 0.01$, hidden size $\in$ {16, 64, 96, 128}, $\lambda \in$ {0.5, 0.6, 0.7, 0.8}, $\eta \in$ {0, 0.1, 0.2}, $\alpha \in$ {0, 0.1, 0.2}, $\beta \in$ {0, 0.1}, $\gamma \in$ {0.1, 0.2}, $m \in$ {6, 9, 12}.
- For our ALIGNS model, on `Cora` and `Citeseer`, the learning rate is 0.001 and weight decay is 0.005. The learning rate is 1e-5 for `Deezer` and 1e-4 for `Actor`, and the weight decay are set as 0.05. For `OGB-Proteins` and `Amazon2M`, the learning rate is 0.01 and no weight decaying is used. No dropout is used for all datasets. The temperature τ is set to 0.25. The weight λ is set to 1 for `Cora`, `Citeseer`, 0.01 for `Actor`, `Deezer`, `Amazon2m` and 0.1 for `OGB-Proteins`. The sampling number K is set as 5 for two large datasets and 10 for others. The dimension of feature map m is 30.

B.2 Details for Inductive Experiments

Architectures. For experiments on inductive settings, baseline models are implemented in the same manner as the transductive setting. The architecture for our ALIGNS is also the same as the transductive setting.

Training Details. The validation and testing graphs are unseen during training in the inductive setting, which means that the train/valid/test adjacent matrix are disjoint to each other. Concretely, in each epoch, we feed the training data matrix into the model, conduct gradient descent with BCE-Loss and Adam optimizer, and then statistic accuracy on the validation and test graphs. Typically, we modify the valid/test graph structures (drop edge, add noise, random initialization and k-NN) to evaluate our model's robustness against distribution shift in this setting. We finally report the test accuracy achieved by the epoch that gives the highest accuracy on the validation graph.

Hyperparameters. We also use validation set for hyper-parameter tuning. The optimal hyperparameters vary with different models and datasets in the inductive setting. We list them in detail as follows:

- For MLP, the learning rate is 1e-3 on `FB100` and 1e-5 on `Twitch`. Two datasets share the same weight decay of 0.1 and the same dropout of 0.5.
- For GCN, the learning rate is 1e-3 on `FB100` and 1e-4 on `Twitch`. Two datasets share the same weight decay of 0.1 and the same dropout of 0.5.
- For GAT, the learning rate is 1e-4 on `FB100` and 1e-4 on `Twitch`. Two datasets share the same weight decay of 0.1. We apply a dropout of 0.5 on `Twitch`, and remove it on `FB100`.
- For MixHop, the learning rate is 1e-4 on `FB100` and 1e-5 on `Twitch`. Two datasets share the same dropout of 0.5. The weight decay on FB100 is 0.05 , other than 0.005 on `Twitch`. We adopt 2 hops for both datasets.
- For JKNet, we use GAT as our backbone and use max-pooling in the final stage. The learning rate is 1e-3 on `FB100` and 1e-4 on `Twitch`. The weight decay on `FB100` is 5e-3, and 5e-2 on `Twitch`. We apply a dropout of 0.4 on `FB100`, and turn it off on `Twitch`.
- For our ALIGNS model, the learning rate is 1e-5 on `Twitch` and 1e-4 on `FB100`. Two datasets share the same weight decay of 0.1. Since `FB100` and `Twitch` are non-homophilous datasets, we abandon edge-level regularization in this setting. We apply dropout of 0.5 on `FB100`, and turn it off on `Twitch`. The temperature τ is set to 0.25 on both datasets. The dimension of feature map m is 60 on `Twitch`, and 80 on `FB100`. In both datasets, the number of samples K is set to 5.

B.3 Details for Graph-Enhanced Experiments

Architectures. The architectures of baselines (GCN, GAT, LDS and IDGL) and ALIGNS model are the same as the transductive setting, except that we use grid search to adaptively tune the hidden size. Besides, we also adopt BatchNorm for baseline models.

Training Details. The input data have no graph structures in this setting. As mentioned in Section 4.4, we use k-NN for artificially constructing a graph to enable message passing. The training procedure is the same as the transductive setting.

Hyperparameters. The hyperparameters for the models are listed as follows.

- For GCN, the learning rate is 0.01 and the weight decay is 5e-4 on both datasets. The size of hidden channel is set to 64. Dropout is not used during training.
- For GAT, the learning rate is 0.001 and the weight decay is 5e-3 on both datasets. The size of hidden channel is 32 on `Mini-ImageNet` and 64 on `20News`. No dropout is used during training. The number of attention heads is 8.
- For LDS, its hyperparameters are determined by grid search in the same manner as in the transductive setting.
- For IDGL, we use its anchor-based version that can scale to these two datasets. Besides, on `20News`, we set: hidden size=64, learning rate=0.01 $\lambda = 0.7$, $\eta = 0.1$, $\alpha = 0.1$, $\beta = 0$, $\gamma = 0.1$, $\epsilon = 0.01$, $m = 12$. On `Mini-ImageNet`, we set: hidden size=96, learning rate=0.01 $\lambda = 0.8$, $\eta = 0.2$, $\alpha = 0$, $\beta = 0$, $\gamma = 0.1$, $\epsilon = 0.01$, $m = 12$.
- For our ALIGNS model, the size of hidden channels is 64 on `20News`, and 128 on `Mini-ImageNet`. The number of attention heads is set to 6 on `Mini-ImageNet` and 4 on `20News`. For both datasets, the learning rate is 0.001, and the weight decay is 0.005. The weight λ is set to 1. The temperature τ is set to 0.25 for both datasets. The number of samples K is 10, and the dimension of feature map m is 30. No dropout is used during training.

C Dataset Information

We present detailed information for our used datasets concerning the data collection, preprocessing and statistic information. Table 5 provides an overview of the datasets we used in the experiment.

Table 5: Information for experiment datasets.

Dataset	Context	Property	# Task	# Nodes	# Edges	# Node Feat	# Class
Cora	Citation network	homophilous	1	2,708	5,429	1,433	7
Citeseer	Citation network	homophilous	1	3,327	4,732	3,703	6
Deezer	Social network	non-homophilous	1	28,281	92,752	31,241	2
Actor	Actors in movies	non-homophilous	1	7,600	29,926	931	5
OGB-Proteins	Protein interaction	multi-task classification	112	132,534	39,561,252	8	2
Amazon2M	Product co-occurrence	long-range dependence	1	2,449,029	61,859,140	100	47
Twitch-DE	User network	inductive learning	1	9,498	153,138	2,545	2
FB100-Baylor93	Social network	inductive learning	1	12,803	1,359,634	5	2
Mini-ImageNet	Image classification	no graph/k-NN graph	1	18,000	0	128	30
20News-Groups	Text classification	no graph/k-NN graph	1	9,607	0	236	10

C.1 Dataset Information

Transductive learning datasets. For experiments on transductive node classification, we evaluate our model on two homophilous datasets `Cora` and `Citeseer` [33], and other two non-homophilous datasets `Actor` [25] and `Deezer` [29]. The first two are citation network datasets that contain sparse bag-of-words feature vectors for each document and a list of citation links between documents. The citation links are treated as (undirected) edges and each document has a class label. `Deezer` is a social network of users on Deezer from European countries, where edges represent mutual follower relationships. The node features are based on artists liked by each user and nodes are labeled with reported gender. `Actor` is a graph representing actor co-occurrence in Wikipedia pages. Each node corresponds to an actor, and the edge between two nodes denotes co-occurrence on the same Wikipedia page. Node features correspond to some keywords in the Wikipedia pages and the nodes are classified into five categories w.r.t. words of actor's Wikipedia. These four datasets are relatively small with thousands of instances and edges (`Deezer` is the largest one with nearly 20K nodes).

To evaluate ALIGNS's scalability, we further consider two large datasets: `OGB-Proteins` [15] and `Amazon2M` [5]. These two datasets have million-level nodes and edges and require the model for scalable training. The `OGB-Proteins` dataset is an undirected, and typed (according to species) graph in which nodes represent proteins and edges indicate different types of biologically meaningful associations between proteins. All edges come with 8-dimensional features, where each dimension represents the approximate confidence of a single association type and takes on values between 0 and 1. The proteins come from 8 species and our task is to predict the presence of 112 protein functions in a multi-label binary classification setup respectively. `Amazon2M` is extracted from Amazon Co-Purchasing network [23], where each node represents a product, and the graph link represents whether two products are purchased together, the node features are generated by extracting bag-of-word features from the product descriptions. The top-level categories are used as labels for the products.

Inductive learning datasets. For experimens on inductive node classification, we evaluate our model on two non-homophilous datasets: `Twitch-DE` [28] and `Facebook100-Baylor93` [35]. `Twitch` contains seven networks where users are nodes and mutual friendships between them are edges. Node features are games liked, location and streaming habits. The class labels denote whether a streamer uses explicit language. `Facebook100` consists of 100 Facebook friendship network snapshots from 2005, each of which has as nodes the Facebook users from a given American university. Each node is labeled with the reported gender of the user. The node features are major, year, high school, etc. Specifically, we use the subgraph `DE` from `Twitch` and the subgraph `Baylor93` from `Facebook100` for experiments. Both of them have roughly 10k nodes.

Graph-enhanced application datasets. We evaluate our model on two datasets without graph structure: `20News-Groups` [24] and `Mini-ImageNet` [37]. The `20News` dataset is a collection of approximately 20,000 newsgroup documents (nodes), partitioned (nearly) evenly across 20 different newsgroups. We take 10 classes from 20 newsgroups and use words (TFIDF) with a frequency of more than 5% as features. The `Mini-ImageNet` dataset consists of 84×84 RGB images from 100 different classes with 600 samples per class. For our experiment use, we choose 30 classes from the dataset, each with 600 images (nodes) that have 128 features extracted by CNN.

C.2 Dataset Preprocessing

All the datasets we used in the experiment are directly collected from the source, except `Mini-ImageNet`, whose features are extracted by ourselves. Following the setting of [31], we com-

pute node embeddings via a CNN model with 4 convolutional layers followed by a fully-connected layer resulting in a 128 dimensional embedding. Finally, the 128 dimensional outputs are treated as the features of the nodes (images) for subsequent GNN-based downstream task.

D More Experiment Results

D.1 Extra Ablation Studies

Some ablation study results have been reported in Section 4.5 and here we supplement extra results on more datasets. In Table 6, we summarize the experiment results on the four transductive datasets for ALIGNS w/ and w/o relational bias and edge-level regularization. The results show that both components contribute to positive effects, especially the edge-level regularization, without which we can see a significant drop for the test accuracy on `Cora` and `Citeseer`. The reason could be that these two datasets are homophilous graphs where observed edges are reliable for providing beneficial information and in contrast, the other datasets are non-homophilous graphs where the training signal from observed edges might not help a lot and could even be harmful. Similar results and interpretations are applicable for relational bias.

Table 6: Ablation study results on transductive datasets, where "rb" denotes relational bias and "reg" represents edge-level regularization.

Dataset	ALIGNS	ALIGNS w/o reg	ALIGNS w/o rb
`Cora`	**88.69** $\pm$ 0.46	81.98 $\pm$ 0.46	88.06 $\pm$ 0.59
`Citeseer`	**76.33** $\pm$ 0.59	70.60 $\pm$ 1.20	74.12 $\pm$ 0.64
`Deezer`	**71.24** $\pm$ 0.32	71.22 $\pm$ 0.32	71.10 $\pm$ 0.36
`Actor`	**35.31** $\pm$ 1.29	35.15 $\pm$ 1.32	34.60 $\pm$ 1.32

D.2 Impact of Temperature and Random Feature Dimension

Fig. 4 shows the impact of the temperature τ and the dimension of feature map m on `Cora`. The figure illustrates that the test accuracy increases and then falls with the temperature changing from low to high values (usually achieves the peak accuracy with a temperature of 0.4). Besides, we can see that when the temperature is relatively small, the test accuracy goes high with the dimension of random features increasing. However, when the temperature is large, the accuracy would drop even with large feature dimension m. Such a phenomenon accords with the theoretical result presented in Section 3.2. For low temperature which enables desirable approximation performance for Gumbel-Softmax, then larger random feature dimension would help to produce better approximation to the original exponentiate-then-dot operator. In contrast, high temperature could not guarantee precise approximation for the original categorical distribution, which deteriorates the performance.

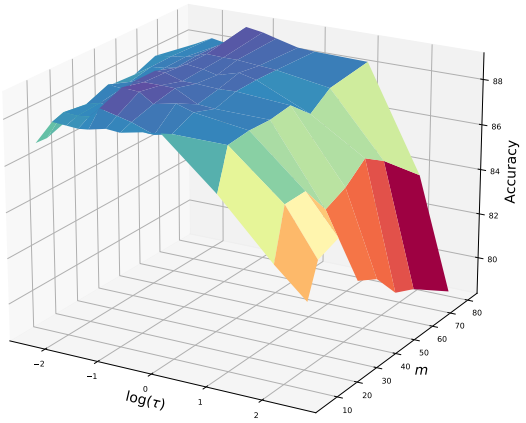
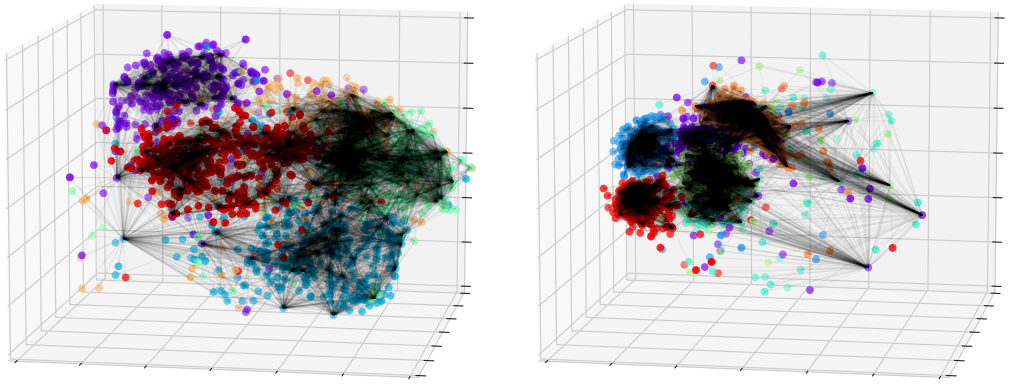


Figure 4: Impact of the temperature τ and the dimension of random feature map m on `Cora`.



(a) 20News-Groups (b) Mini-ImageNet

Figure 5: Visualization of node embeddings and edge connections produced by ALIGNS on graph-enhanced application datasets. We choose partial classes for visualization and mark the nodes with a particular class with one color.

D.3 Visualization and Implications

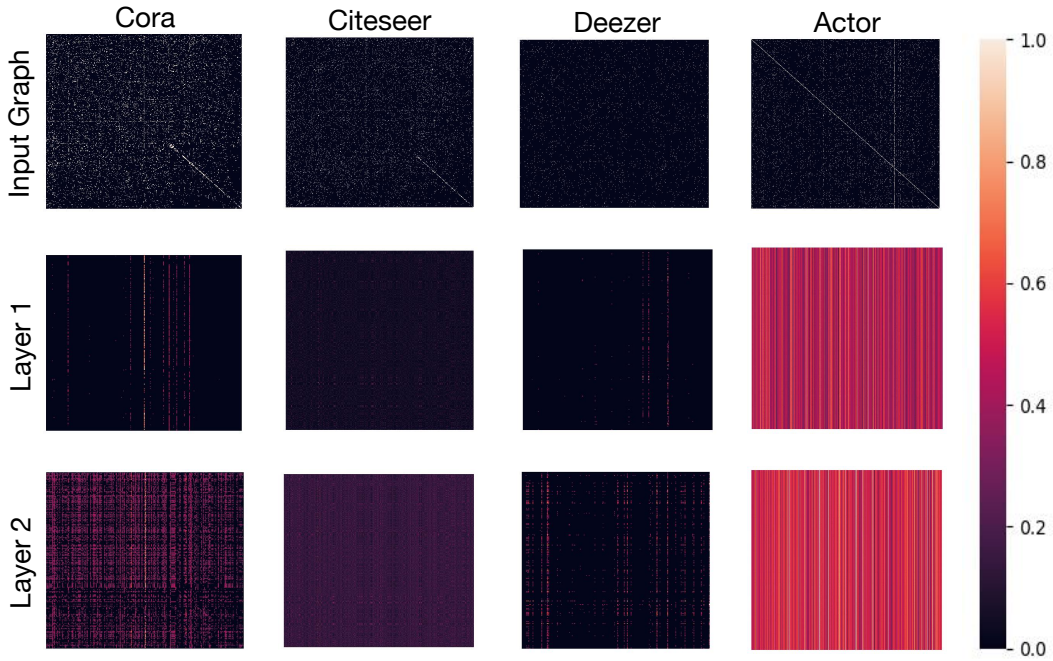


Figure 6: Visualization for input graph structures and latent graph structures (given by two layers of ALIGNS) with colors reflecting the weights.

We provide the visualization results of node embeddings and edge connections (filter out the edges with weights larger than a threshold) on 20News-Groups and Mini-Imagenet in Fig. 5. The figures show that ALIGNS tends to assign more weights for nodes with the same class and sparse edges for nodes with different classes. This helps to interpret why ALIGNS improves the performance on downstream node-level prediction: the latent structures can propagate useful information to help the model learn better node representations that can be easily distinguished by the classifier. Besides, we plot the attention maps of two model layers in Fig. 6 and also the input graph structures on Cora, Citeseer, Deezer and Actor. We can see that the latent structures learned by ALIGNS have different patterns than the observed ones, especially for heterophilic graphs. Another interesting

phenomenon is that there exist some dominant nodes which are assigned large weights by other nodes, forming some vertical 'lines' in the heatmap. This suggests that these nodes could contain critical information for the learning tasks and play as pivots that could improve the connectivity of the whole system.

E Outlooks and Potential Impacts

The proposed model in this paper mainly focuses on node-level tasks for graph-structured data, though it could be extended to other graph-related tasks including link prediction, node regression, sub-graph recognition by using the ALIGNS network as a backbone encoder for node representation learning. For practitioners who would like to apply our model into more industry applications, like recommendation, advertisement, autonomous driving, etc. that involve graph data processing, one can also treat our kernelized Gumbel-Softmax operator as a plug-in component for enabling efficient and scalable learning over data points' inter-dependence and latent interactions, which may help to enhance the representational power and capacity for existing encoder architectures. To stay focused in this paper, we leave these interesting and promising directions as future investigations. And, to our knowledge, we have not identified any negative social impact of this work.