Supplementary Material: GLDL: Graph Label Distribution Learning

Roadmap

The supplementary material has been meticulously curated to complement and further elucidate the contents of the main manuscript. A structured guide to navigate the content is as follows. Appendix A presents the pseudo-code of our proposed `GLDL` algorithm and the detailed implementation of its graph neural network architecture. Appendix B furnishes extended statistics of the datasets employed for benchmarking our experiments; a concise version of this information is touched upon in Section 6 of the main manuscript. In adherence to the stringent standards set by the AAAI reproducibility checklist, Appendix C documentes the specific ranges used for hyperparameter search. This exhaustive tuning facilitated optimal performance across all the evaluated methods, including our `GLDL` algorithm and its rival models. Appendix D delves into a broader spectrum of results obtained using the `GLDL` approach, which had to be excluded from the main manuscript owing to page limits. Appendix E elaborates an exhaustive proof for Theorem 1 as introduced in Section 5 of the main manuscript.

Appendix A: Pseudo-code and Implementation Details

Algorithm 1 shows a general training pipeline for the framework. Several modules are introduced in detail below.

Label-Label Graph Module The GCN^c architecture for label graph G^c consists of GCN layers, projection layers, and a softmax layer. The 1st GCN layer integrates node information from both label-label graph G^c and node-node graph G following Eq. (1) and line 13 in Algorithm 1. The rest of the GCN layers integrate node information from nodes within graph G^c alone. The learned hidden label embedding $H^c = [h^c_1; \cdots ; h^c_q]$ is then forward projected to the original feature space to ensure that the label representation can be aligned with the node features. This alignment process is crucial to allow node propagation between graphs in the 1st GCN layer. The learned label representation is also used to regularize the dynamic graph generator $GG(H^c)$ as part of the momentum loss for A^c_{new} in Fig 2 and line 27 in Algorithm 1. A softmax layer is applied to obtain probability values needed for computing the cross-entropy loss in Eq. (6) and line 16 in Algorithm 1. The learned embedding is updated as the new label features for a certain frequency denoted by freq$_c$ following line 24 in Algorithm 1.

Node-Node Graph Module The GCN architecture for node graph G is the same as the GCN^c except for the num-

Algorithm 1: The GLDL algorithm

```
   Input: G:(A,X,Y),freq$_v$,freq$_c$, epochs, epochs$_{GG}$
   Model: GCN,GCN$^c$,GG($\cdot$)
   Init: $A^h \in \mathbb{R}^{(m+q,m+q)} \leftarrow 0$,$E^h \in \mathbb{R}^{(m,q)} \leftarrow 0$
   Output: $\hat{Y}$
1  $X^c \leftarrow Agg(X_{train})$
2  $A^c \leftarrow GG(X^c)$
3  for $i \leq m$ do
4      for $j \leq q$ do
5          if argmax$(Y_i) = j$ then
6              $E^h_{i,j} \leftarrow 1$
7  $A^h \leftarrow \begin{bmatrix} A & E^h \\ (E^h)^T & A^c \end{bmatrix}$
   # $A^h$ is heterogeneous graph connecting $G$ and $G^c$
   # $E^h$ are the induced edges connecting two graphs
8  $X^h \leftarrow [X \; ; \; X^c]$
9  $X^{hc} \leftarrow [X \; ; \; X^c]$
10 for $i \leq epochs$ do
11     $H^1 \leftarrow$ GCN$^{(1)}(A^h, X^h)$
       # GCN$^{(1)}$ means 1st GCN layer
12     $\hat{Y} \leftarrow$ GCN$(A, H^1)$
13     $H^{c(1)} \leftarrow$ GCN$^{c(1)}(A^h, X^{hc})$.embedding()
       # GCN$^{c(1)}$ means 1st GCN$^c$ layer
14     $\hat{Y}^c \leftarrow$ GCN$^c(A^c, H^{c1})$
15     $\ell_{KL}(Y, \hat{Y}) \leftarrow$ compute loss using Eq. (5)
16     $\ell_{CE}(Y^c, \hat{Y}^c) \leftarrow$ compute loss using Eq. (6)
17     compute gradient of $\ell_{KL}$and $\ell_{CE}$
18     update GCN and GCN$^c$ with gradient
19 if $i$ % $freq_v == 0$ then
20     $H \leftarrow$ GCN.embedding()
21     $X^h \leftarrow [H \; ; \; X^c]$
       # Concatenate H and $X^c$
22 if $i$ % $freq_c == 0$ then
23     $H^c \leftarrow$ GCN$^c$.embedding()
24     $X^{ch} \leftarrow [X \; ; \; H^c]$
       # Concatenate X and $H^c$
25     if dynamic mode then
26         for $j = 1, 2, \ldots, epochs_{GG}$ do
27             Train $GG(\cdot)$ with Eqs. , (7) , (8)
28         $A^c \leftarrow GG$.inference$(H^c)$
29         update $A^h$ with $A^c$
   # embedding() extract embedding after projection
30 $\hat{Y}^c \leftarrow GCN^c(A^h, A^c, X^h, X^c)$
   # output predicted distribution for inference modes
31 return $\hat{Y}^c$
```

ber of nodes, independent learnable weight, and final objective function. After applying the softmax function, KL-divergence loss as stated in Eq. (5) is used as the loss function. The learned embedding is updated using a different frequency denoted by freq_v following line 21 in Algorithm 1. The updated embedding $H = [h_1; \cdots ; h_n]$ is also used to regularize the dynamic graph generator $GG(Agg(H))$ as part of the consistency loss for A^c_{syn} following line 27.

Dynamic Graph Module This module is an optional module to replace the static graph generation function $GG(\cdot)$ as stated in Eq. (3). Following Eq. (7), (8) or from lines 25 to 29 in Algorithm 1, the parametric model is trained every certain epoch (same frequency freq_c as the label features are updated for simplicity of the model) and then infer a new graph topology A^c_{new} to replace the original A^c_{old} for succeeding training.

Appendix B: Additional Dataset Statistics and Benchmark Creation Method

Dataset Creation In order to create datasets with distribution labels for each node, heterogeneity of datasets was utilized. Borrowing the metapath idea, conversion of heterogeneous datasets can be achieved by homogenizing along a certain metapath (Fu et al. 2020), composed of different node types. During the homogenization process, auxiliary node types or preexisting labels along the metapath can be aggregated and normalized to naturally create distribution labels for each node. Metapaths are chosen to generate meaningfully aggregated homogeneous graphs and distribution labels that could have potential applications. For example, co-authorship and subject distribution labels can be used to more efficiently organize citation networks.

In the following, we briefly outline the node label distribution generation process for the four benchmark networks.

- **DBLP**: Originally a citation network from (Tang et al. 2008) composed of Author, Paper, Conference, and Term nodes, DBLP is homogenized through an Author-Paper-Author metapath, resulting in a coauthorship graph. The nodes of the graph represent authors and the edges indicate that two authors share at least one paper. Each of these authors is then labeled with a distribution composed of the conference area in which papers are published, where the distribution is defined as the percentage of papers published at the respective conference area with respect to the total number of papers published by the author. The node features are a bag of words representation of all authors' papers.
- **Yelp**: Originally a review network composed of Business and User nodes connected by Review, Check-in, and Tip edges, Yelp was aggregated along the Business-User-Business metapath, using review edges. The nodes of the graph represent businesses, and the edges indicate a common customer base between businesses. The businesses were then labeled with a distribution built from the star ratings extracted from their review edges. Business features are a bag-of-word representation extracted from their reviews.

Table 1: DBLP dataset average label distributions (mean±Std) for all nodes *w.r.t.* different classes. To generate this average, label distributions are grouped by their dominant class. Average label distributions are then calculated within their respective groups. Standard deviations are calculated between each class within each individual group. The table is $q \times q$, where q denotes the number of classes. The diagonal values denote the dominant class's probability value. The lower the diagonal values, the more spread out the class probability is.

Class 1	**Class 2**	**Class 3**	**Class 4**
0.8774 ± 0.1755	0.0415 ± 0.1086	0.0131 ± 0.0619	0.0680 ± 0.1226
0.0628 ± 0.1188	0.7733 ± 0.2157	0.0767 ± 0.1463	0.0873 ± 0.1536
0.0161 ± 0.0576	0.0438 ± 0.1023	0.8667 ± 0.1881	0.0734 ± 0.1472
0.0309 ± 0.0879	0.0364 ± 0.0907	0.0266 ± 0.0757	0.9061 ± 0.1652

Table 2: Yelp dataset average label distributions (mean±Std) for all nodes *w.r.t.* different classes. To generate this average, label distributions are grouped by their dominant class. Average label distributions are then calculated within their respective groups. Standard deviations are calculated between each class within each individual group. The table is $q \times q$, where q denotes the number of classes. The diagonal values denote the dominant class's probability value. The lower the diagonal values, the more spread out the class probability is.

Class 1	**Class 2**	**Class 3**	**Class 4**	**Class 5**
0.5116 ± 0.1742	0.1003 ± 0.0865	0.0861 ± 0.0887	0.1137 ± 0.0937	0.1883 ± 0.1186
0.1310 ± 0.1086	0.3639 ± 0.0828	0.1484 ± 0.1041	0.1957 ± 0.1112	0.1611 ± 0.1268
0.1154 ± 0.0854	0.1203 ± 0.0976	0.3967 ± 0.1067	0.2239 ± 0.1006	0.1437 ± 0.1106
0.1180 ± 0.0882	0.0883 ± 0.0747	0.1361 ± 0.0865	0.4218 ± 0.1142	0.2358 ± 0.1128
0.1079 ± 0.1011	0.0507 ± 0.0596	0.0580 ± 0.0648	0.1358 ± 0.1119	0.6476 ± 0.1950

- **Yelp2**: Another graph generated from the same heterogeneous Yelp graph with a larger bag-of-word feature space. We use Yelp2 to validate how node features impact the algorithm performance for networks with similar topologies.
- **ACM**: Originally a citation network from (Tang et al. 2008) composed of Author, Paper, and Subject nodes, ACM was homogenized similarly to DBLP but was labeled using subject nodes instead of the conference nodes. The resultant distribution represents the disciplines in which each author has published. The node features are also a bag-of-word representation of all authors' papers.

Figure 5 reports four datasets' data statistics, including the number of nodes, edges, homophily property, etc.

Label Distribution Dataset Statistics

Figure 1 and 2 show average label distributions (mean±std) for each class under the Yelp and DBLP dataset, respectively. For each dataset, the nodes whose dominant label is the indexed class are used to calculate the average label distributions for each label. This provides a way for us to observe the spread of each node's label distribution (*e.g.* whether labels are spread out or are dominant by a single class).

Table 1 and 4 report the average distribution for each class of the DBLP and ACM datasets. Their diagonal terms are both very high indicating that the overall distribution is dominant by a single class while Table 2 and 3 reporting Yelp and Yelp2 datasets show a more smooth distribution curve with less peak diagonal terms.

Table 3: Yelp2 dataset average label distributions (mean$\pm$Std) for all nodes *w.r.t.* different classes. To generate this average, label distributions are grouped by their dominant class. Average label distributions are then calculated within their respective groups. Standard deviations are calculated between each class within each individual group. The table is $q \times q$, where q denotes the number of classes. The diagonal values denote the dominant class's probability value. The lower the diagonal values, the more spread out the class probability is.

Class 1	**Class 2**	**Class 3**	**Class 4**	**Class 5**
0.5356 ± 0.1811	0.0928 ± 0.0847	0.0758 ± 0.0840	0.1050 ± 0.0977	0.1908 ± 0.1268
0.1258 ± 0.0835	0.3840 ± 0.1254	0.1279 ± 0.1039	0.1722 ± 0.0999	0.1901 ± 0.1130
0.1278 ± 0.0863	0.1108 ± 0.0857	0.3822 ± 0.0982	0.2061 ± 0.1050	0.1731 ± 0.1177
0.1194 ± 0.0876	0.0853 ± 0.0760	0.1304 ± 0.0840	0.4283 ± 0.1185	0.2365 ± 0.1143
0.1114 ± 0.1048	0.0455 ± 0.0602	0.0522 ± 0.0659	0.1249 ± 0.1111	0.6660 ± 0.1991

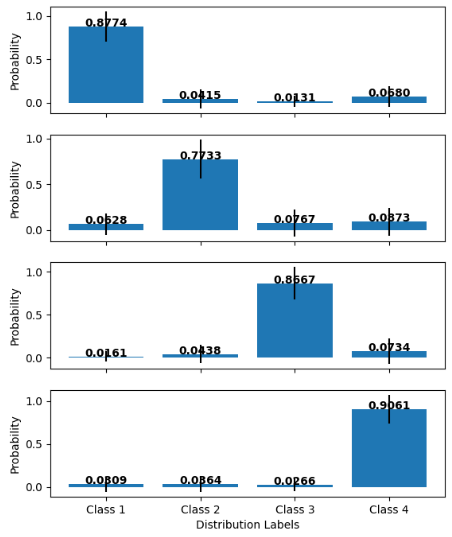

Figure 1: DBLP average node lable distribution *w.r.t* different classes. For each dataset, the nodes whose dominant label is the x-axis indexed class are used to calculate the average label distributions for each label.

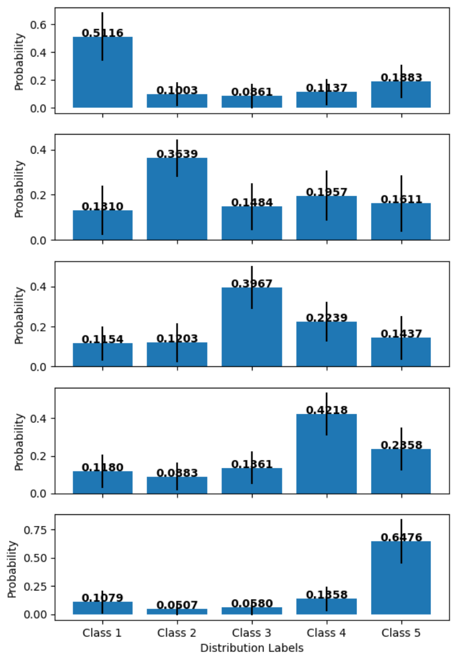

Figure 2: Yelp average node lable distribution *w.r.t* different classes. For each dataset, the nodes whose dominant label is the x-axis indexed class are used to calculate the average label distributions for each label.

Appendix C: Additional Experiment Settings

In our experiments, each dataset is split using 50-20-30 training-validation-test percentages. Grid search is used for hyperparameter search. Our validation loss is KL-divergence alone. The early-stopping mechanism, with patience set to 20, is used as a convergence criterion for the training stage of all the models. The range for each hyperparameter searched is reported with the format of hyperparameter name followed by $[\cdots]$ as shown below.

For the GCN baseline, hyperparameters include the hidden dimension $[32, 64, 128, 256, 512]$, and the number of layers $[2, 3, 4]$ for the GCN, along with the learning rate $[0.005, 0.0005]$ and weight decay $[0, 0.0005]$ for the Adam optimizer.

For our static model GLDL$_s$, additional hyperparameters, in addition to those of GCN baselines, include number of layers for the GCN^c $[2, 3, 4]$, the number of epochs for updating node features(freq$_v$) over range $[10, 20, 30, 40, 50, 60, 70]$, and the number of epochs for updating label features(freq$_c$) $[10, 20, 30, 40, 50, 60, 70]$.

For the dynamic setting GLDL$_d$, extra hyperparameters, in addition to those of GLDL$_s$, include the hidden dimension $[32, 64, 128, 256, 512]$ and layer size $[1, 2]$ for $GG(\cdot)$ along with the fixed maximum epochs training $GG(\cdot)$, denoted as epoch$_{GG}$ $[30, 50, 70]$. Empirically, 50 or 70 is set for epoch$_{GG}$ without significant change for the final performance.

Due to the huge hyperparameter search space for the dynamic mode, we stopped searching whenever a configuration is better than the current existing baselines. In general, the actual performance of the dynamic settings with the full search of the hyperparameter space should always be the same or better than our reported performance. Note that a complete grid search is not always necessary for a suboptimal solution.

It is observed during the sensitivity analysis that the combination of the number of GCN layers, GCN^c layers, and MLP layers of $GG(\cdot)$ are the most sensitive hyperparameters of the GLDL$_d$ model. There is no strong correlation evidence between the sensitive hyperparameters and hyperparameters with large search space (freq$_v$ and freq$_c$), *i.e.*, a better combination of layers behave consistently better even changing other hyperparameters.

Appendix D reports more parameter sensitivity study results. One efficient searching strategy is thus to fix every other hyperparameter and search for the most sensitive hyperparameter first, followed by the second sensitive and

Table 4: ACM dataset average label distributions (mean±Std) for all nodes *w.r.t.* different classes. To generate this average, label distributions are grouped by their dominant class. Average label distributions are then calculated within their respective groups. Standard deviations are calculated between each class within each individual group. The table is $q \times q$, where q denotes the number of classes. The diagonal values denote the dominant class's probability value. The lower the diagonal values, the more spread out the class probability is.

Class 1	**Class 2**	**Class 3**	**Class 4**	**Class 5**	**Class 6**	**Class 7**	**Class 8**	**Class 9**	**Class 10**	**Class 11**
0.9615 ± 0.1387	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0385 ± 0.1387	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000
0.0000 ± 0.0000	0.8533 ± 0.2477	0.0384 ± 0.1266	0.0147 ± 0.0776	0.0090 ± 0.0495	0.0441 ± 0.1262	0.0203 ± 0.0872	0.0098 ± 0.0700	0.0105 ± 0.0540	0.0000 ± 0.0000	0.0000 ± 0.0000
0.0000 ± 0.0000	0.0014 ± 0.0194	0.9070 ± 0.1947	0.0300 ± 0.1083	0.0023 ± 0.0262	0.0134 ± 0.0675	0.0096 ± 0.0542	0.0289 ± 0.1055	0.0051 ± 0.0437	0.0000 ± 0.0000	0.0024 ± 0.0311
0.0000 ± 0.0000	0.0004 ± 0.0081	0.0053 ± 0.0353	0.9139 ± 0.1866	0.0023 ± 0.0274	0.0081 ± 0.0566	0.0037 ± 0.0392	0.0497 ± 0.1428	0.0101 ± 0.0644	0.0006 ± 0.0121	0.0060 ± 0.0514
0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0025 ± 0.0249	0.8134 ± 0.2451	0.0616 ± 0.1537	0.0388 ± 0.1170	0.0670 ± 0.1649	0.0168 ± 0.0771	0.0000 ± 0.0000	0.0000 ± 0.0000
0.0000 ± 0.0000	0.0025 ± 0.0209	0.0092 ± 0.0464	0.0061 ± 0.0359	0.0090 ± 0.0429	0.8457 ± 0.2242	0.0707 ± 0.1465	0.0258 ± 0.0946	0.0255 ± 0.0935	0.0035 ± 0.0349	0.0021 ± 0.0228
0.0000 ± 0.0000	0.0007 ± 0.0132	0.0047 ± 0.0311	0.0017 ± 0.0209	0.0016 ± 0.0161	0.0264 ± 0.0854	0.9115 ± 0.1847	0.0234 ± 0.1002	0.0276 ± 0.1107	0.0016 ± 0.0165	0.0007 ± 0.0132
0.0000 ± 0.0006	0.0000 ± 0.0018	0.0035 ± 0.0301	0.0022 ± 0.0220	0.0014 ± 0.0178	0.0026 ± 0.0240	0.0025 ± 0.0233	0.9598 ± 0.1177	0.0248 ± 0.0953	0.0016 ± 0.0240	0.0013 ± 0.0210
0.0000 ± 0.0000	0.0000 ± 0.0000	0.0003 ± 0.0072	0.0000 ± 0.0000	0.0005 ± 0.0109	0.0044 ± 0.0327	0.0071 ± 0.0413	0.0181 ± 0.0725	0.9678 ± 0.1018	0.0009 ± 0.0217	0.0009 ± 0.0217
0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0045 ± 0.0302	0.0045 ± 0.0302	0.9909 ± 0.0603	0.0000 ± 0.0000
0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0105 ± 0.0540	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	0.9895 ± 0.0540

those who have weak correlation but with large searching space, *i.e.*, freq$_v$ and freq$_c$ in our cases, etc. Such a strategy is linear in terms of the number of hyperparameters with large searching space and is thus more efficient than grid search. We used such a strategy for the ACM data hyperparameter search which result in efficient computation time. Such a hyperparameter searching strategy is useful to balance model performance and training efficiency.

Appendix D: Additional Results and Analysis

Over-smooth Phenomenon Analysis

When testing the Yelp dataset, it is observed that increasing the number of GCN layers improves the performance of the six distribution measures even when the number of layers is far from common settings. The output of the predicted distribution shows that all nodes' distribution becomes similar. Define a node's single label as its index of the class with the maximum probability distribution, the most dominant class is the label that has most nodes.

The experiment shows that when the number of GCN layers becomes very large, all nodes are predicted to have the same dominant class label. As shown in Fig 3, the weighted f1 score decreases first and then becomes flat showing that all predictions are dominant class. Meanwhile, the intersection score becomes higher, which suggests a better performance. However, the model actually performs worse because it predicts every node to be the same label distribution.

The same experiment setting is carried out on the ACM dataset as shown in Fig 4. While it looks different in the first 30 GCN layers, we can observe the same pattern for the GCN layers from 30 to 70 where the intersection increases with the accuracy and the weighted-f1 score stays flat.

The difference between the two datasets is the rate that the weighted f1 score becomes flat, *i.e.*, and the rate that the GCN model becomes over-smoothed. While estimating when the GCN model becomes over-smoothed is hard in general, it is, therefore, important to include the accuracy and weighted-f1 score as part of the evaluation metric. Another reason that GCN deteriorates more quickly with layers increasing on Yelp than ACM might be the low homophily score, as shown in Table 5, where GCN is considered as being biased towards a high homophilic graph.

Table 5: Dataset statistics. Homophily is computed as the fraction of edges connecting nodes sharing the same labels (Ma et al. 2022), where labels used are the dominant class with the maximum probability for each node. Lower homophily indicates graph is heterophilous.

Dataset	**Nodes**	**Edges**	**Average Degree**	**Homophily**	**# of Labels**	**# of Features**	**Metapath**
DBLP	1711	5796	3.387492694	0.78	4	334	Author-Paper-Author
Yelp	2719	38233	14.06141964	0.5019	5	1640	Business-User-Business
Yelp2	3000	33857	11.28566667	0.46	5	6167	Business-User-Business
ACM	6007	25338	4.218078908	0.92	11	1903	Author-Paper-Author

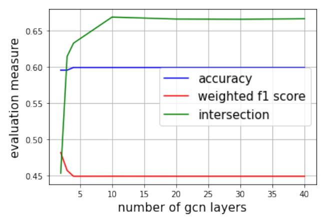

Figure 3: Over-smooth test on the Yelp dataset

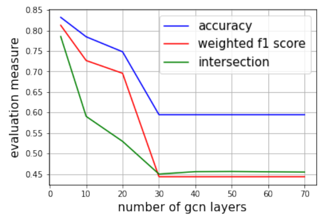

Figure 4: Over-smooth test on the ACM dataset

Ablation Study of Loss Terms on the Algorithm Performance To verify the effectiveness of our unsupervised loss, variants of our dynamic version are trained separately under the ACM dataset: no training loss; training with consistent loss only ($\alpha = 0$); training with momentum loss only (set α to a large number). The ACM dataset is selected because it has more than 10 classes and is sufficient to validate the effectiveness of the graph generation process.

The results in Table 6 show that consistency loss provides a good improvement in overall performance and using both losses provides a better Cosine distance and KL divergence. No training loss performs the worst with only 1 best performance metric among all eight metrics. This shows empirically that our proposed unsupervised training loss does provide benefits for the overall model performance.

Sensitive Analysis of Hyperparameter Selection To better understand how hyperparameters affect the GLDL$_d$ model, an ablation study is conducted on different hyperparameters for the GLDL$_d$ model using the ACM dataset. It is observed that the combination of the number of layers for GCN_c, GCN, $GG(\cdot)$, and the hidden dimension of the

Table 6: Variants of Dynamic mode for testing effectiveness of the proposed unsupervised loss. The number of best performance counts the number of best (Bold) or second-best (Italian) results for each model. CHD: Chebyshev Distance, COD: Cosine Distance, CAD: Canberra Distance, CLD: Clark Distance, IND: Intersection Distance. A ↑ / ↓ symbol denotes that the measured values are higher/lower the better.

	ACM								
Dynamic variants	CHD↓	COD↓	CAD↓	CLD↓	IND↑	KL Divergence↓	Accuracy↑	weighted F1 score↑	# of best performance metric↑
No training loss	*0.3406*	0.2199	9.435	2.9524	0.6557	0.9724	0.7072	0.6837	1/8
Consistency alone	**0.3386**	*0.2179*	**9.417**	**2.9478**	**0.658**	*0.9648*	**0.7171**	**0.6953**	8/8
Momentum alone	0.3476	0.2321	*9.4196*	*2.9491*	0.6487	1.0297	0.7022	0.6693	2/8
Consistency+momentum	0.3427	**0.2178**	9.4921	2.9693	*0.6536*	**0.9472**	*0.7121*	*0.6918*	5/8

GCN structure are the most sensitive hyperparameters to the model. Fixing the other hyperparameter values, a grid search is conducted on the combination of the three hyperparameters with a range of $[2,3,4]$,$[1,2]$, and $[2,3,4]$ corresponding to the number of layers for GCN, GCN_c, and $GG(\cdot)$.

The results Table 7 verify whether there exists a significant performance change for the pattern of the layer combination. When changing the hyperparameters with large searching space (freq$_v$ and freq$_c$), a different combination of freq$_v$ and freq$_c$ is shown in Table 7. The same optimal combinations of the layers suggest that there is no strong correlation between the two.

In Table 8, one optimal choice of layer combination is fixed, and the hidden dimension size of the GCN and MLP for GG is changed. A pattern shows that increasing the GCN hidden dimension while keeping a moderate size of MLP improves the performance of the model, and such a trend is consistent between the validation set and test set. A large GCN hidden dimension improves the model performance indicating that our model has better expressive power than GCN alone. The hyperparameter search over the GCN baseline suggests that increasing the hidden dimension up to a similar value of GLDL provides worse results, *i.e.*, GCN gets overfitting with a large hidden dimension.

Table 7: Sensitivity of # of layers and changes of the combination of $freq_c/freq_v$. * means the value is far worse than the recorded results. All other not mentioned hyperparameters are fixed. Special note on hidden GCN dimension is 128 and hidden MLP dimension is 64.

$freq_c/freq_v$	50/30	70/50
Layers$_{comb}$	Val/Test	Val/Test
2/1/2	0.9336/1.0478	0.9473/1.0355
2/1/3	0.9347/1.0669	0.944/1.0303
2/1/4	0.9386/1.0853	0.9488/1.0141
2/2/2	**0.9239/0.989**	**0.9268/0.9844**
2/2/3	**0.9283/0.9927**	**0.9224/0.9875**
2/2/4	**0.9246/0.9897**	**0.9325/0.9967**
3/1/2	*	*
3/1/3	*	*
3/1/4	*	*
3/2/2	0.9476/1.0037	0.9545/1.0096
3/2/3	0.9495/1.006	0.9488/1.0016
3/2/4	0.9512/1.0069	*
4/1/2	*	*
4/1/3	*	*
4/1/4	*	*
4/2/2	*	*
4/2/3	*	*
4/2/4	*	*

Appendix E: Proof of Theorem 1

We derive the theoretical performance of Graph Label Distribution Learning (GLDL). To proceed, we make some mild assumptions as follows. First, we consider G as simple, undirected graphs with no loop and the maximum degree of $d-1$. Second, the GCN has in total l layers with the maximum hidden dimension k. Third, the nodal feature vectors are normalized and reside in an ℓ_2-ball of radius B, such that $\|x_i^j\|_2 \leq B$, where x_i^j denotes the i-th node's representation at the j-th layer.

Denoted by $L_{\mathcal{G}}\left(f_w\right)$ and $L_{(X,A)}\left(f_w\right)$ are the generalization error over a graph distribution $\mathcal{G}$ and the empirical error on the training graph data (X,A), respectively, where $(X,A) \overset{\text{iid}}{\sim} \mathcal{G}$. We define $\phi(\cdot,\cdot)$ the distance metric such that $|\phi(u,p)-\phi(u,q)| \leq (\sqrt{p}+1)\|p-q\|_2$, $\forall u,p,q \in \mathbb{R}^m$. The error terms can be defined on the function f_w as follows.

$$L_{\mathcal{G}}\left(f_w\right) = \mathbb{E}_{(X,A)\sim\mathcal{G}}\mathbb{E}_{y_i\sim Y}\,\phi\left(f_w(X,A)[i], y_i\right),$$

$$L_{(X,A)}\left(f_w\right) = \frac{1}{nc}\sum_{i=1}^{n}\sum_{j=1}^{c} f_w(X,A)[i,j]\ln\frac{f_w(X,A)[i,j]}{y_{i,j}},$$

where $f_w(X,A)[i] \in \mathbb{R}^c$ and $y_i \in \mathbb{R}^c$ represent the predicted and ground-truth label distribution of the i-th node, respectively. Denoted by $f_w(X,A)[i,j]$ the predicted probability that node i belongs to the j-th class. We then have

Theorem 1 *For any $B>0, l>1$, let $f_w \in \mathcal{H}: \mathcal{X}\times\mathcal{G}\rightarrow\mathbb{R}^c$ be an l-layer GCN, parameterized by $W_1,\ldots,W_l$. Then for any $\delta,\gamma>0$, with probability at least $1-\delta$ we have*

$$L_{\mathcal{G}}\left(f_w\right) - L_{(X,A)}\left(f_w\right) \leq \frac{2(\sqrt{2q}+\sqrt{2})q}{\sqrt{n}}\max_{i\in[n],j\in[l]}\left\|\mathbf{x}_i^j\right\|_2 + 3b\sqrt{\frac{\log 2/\delta}{2n}} + \mathcal{O}\left(\sqrt{\frac{B^2d^{l-1}l^2k\log(lk)\mathcal{D}(W_i)+\log\frac{nl}{\delta}}{\gamma^2 n}}\right), \tag{1}$$

where $\mathcal{D}(W_i) = \prod_{i=1}^{l}\|W_i\|_2^2 \cdot \sum_{i=1}^{l}\left(\|W_i\|_F^2 / \|W_i\|_2^2\right)$ bounds the hypothesis space and b is a constant.

Proof We first try to explore the boundary of a GCN without label distribution, and use the multi-class γ-margin loss following (Bartlett, Foster, and Telgarsky 2017; Neyshabur, Bhojanapalli, and Srebro 2018). The generalization error

Table 8: The sensitivity of hidden dimension for GCN and MLP under fixed 50/30 frequency update and 2/2/3 Layers$_{comb}$. * means the value is far worse than the recorded results. All other not mentioned hyperparameters are fixed.

Hidden$_{comb}$	Val/Test
64/32	*
128/32	0.9229/0.9873
256/32	*0.8799/0.9345*
32/64	*
64/64	*
256/64	*0.879/0.9346*
512/64	**0.855/0.91**
32/128	*
64/128	*
128/128	0.9233/0.9879

and the empirical error are defined as,

$$L_{\mathcal{G}}\left(f_w\right)=\underset{z\sim\mathcal{D}}{\mathbb{P}}\left(f_w(X,A)[y]\leq\gamma+\max_{j\neq y}f_w(X,A)[j]\right),$$
$$L_{(X,A)}\left(f_w\right)=\frac{1}{n}\sum_{z_i\in S}\mathbf{1}\left(f_w(X,A)[y]\leq\gamma+\max_{j\neq y}f_w(X,A)[j]\right), \tag{2}$$

where $\gamma>0$ and $f_w(X,A)$ is the l-th layer representations, i.e., $H_l=f_w(X,A)$. $\mathbf{1}$ is a all-one vector. Two necessary lemmas are derived from this. First, a margin-based generalization bound is given to guarantee that, as long as the change of the output brought by the perturbations is small with a large probability, the corresponding generalization bound (Neyshabur, Bhojanapalli, and Srebro 2018) is defined as:

Lemma 1 *Let $f_w(x):\mathcal{X}\rightarrow\mathbb{R}^K$ be any model with parameters w, and let P be any distribution on the parameters that is independent of the training data. For any w, we construct a posterior $Q(w+u)$ by adding any random perturbation u to w, s.t., $\mathbb{P}\left(\max_{x\in\mathcal{X}}\left|f_{w+u}(x)-f_w(x)\right|_\infty<\frac{\gamma}{4}\right)>\frac{1}{2}$. Then, for any $\gamma,\delta>0$, with probability at least $1-\delta$ we have:*

$$L_{\mathcal{G}}\left(f_w\right)\leq L_{(X,A)}\left(f_w\right)+\sqrt{\frac{2D_{\mathrm{KL}}(Q(w+u)\|P)+\log\frac{8n}{\delta}}{2(n-1)}}, \tag{3}$$

where n is the number of instances of training set. Hence, in order to apply Lemma 1 , we must ensure that the change of the output brought by the weight perturbations is small with a large probability. In the following lemma, we bound this change using the product of the spectral norms of learned weights at each layer and a term depending on some statistics of the graph(Liao, Urtasun, and Zemel 2020).

Lemma 2 *For any $B>0, l>1$, let $f_w\in\mathcal{H}:\mathcal{X}\times\mathcal{G}\rightarrow\mathbb{R}^K$ be a l-layer GCN. Then for any w, and $x\in\mathcal{X}_{B,h_0}$, and any perturbation $u=\operatorname{vec}\left(\{U_i\}_{i=1}^l\right)$ such that $\forall i\in\mathbb{N}_l^+,\|U_i\|_2\leq\frac{1}{l}\|W_i\|_2$, the change in the output of GCN is bounded as,*

$$\left|L_{(X,A)}\left(f_{(w+u)}\right)-L_{(X,A)}\left(f_w\right)\right|_2 \leq eBd^{\frac{l-1}{2}}\left(\prod_{i=1}^{l}\|W_i\|_2\right)\sum_{k=1}^{l}\frac{\|U_k\|_2}{\|W_k\|_2}. \tag{4}$$

Then, we can deduce the bound for a GCN without label distribution. First, let $\beta=\left(\prod_{i=1}^l\|W_i\|_2\right)^{1/t}$. Weights are normalized as $\tilde{W}_i=\frac{\beta}{\|W_i\|_2}W_i$. Due to the homogeneity of ReLU, i.e., $a\phi(x)=\phi(ax),\forall a\geq0$. $\prod_{i=1}^l\|W_i\|_2=\prod_{i=1}^l\left\|\tilde{W}_i\right\|_2$ and $\|W_i\|_F/\|W_i\|_2=\left\|\tilde{W}_i\right\|_F/\left\|\tilde{W}_i\right\|_2$, i.e., the terms present in the bound remain unchanged after normalization. Therefore, w.l.o.g., we conduct the assumption that the norm is equal across all layers, i.e., $\forall i,\|W_i\|_2=\beta$. Consider the prior $P=\mathcal{N}\left(0,\sigma^2 I\right)$ and the random perturbation $u\sim\mathcal{N}\left(0,\sigma^2 I\right)$. Note that the σ of the prior and the perturbation keep the same and will be set according to β. Specifically, the value of σ is set based on some approximation $\tilde{\beta}$ of β since the prior P can not be directly deduced by any learned weights. We select the approximation $\tilde{\beta}$ as the cover set that encompasses the possible significant range of β. For the moment, assuming we have a constant $\tilde{\beta}$, we can examine β that fulfills the condition $|\beta-\tilde{\beta}|\leq\frac{1}{l}\beta$. Note that this also implies

$$\begin{aligned}|\beta-\tilde{\beta}|\leq\frac{1}{l}\beta,&\Rightarrow\left(1-\frac{1}{l}\right)\beta\leq\tilde{\beta}\leq\left(1+\frac{1}{l}\right)\beta,\\&\Rightarrow\left(1-\frac{1}{l}\right)^{l-1}\beta^{l-1}\leq\tilde{\beta}^{l-1}\leq\left(1+\frac{1}{l}\right)^{l-1}\beta^{l-1},\\&\Rightarrow\left(1-\frac{1}{l}\right)^{l}\beta^{l-1}\leq\tilde{\beta}^{l-1}\leq\left(1+\frac{1}{l}\right)^{l}\beta^{l-1},\\&\Rightarrow\frac{1}{e}\beta^{l-1}\leq\tilde{\beta}^{l-1}\leq e\beta^{l-1}.\end{aligned} \tag{5}$$

According to (Tropp 2012), for random perturbations $U_i\in\mathbb{R}^{h\times h}$ which is object to the distribution $U_i\sim\mathcal{N}\left(\mathbf{0},\sigma^2 I\right)$,

$$\mathbb{P}\left(\|U_i\|_2\geq t\right)\leq 2ke^{-t^2/2k\sigma^2}. \tag{6}$$

We consider the perturbations of all layers, thus:

$$\begin{aligned}\mathbb{P}\left(\|U_1\|_2<t\&\cdots\&\|U_l\|_2<t\right)&=1-\mathbb{P}\left(\exists i,\|U_i\|_2\geq t\right),\\&\geq1-\sum_{i=1}^{l}\mathbb{P}\left(\|U_i\|_2\geq t\right),\\&\geq1-2lke^{-t^2/2k\sigma^2}.\end{aligned} \tag{7}$$

Setting $2lke^{-t^2/2k\sigma^2}=\frac{1}{2}$, we can have $t=\sigma\sqrt{2k\log(4lk)}$.

The likelihood that the spectral norm perturbation of any layer not exceeds $\sigma\sqrt{2h\log(4lk)}$ is at least $\frac{1}{2}$. Plugging this bound into Lemma 2, we have with probability at least $\frac{1}{2}$,

$$\begin{aligned}
|f_{w+u}(X,A) - f_w(X,A)|_2 &\le eBd^{\frac{l-1}{2}}\left(\prod_{i=1}^{l}\|W_i\|_2\right)\sum_{i=1}^{l}\frac{\|U_i\|_2}{\|W_i\|_2}, \\
&= eBd^{\frac{l-1}{2}}\beta^l\sum_{i=1}^{l}\frac{\|U_i\|_2}{\beta}, \\
&\le eBd^{\frac{l-1}{2}}\beta^{l-1}l\sigma\sqrt{2k\log(4lk)}, \\
&\le e^2Bd^{\frac{l-1}{2}}\tilde{\beta}^{l-1}l\sigma\sqrt{2k\log(4lk)}, \\
&\le \frac{\gamma}{4},
\end{aligned} \tag{8}$$

where we can set $\sigma = \frac{\gamma}{4e^2Bd^{\frac{L-1}{2}}\tilde{\beta}^{l-1}l\sqrt{k\log(4lk)}}$ to get a new inequality. Note that Lemma 2 also requires $\forall i \in \mathbb{N}_l^+, \|U_i\|_2 \le \frac{1}{l}\|W_i\|_2$. The requirement is satisfied if $\sigma \le \frac{\beta}{l\sqrt{2k\log(4lk)}}$ which in turn can be satisfied if

$$\frac{\gamma}{4eBd^{\frac{l-1}{2}}\beta^{l-1}l\sqrt{2k\log(4lk)}} \le \frac{\beta}{l\sqrt{2k\log(4lk)}}, \tag{9}$$

since the chosen value of σ satisfies $\sigma \le \frac{\gamma}{4eBd^{\frac{l-1}{2}}\beta^{l-1}l\sqrt{2k\log(4lk)}}$.

Note that (9) can be rewritten as $\frac{\gamma}{4eB}d^{\frac{1-1}{2}} \le \beta^l$. KL term in the PAC-Bayes bound in Lemma 1 is deduced as:

$$\begin{aligned}
\mathrm{KL}(Q\|P) &= \frac{|w|_2^2}{2\sigma^2} = \frac{4e^2B^2d^{l-1}\beta^{2l-2}l^2k\log(4lk)}{2\gamma^2}\sum_{i=1}^{l}\|W_i\|_F^2, \\
&\le \mathcal{O}\left(\frac{B^2d^{l-1}\beta^{2l}l^2k\log(lk)}{\gamma^2}\sum_{i=1}^{l}\frac{\|W_i\|_F^2}{\beta^2}\right), \\
&\le \mathcal{O}\left(B^2d^{l-1}l^2k\log(lk)\frac{\prod_{i=1}^{l}\|W_i\|_2^2}{\gamma^2}\sum_{i=1}^{l}\frac{\|W_i\|_F^2}{\|W_i\|_2^2}\right).
\end{aligned} \tag{10}$$

From Lemma 1, fixing any $\tilde{\beta}$, with probability $1-\delta$ and for all w such that $|\beta - \tilde{\beta}| \le \frac{1}{l}\beta$. Let $\mathcal{D}(W_i) = \prod_{i=1}^{l}\|W_i\|_2^2 \cdot \sum_{i=1}^{l}\left(\|W_i\|_F^2 / \|W_i\|_2^2\right)$ we can get,

$$\begin{aligned}
L_{\mathcal{G}}(f_w) &\le L_{(X,A)}(f_w) \\
&\quad + \mathcal{O}\left(\sqrt{\frac{B^2d^{l-1}l^2k\log(lk)\mathcal{D}(W_i) + \log\frac{n}{\delta}}{\gamma^2 n}}\right).
\end{aligned} \tag{11}$$

The choice of $\tilde{\beta}$ is important, such that for any β, we can bound the generalization error like (11). First, the value range of β is decided as the following,

$$\frac{1}{\sqrt{d}}\left(\frac{\gamma\sqrt{d}}{2B}\right)^{1/l} \le \beta \le \frac{1}{\sqrt{d}}\left(\frac{\gamma\sqrt{nd}}{2B}\right)^{1/l}, \tag{12}$$

since otherwise the bound holds trivially as $L_{(X,A)}(f_w) \le 1$ by definition. Note that the lower bound in (12) ensures that (9) holds which in turn justifies the applicability of Lemma 2. If $\beta < \frac{1}{\sqrt{d}}\left(\frac{\gamma\sqrt{d}}{2B}\right)^{1/l}$, then for any (X,A) and any $j \in \mathbb{N}_K^+, |f(X,A)[j]| \le \frac{\gamma}{2}$. To prove this, we have,

$$\begin{aligned}
|f_w(X,A)[j]| &\le |f_w(X,A)|_2 = \left|\frac{1}{n}\mathbf{1}_n\mathbf{x}^{l-1}W_l\right|_2, \\
&\le \frac{1}{n}\left|\mathbf{1}_n\mathbf{x}^{l-1}\right|_2\|W_l\|_2, \\
&\le \|W_l\|_2\max_i\left|\mathbf{x}^{l-1}[i,i]\right|_2, \\
&\le Bd^{\frac{l-1}{2}}\prod_{i=1}^{l}\|W_i\|_2 = d^{\frac{l-1}{2}}\beta^l B, \\
&= d^{\frac{l-1}{2}}B\frac{\gamma}{2Bd^{\frac{l-1}{2}}} \le \frac{\gamma}{2}.
\end{aligned} \tag{13}$$

Therefore, we always have $L_{(X,A)}(f_w) = 1$ when $\beta < \frac{1}{\sqrt{d}}\left(\frac{\gamma\sqrt{d}}{2B}\right)^{1/l}$. Alternatively, if $\beta > \frac{1}{\sqrt{d}}\left(\frac{2\sqrt{nd}}{2B}\right)^{1/l}$, the term inside the big-O notation in (11) would be,

$$\begin{aligned}
\sqrt{\frac{B^2d^{l-1}l^2k\log(lk)\mathcal{D}(W_i) + \log\frac{n}{\delta}}{\gamma^2 n}} &\ge \sqrt{\frac{l^2k\log(lk)}{4}\sum_{i=1}^{l}\frac{\|W_i\|_F^2}{\|W_i\|_2^2}}, \\
&\ge \sqrt{\frac{l^2k\log(lk)}{4}} \ge 1,
\end{aligned} \tag{14}$$

where we set $k \ge 2$ in practice and $l \ge 2$, while considering the fact $\|W_i\|_F \ge \|W_i\|_2$. To account for β within the range defined by (12), a condition ensuring that $|\beta - \tilde{\beta}| \le \frac{1}{l}\beta$ would be $|\beta - \bar{\beta}| \le \frac{1}{l\sqrt{d}}\left(\frac{\gamma\sqrt{d}}{2B}\right)^{1/l}$. So, if we can identify an overlap of the span in (12) with a radius of $\frac{1}{l\sqrt{d}}\left(\frac{\partial\sqrt{d}}{2B}\right)^{1/l}$ and ascertain that bounds akin to (11) are met when $\tilde{\beta}$ assumes any value within that overlap, then a bound that is valid for every β can be established. Clearly, it is only essential to ponder a coverage C of magnitude $|C| = \frac{l}{2}\left(n^{\frac{1}{2}} - 1\right)$. Thus, representing the event of (11) when $\tilde{\beta}$ adopts the i-th value from the coverage as E_i, we deduce:

$$\begin{aligned}
\mathbb{P}\left(E_1\&\cdots\&E_{|C|}\right) &= 1 - \mathbb{P}\left(\exists i, \bar{E}_i\right), \\
&\ge 1 - \sum_{i=1}^{|C|}\mathbb{P}\left(\bar{E}_i\right) \ge 1 - |C|\delta.
\end{aligned} \tag{15}$$

Note $\bar{E}_i$ denotes the complement of E_i. Hence, we now present the PAC-Bayes generalization bound (Liao, Urtasun, and Zemel 2020) of GCNs as Theorem 2.

Theorem 2 *For any $B > 0, l > 1$, let $f_w \in \mathcal{H} : \mathcal{X} \times \mathcal{G} \to \mathbb{R}^K$ be a l layer GCN. Then for any $\delta, \gamma > 0$, with probability at least $1 - \delta$ over the choice of an i.i.d. size-n training set*

S according to $\mathcal{D}$, for any w, we have,

$$\begin{aligned}
&L_{\mathcal{G}}\left(f_w\right) \leq L_{(X,A)}\left(f_w\right) \\
&\quad+\mathcal{O}\left(\sqrt{\frac{B^2 d^{l-1} l^2 k \log(lk)\mathcal{D}(W_i)+\log\frac{n|C|}{\delta}}{\gamma^2 n}}\right), \\
&=L_{(X,A)}\left(f_w\right) \\
&\quad+\mathcal{O}\left(\sqrt{\frac{B^2 d^{l-1} l^2 k \log(lk)\mathcal{D}(W_i)+\log\frac{nl}{\delta}}{\gamma^2 n}}\right),
\end{aligned} \tag{16}$$

Then we consider a GCN with label distribution learning, and explore the bound in term of Rademacher complexity. Specifically, let $\eta(\cdot)$ be a label distribution function, thus $\eta\left(\mathbf{x}_i\right)=\left\{\eta_{\mathbf{x}_i}^{y_1},\ldots,\eta_{\mathbf{x}_i}^{y_p}\right\}$. Given a function class $\mathcal{H}$ and a loss function $\ell(\cdot)$, for function $h\in\mathcal{H}$, its corresponding risk and empirical risk are defined as $L_{\mathcal{G}}\left(f_w\right)=\mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[\ell(h(\mathbf{x}),\eta(\mathbf{x}))]$ and $L_{(X,A)}\left(f_w\right)=\frac{1}{n}\sum_{i=1}^n\ell\left(h\left(\mathbf{x}_i\right),\eta\left(\mathbf{x}_i\right)\right)$, respectively. Recall the definition of rademacher complexity w.r.t. S and ℓ,

$$\hat{\mathcal{R}}_n(\ell\circ\mathcal{H}\circ S)=\mathbb{E}_{\epsilon_1,\ldots,\epsilon_n}\left[\sup_{h\in\mathcal{H}}\frac{1}{n}\sum_{i=1}^n\ell\left(h\left(\mathbf{x}_i\right),\eta\left(\mathbf{x}_i\right)\right)\epsilon_i\right],$$

where $\epsilon_1,\ldots,\epsilon_n$ are n independent rademacher random variables with $\mathbb{P}\left(\epsilon_i=1\right)=\mathbb{P}\left(\epsilon_i=-1\right)=1/2$.

Then we can derive the following necessary lemma (Bartlett and Mendelson 2002) (Schapire and Freund 2012),

Lemma 3 *Let $\mathcal{H}$ be a family of functions. For a loss function ℓ bounded by μ, then for any $\delta>0$, with probability at least $1-\delta$, for all $h\in\mathcal{H}$ such that*

$$L_{\mathcal{D}}(h)\leq L_S(h)+2\hat{\mathcal{R}}_n(\ell\circ\mathcal{H}\circ S)+3\mu\sqrt{\frac{\log 2/\delta}{2n}}. \tag{17}$$

and the theorem of Rademacher complexity of the method for loss function KL-divergence (Wang and Geng 2019):

Theorem 3 *Let $\mathcal{H}$ be a family of functions for multi-output linear regression, and $\mathcal{H}_j$ be a family of functions for the j-th output. Rademacher complexity of ME(Maximum-Entropy) with KL loss satisfies*

$$\hat{\mathcal{R}}_n(\mathrm{KL}\circ\mathrm{SF}\circ\mathcal{H}\circ S)\leq(\sqrt{2q}+\sqrt{2})\sum_{j=1}^q\hat{\mathcal{R}}_n\left(\mathcal{H}_j\circ S\right). \tag{18}$$

Note that $\mathrm{KL}(\mathbf{u},\cdot)$ is not ρ-Lipschitz over $\mathbb{R}^m$ for any $\rho\in\mathbb{R}$ and $\mathbf{u}\in\mathbb{R}^m$. Define function $\phi(\cdot,\cdot)$ as $\mathrm{KL}(\cdot,\mathrm{SF}(\cdot))$. Next we show that $\phi(\mathbf{u},\cdot)$ satisfy Lipschitzness. For $\mathbf{p},\mathbf{q}\in\mathbb{R}^m$,

$$|\phi(\mathbf{u},\mathbf{p})-\phi(\mathbf{u},\mathbf{q})|=|\mathrm{KL}(\mathbf{u},\mathrm{SF}(\mathbf{p}))-\mathrm{KL}(\mathbf{u},\mathrm{SF}(\mathbf{q}))|,$$

which equals

$$\begin{aligned}
&\left|\sum_{i=1}^p u_i\left(\ln\frac{\exp\left(p_i\right)}{\sum_{j=1}^n\exp\left(p_j\right)}-\ln\frac{\exp\left(q_i\right)}{\sum_{j=1}^n\exp\left(q_j\right)}\right)\right|, \\
&\leq\sum_{i=1}^p\left|\ln\left(1+\sum_{j\neq i}\mathrm{e}^{p_j-p_i}\right)-\ln\left(1+\sum_{j\neq i}\mathrm{e}^{q_j-q_i}\right)\right|u_i.
\end{aligned}$$

Observing that $\ln\left(1+\sum_j\exp v_i\right)$ is 1-Lipschitz for $\mathbf{v}\in\mathbb{R}^m$, thus right-hand side of preceding equation is bounded by

$$\begin{aligned}
&\sum_{i=1}^p u_i\left\|\mathbf{p}-\mathbf{1}\cdot p_i-\mathbf{q}+\mathbf{1}\cdot q_i\right\|_2, \\
&\leq\|\mathbf{p}-\mathbf{q}\|_2+\sqrt{c}\sum_{i=1}^p u_i\left|p_i-q_i\right|, \\
&\leq(\sqrt{p}+1)\|\mathbf{p}-\mathbf{q}\|_2,
\end{aligned}$$

namely, ϕ is $(\sqrt{p}+1)$-Lipschitz. According to (Maurer 2016), we have:

$$\begin{aligned}
\hat{\mathcal{R}}_n(\mathrm{KL}\circ\mathrm{SF}\circ\mathcal{H}\circ S)&\leq\sqrt{2}(\sqrt{q}+1)\sum_{j=1}^q\hat{\mathcal{R}}_n\left(\mathcal{H}_j\circ S\right), \\
&=(\sqrt{2q}+\sqrt{2})\sum_{j=1}^q\hat{\mathcal{R}}_n\left(\mathcal{H}_j\circ S\right),
\end{aligned}$$

Although only implementing KL-divergence does not satisfy Lipschitzness, the combination of KL and Softmax(SF) is $(\sqrt{p}+1)$-Lipschitz. Define class of functions of j-th output with weight constraints as $\mathcal{H}_j=\left\{x\rightarrow\mathbf{w}_j\cdot x:\left\|\mathbf{w}_j\right\|_2\leq 1\right\}$. According to (Kakade, Sridharan, and Tewari 2008), rademacher complexity of $\mathcal{H}_j$ satisfies

$$\hat{\mathcal{R}}_n\left(\mathcal{H}_j\circ S\right)\leq\frac{\max_{i\in[n]}|\ \|x_i\|_2}{\sqrt{n}}.$$

Then right-hand side of (18) is bounded as

$$\hat{\mathcal{R}}_n(\mathrm{KL}\circ\mathrm{SF}\circ\mathcal{H}\circ S)\leq\frac{(\sqrt{2q}+\sqrt{2})q}{\sqrt{n}}\max_{i\in[n]}\left\|x_i\right\|_2. \tag{19}$$

According to Lemma 3, while we have the rademacher complexity of LDL with KL-divergence loss defined in (19), the bound can be defined as:

$$\begin{aligned}
L_{\mathcal{G}}\left(f_w\right)-L_{(X,A)}\left(f_w\right)\leq&\ 2\frac{(\sqrt{2q}+\sqrt{2})q}{\sqrt{n}}\max_{i\in[n]}\left\|x_i\right\|_2 \\
&+3\mu\sqrt{\frac{\log 2/\delta}{2n}}.
\end{aligned} \tag{20}$$

Considering the node representations vary in different layers, Without loss of generality, we replace $\max_{i\in[n]}\left\|x_i\right\|_2$ with $\max_{i\in[n],j\in[l]}\left\|\mathbf{x}_i^j\right\|_2$ to find the maximum node representation, instead of node feature. Then recall the PAC risk bound defined in (16),

$$\begin{aligned}
&L_{\mathcal{G}}\left(f_w\right)-L_{(X,A)}\left(f_w\right) \\
&\quad\leq\mathcal{O}\left(\sqrt{\frac{B^2 d^{l-1} l^2 k \log(lk)\mathcal{D}(W_i)+\log\frac{nl}{\delta}}{\gamma^2 n}}\right),
\end{aligned}$$

We use the sum of both above bounds as the final risk bound for a GCN with KL-divergence loss for the label distribution learning problem.

$$L_{\mathcal{G}}(f_w) - L_{(X,A)}(f_w) \leq \frac{2(\sqrt{2q}+\sqrt{2})q}{\sqrt{n}} \max_{i\in[n],j\in[l]} \left\| \mathbf{x}_i^j \right\|_2 + 3b\sqrt{\frac{\log 2/\delta}{2n}} + \mathcal{O}\left(\sqrt{\frac{B^2 d^{l-1} l^2 k \log(lk)\mathcal{D}(W_i) + \log\frac{nl}{\delta}}{\gamma^2 n}}\right), \tag{21}$$

As (Cha 2007) suggests that 0 is replaced by a very small value, say $\gamma > 0$, for division by 0 when implementing KL divergence, then for probability distribution $\mathbf{p}, \mathbf{q} \in \mathbb{R}^m$ with $p_i \geq \gamma, q_i \geq \gamma$

$$\mathrm{KL}(\mathbf{p}, \mathbf{q}) = \sum_{i=1}^{n} p_i \ln \frac{p_i}{q_i} \leq \sum_{i=1}^{n} p_i \ln \frac{1}{\gamma} \leq -\ln\gamma,$$

thus there exists a constant $b \geq -\ln\gamma$ such that $\mathrm{KL}(\cdot,\cdot) \leq b$ (e.g., $b = 35$ for $\gamma = 1 \times 10^{-15}$).

Hence, all of the above concludes the proof.

References

Bartlett, P. L., and Mendelson, S. 2002. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research* 3(Nov):463–482.

Bartlett, P. L.; Foster, D. J.; and Telgarsky, M. J. 2017. Spectrally-normalized margin bounds for neural networks. *NeurIPS* 30.

Cha, S.-H. 2007. Comprehensive survey on distance/similarity measures between probability density functions. *City* 1(2):1.

Fu, X.; Zhang, J.; Meng, Z.; and King, I. 2020. Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding. In *Proceedings of The Web Conference 2020*, WWW '20, 2331–2341. New York, NY, USA: Association for Computing Machinery.

Kakade, S. M.; Sridharan, K.; and Tewari, A. 2008. On the complexity of linear prediction: Risk bounds, margin bounds, and regularization. *Advances in neural information processing systems* 21.

Liao, R.; Urtasun, R.; and Zemel, R. 2020. A pac-bayesian approach to generalization bounds for graph neural networks. In *ICLR*.

Ma, Y.; Liu, X.; Shah, N.; and Tang, J. 2022. Is homophily a necessity for graph neural networks? In *International Conference on Learning Representations*.

Maurer, A. 2016. A vector-contraction inequality for rademacher complexities. In *Algorithmic Learning Theory: 27th International Conference, ALT 2016, Bari, Italy, October 19-21, 2016, Proceedings 27*, 3–17. Springer.

Neyshabur, B.; Bhojanapalli, S.; and Srebro, N. 2018. A pac-bayesian approach to spectrally-normalized margin bounds for neural networks. In *International Conference on Learning Representations*.

Schapire, R. E., and Freund, Y. 2012. Foundations of machine learning.

Tang, J.; Zhang, J.; Yao, L.; Li, J.; Zhang, L.; and Su, Z. 2008. Arnetminer: Extraction and mining of academic social networks. In *KDD'08*, 990–998.

Tropp, J. A. 2012. User-friendly tail bounds for sums of random matrices. *Foundations of computational mathematics* 12:389–434.

Wang, J., and Geng, X. 2019. Theoretical analysis of label distribution learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 5256–5263.