ON LOCALITY IN GRAPH LEARNING VIA GRAPH NEURAL NETWORK

Anonymous authors
Paper under double-blind review

ABSTRACT

Theoretical understanding of the learning process of graph neural network (GNN) has been lacking, especially for node-level tasks where data are not independent and identically-distributed (I.I.D.). The common practice in GNN training is to adopt strategies from other machine learning families, despite the striking differences between learning non-graph and graph-structured data. This results in unstable learning quality (e.g., accuracy) for GNN. In this paper, we study the relation between the training set in the input graph and the performance of GNN. Combining the topology awareness of GNN and the dependence (topology) of data samples, we formally derive a structural relation between the performance of GNN and the coverage of the training set in the graph. More specifically, the distance of the training set to the rest of the vertexes in the graph is negatively correlated to the learning outcome of GNN. We further validate our theory on different graph data sets with extensive experiments. Using the derived result as guidance, we also investigate the initial data labelling problem in active learning of GNN and show that locality-aware data labelling substantially outperforms the prevailing random sampling approach.

1 INTRODUCTION

Graph Neural Network (GNN) is a family of machine learning (ML) models tailored for learning from graph-structured data (Duvenaud et al., 2015; Li et al., 2017b; Gilmer et al., 2017; You et al., 2019). Recently, great success has been shown using models such as GCN (Kipf & Welling, 2017), GraphSAGE (Hamilton et al., 2018) and GAT (Veličković et al., 2017) on tackling various graph-related problems in domains such as chemistry (Gilmer et al., 2017), biology (Duvenaud et al., 2015), social networking (Hamilton et al., 2018; Chen et al., 2018; 2017b) and traffic networks (Li et al., 2017a). Despite their practicality and potential, theoretical understanding of the learning process of GNNs is still underexplored, especially given their difference in data dependency from other ML families.

In most ML settings, data samples are assumed to be created independently and identically (I.I.D.) from a certain distribution. Because of this, uniform sampling is effective in many data selection processes such as training/validation/test set partition, batch sampling and initial labelling set selection (Settles, 2009; Ren et al., 2020). For learning from graph-structured data, inherent dependencies exist among data samples, as captured by the graph structure. Taking training/validation/test set partition for example, different random partitions of the labelled nodes significantly impact the performance of the GNN model (Shchur et al., 2019), especially when the size of the training set is small relative to the data population. Fig. 1 illustrates this phenomenon on the Cora data set (Sen et al., 2008). This calls for a better understanding of the relation between the dependence among data samples and the learning process of GNN.

Network homophily (McPherson et al., 2001) is a fundamental principle that describes the dependence among data samples in many real-world networks. Network homophily implies that nodes close by in (graph) distance tend to have similar features and labels. Most existing GNN designs assume strong homophily in the graph data (Zhu et al., 2020) and exploit it by propagating features and aggregating them within various graph neighbourhoods using different mechanisms (Hamilton et al., 2018; Veličković et al., 2017; Kipf & Welling, 2017). The capability to capture topological information from the underlying graph structure is the key to contribute to the success of GNN.

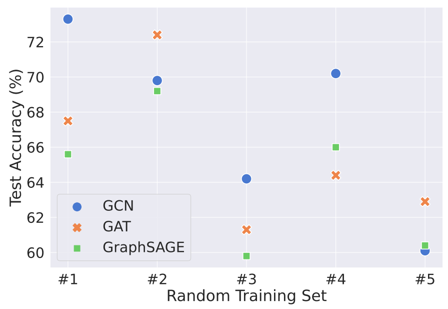

Figure 1: Performance of GCN, GraphSAGE and GAT on Cora dataset with different random partitions of training set (of size 35) and test set (within the labelled data set).

The following question is raised: can we combine this network principle and topology awareness of GNN to derive useful insights for more effective graph learning via GNN? Most existing efforts focus on the (vertex) sampling strategy to accelerate GNN training (Chen et al., 2018) or to conserve memory during GNN training (Ying et al., 2018). There is little understanding or work on the underlying principle between the dependency of the data samples and the performance of GNN models Ma et al. (2021).

To address this question, we theoretically and experimentally investigate how and why different random partitions of training sets affect the learning outcome (test performance) of GNN in node-level tasks. We focus on graph data sets where the network homophily property holds. We formally show the spatial locality in the inference ability of GNN (under certain conditions), i.e., GNN predicts better on vertexes close to the training set.

Our main contributions are summarized as follows.

1. We prove that GNN predicts better on vertexes that are closer to the training set under certain assumptions. This implies that better coverage of the training set over the rest of the graph (in terms of neighbourhoods of a small radius) can translate into better generalization performance of the GNN model in the semi-supervised setting.
2. We experimentally validate our theory and assumptions on Cora, Citeseer and PubMed data sets with prevailing GNN models such as GCN, GAT and GraphSAGE.
3. Using the observation and theory as guidance, we formulate two optimization problems for studying (1) the "cold start" problem in active learning (Grimova et al., 2018), and (2) minimal size of the labelled data set with respect to the model performance. The first optimization provides a strategy for selecting the most "economic" initial data set to be labelled and the other gives insights on how many vertexes should be labelled initially for the desired learning outcome of a GNN model.

This work represents our first attempt to unwrap a better understanding of the learning process of GNN. It is novel to combine the network principle of the dependence among data samples and topology awareness of GNN to formally infer useful insights (based on graph structure) for GNN learning. The results suggest that the topology awareness of GNN is not only an important property to improve GNN performance, as shown in (Li et al., 2020; You et al., 2019; Xu et al., 2018; Nishad et al., 2021), but also useful guidance for making informed decisions in the learning process of GNN.

2 Related Work

Expressive Power of GNN. Modern GNN architectures are inspired by the Weisfeiler–Lehman (WL) isomorphism test (Leman & Weisfeiler, 1968) which uses the graph structure to propagate information (Kipf & Welling, 2017). One idea in characterizing the power of GNN is to measure its ability in differentiating different graph structures, referred to as the expressive power of GNN (we refer to Sato (2020) for a survey on this line of works). In particular, Xu et al. (2018) proved that the expressive power of GNNs is no more than the WL test. This has inspired a number of studies on improving the expressive power of GNN by enabling it to differentiate graph structures that cannot be distinguished by the WL-test (Morris et al., 2020). The studies in (You et al., 2019; Li et al., 2020; Nishad et al., 2021; Zhou et al., 2020) show that combining graph topology information (distance) into the learning process of GNN (e.g., for feature encoding, aggregation, normalization) can improve the performance of GNN on various tasks.

Generalization of GNN. Studies aiming to understand generalization performance of GNNs on node-level tasks are rather limited (Ma et al., 2021). To our best knowledge, there are two very recent studies that investigate/demonstrate similar observations as ours. Zhu et al. (2021) studied the problem in GNN learning that the test set and the training set are generated from different distributions. They found the test error of GNNs is inversely related to their defined metric (distance) between test set and training set, which is similar to our discovery here. However, their work focused on designing a regularization mechanism to allow the learnt distribution to approximate the distribution in the test set. Ma et al. (2021) studied a similar problem as ours by extending the PAC-Bayesian analysis for I.I.D. data to analyzing the generalization performance of GNNs. They prove that given a fixed training set, GNN generalization performance varies among different subgroups of the test population defined by a feature distance. Our study complements their study in the case that graph distance metric is used. Our analysis provides a different and more direct perspective for understanding how graph structure information influences GNN learning performance.

3 Preliminaries

3.1 Graph and Notation

Let $G = (V, E)$ be the input graph with node feature vector X_v for all $v \in V$. In this paper, we assume G is connected and focus our analysis on the node classification task where a label y_v is associated with vertex v, and the objective is to learn a representation vector h_v of v such that v's label can be predicted as $y_v = f(h_v)$, where f is the prediction function. We use $d(x, y)$ to denote the distance between x and y: (1) $d(.,.)$ is defined to be the shortest path distance if x and y are vertexes in the graph; (2) $d(.,.)$ is the Euclidean distance if x and y are representations in the embedding space; (3) $d(.,.)$ is the distance between x and the closest element in Y, if Y is a set. For example, let u be a vertex and D be a set of vertexes, and then $d(u, D) := \min_{v \in D}\{d(u, v)\}$. Let $\mathbb{N}$ and $\mathbb{R}$ denote the set of natural numbers and real numbers, respectively. $\mathbb{R}_+$ is the set of non-negative real numbers.

3.2 Graph Neural Network

GNN combines the graph structure and node features X_v to learn a representation vector h_v of node v. Modern GNNs adopt a neighborhood aggregation scheme, where the representation of a node is updated iteratively by aggregating representations of its neighbors. After k iterations of aggregation, a node's representation captures the structural information within its k-hop neighborhood in the graph. Let $\mathcal{H} \subseteq \mathbb{R}^m$ be the embedding space of the learnt representations of vertexes and $m \in \mathbb{N}$ is the dimension. $\mathcal{M}_\theta : V \mapsto \mathcal{H}$ denotes the GNN model with parameter θ that maps a vertex $v \in V$ into a representation vector h_v in the embedding space $\mathcal{H}$, following a neighborhood aggregation scheme. Let $f : \mathcal{H} \mapsto \mathbb{R}^c$ be the prediction function that maps an embedding vector h_v to a vector in $\mathbb{R}^c$, in which $c \in \mathbb{N}$ is the number of classes and each entry represents the probability of belonging to the respective class. The GNN prediction process can be viewed as composition of $f \circ \mathcal{M}_\theta : V \mapsto \mathbb{R}^c$. Let $\mathcal{L} : \mathbb{R}^c \mapsto \mathbb{R}_+$ denote the loss function which evaluates how accurate the prediction is.

As the graph structure (topology) among vertexes provides important information, it is important for GNN to preserve as much topological information as possible for embedding. Indeed, it has been formally and experimentally shown in (Li et al., 2020; You et al., 2019; Nishad et al., 2021) that improving the ability of GNN to preserve topological information such as shortest path distance can improve the expressive power of GNN. Here, we explore how to combine the topology awareness of GNN with the network homophily principle to derive useful insight for GNN learning. In particular, we focus on the ability of GNN to preserve distances information among vertexes, i.e., to have a low distortion between graph distance and embedding distance. The distortion of a function between two metric spaces is defined as follows.

Definition 1 (distortion). *Given two metric spaces $(\mathcal{E}, d)$ and $(\mathcal{E}', d')$ and a mapping $f : \mathcal{E} \mapsto \mathcal{E}'$, f is said to have distortion α, if there exists a constant $r > 0$ such that $\forall u, v \in \mathcal{E}$, $rd(u, v) \leq d'(f(u), f(v)) \leq \alpha r d(u, v)$*

You et al. (2019); Li et al. (2020); Nishad et al. (2021) have presented effective ways to increase the awareness of GNNs to vertex distances, i.e, to decrease the distortion rate α. For example, one can encode the shortest path distance (SPD) or other graph structure information to be part of the node feature for each vertex (Li et al., 2020). Based on Bourgains' theorem (Bourgain, 1985), You et al. (2019) propose a vertex-distance-aware mechanism by selecting a set of anchor vertexes to provide positional information, and use the positional information to adjust the intermediate activation in GNN learning process. It is also discussed in (You et al., 2019) that the commonly used GNNs are a local version of the vertex-distance-aware mechanism they proposed, as the commonly used GNNs aggregate information for the training node from its neighbourhood. This neighbourhood aggregation mimics the existence of anchor vertexes and can provide local position information. This implies that common GNNs are already equipped with some ability to capture vertex distances (low distortion), at least between neighbours whose representations are aggregated. We conduct experiments in Sec. 4.3 to further validate this.

4 Locality in the Learning Process of GNN

4.1 Locality in GNN Performance

We now show how to use GNN properties to derive local structural behaviour in GNN learning. We make the following assumptions for our theoretical analysis.

Assumption 1 (Local curvature)**.** *For all representations $h \in \mathcal{H}$, $\frac{d}{dh}\mathcal{L}(f(h))$ and $\frac{d^2}{d^2h}\mathcal{L}(f(h))$ exist, and are continuous and bounded.*

Assumption 2 (Well-trained)**.** *For a given training set D, the training process of the GNN converges to a set of parameters θ_D such that for any $\epsilon > 0$ and $v \in D$, we have $\mathcal{L}(f(\mathcal{M}_{\theta_D}(v))) < \epsilon$.*

Assumption 1 is a standard technical assumption to make the analysis feasible. Regarding Assumption 2, it has been formally proved that GNNs can achieve universal approximation power (Keriven & Peyré, 2019; Brüel-Gabrielsson, 2020; Maron et al., 2019), and under mild conditions and with enough parameters in the model (Oymak & Soltanolkotabi, 2019), a model with universal approximation power can achieve zero loss on the training set upon convergence, i.e., the property in Assumption 2, with high probability. This can be extended to the GNN case as long as there are no training vertexes with an identical neighbourhood and the same features but different labels. We assume the loss converges to zero for simplicity in our theoretical analysis, and show in our experiments in Sec. 4.3 that converging to a small loss (instead of zero) is sufficient for a GNN model to show the structural behaviour to be derived. Our first result below gives the locality in the predicting power of GNN around vertexes in the training set.

Proposition 1. *Let D be a given training set and θ_D be the set of parameters as defined in Assumption 2. Under Assumption 1, we have that for any $v \in D$ and $h_v = \mathcal{M}_{\theta_D}(v)$, there exists $r_v > 0$ such that for representations $h, h' \in \mathcal{H}$, if $d(h, h_v) < d(h', h_v) < r_v$, then $\mathcal{L}(f(h)) < \mathcal{L}(f(h'))$.*

We provide a proof for Proposition 1 in Appendix A. Proposition 1 states that given a GNN model with a large enough learning capacity (aka sufficient parameters in the model), for each vertex v in the training set, there exists a local neighbourhood of radius r_v surrounding representation h_v of v in the embedding space, on which the prediction performance (loss) admits simple monotonicity with respect to the distance between the representations. This result suggests that if we look at the landscape of the loss function from the perspective of the embedding space, the representation of each vertex in the training set induces a "valley" effect in the prediction performance (lowest loss at h_v within the neighbourhood), where losses incurred by representations nearby are larger if the respective distance to h_v is larger. Fig. 2 gives an illustration.

4.2 Structural relation between training set and GNN Performance

We next study the prediction power of GNN with respect to the (graph) distance between training set and test set. We first extend the result in Proposition 1 to study the prediction performance of GNN on different test sets.

Theorem 1. *Let $\mathcal{M}$ be a GNN model trained on vertex set D with distortion α and θ_D is the set of learned model parameters upon convergence. Let T and T' be two test sets whose vertexes are in the*

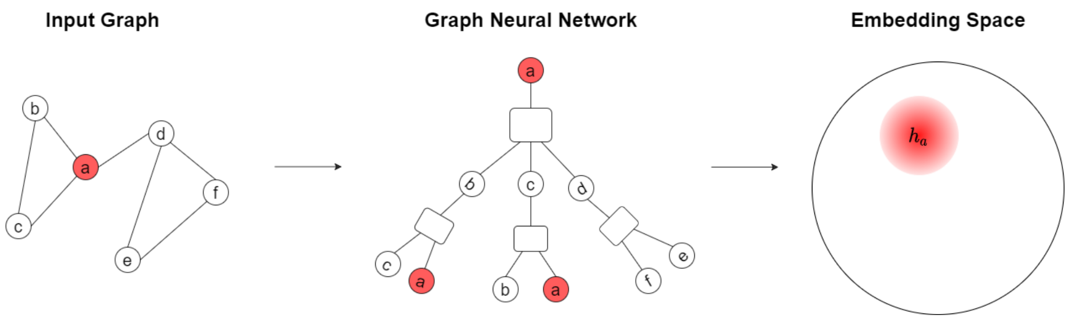

Figure 2: Locality of GNN performance induced by training vertexes. Vertex a is mapped to embedding h_a. Colored circle in embedding space represents a neighborhood centered at h_a. Color density represents the loss value: the darker, the lower the loss is (local minimal loss occurs at h_a).

local region of some vertexes in D as given in Proposition 1. There is a one-to-one mapping $g : T \mapsto T'$ with $d(u, D) < \frac{1}{\alpha} d(g(u), D), \forall u \in T$. Under Assumptions 1 and 2, there exists $\delta > 0$ such that if $\sum_{v \in T'} d(v, D) > \alpha\delta \sum_{u \in T} d(u, D)$, we have $\sum_{u \in T} \mathcal{L}(f(\mathcal{M}_{\theta_D}(u))) < \sum_{v \in T'} \mathcal{L}(f(\mathcal{M}_{\theta_D}(v)))$.

The proof of Theorem 1 is given in Appendix B. The core idea of the proof is to connect graph distance of vertexes with embedding distance and to extend the result of Proposition 1 to analyze the relation between the distance of different test sets to the training set and the performance (test loss) of the learned model. Theorem 1 gives a structural relation on the prediction power of GNN on different vertexes with respect to their graph distance to the training set. More specifically, GNN predicts better on closer vertexes. We proved this structural relation in the case that the distortion of the GNN is small enough so that relative order of distances across different test sets is preserved in the embedding space. We provide experimental validation on this premise. Relaxing the condition on the distortion rate and obtaining a complete relation renders an interesting future direction to investigate. A similar result was proved in (Ma et al., 2021) using the distance between training set and test set in the feature space. Here, we motivate our result from the perspectives of network homophily and topology awareness of GNN. Further, we can extend the result above to compare different pairs of training set and test set.

Theorem 2. *Let $\mathcal{M}$ be a GNN model trained on vertex sets D, D' with distortion α. Let $\theta_D, \theta_{D'}$ be the learned model parameters upon convergence, respectively. Let T and T' be two test sets whose vertexes are in the local region of some vertexes in D, D', respectively, as given in Proposition 1. There is a one-to-one mapping $g : T \mapsto T'$ with $d(u, D) < \frac{1}{\alpha} d(g(u), D'), \forall u \in T$. Under Assumptions 1 and 2, there exists $\delta' > 0$ such that if $\sum_{v \in T'} d(v, D') > \alpha\delta' \sum_{u \in T} d(u, D)$, we have $\sum_{u \in T} \mathcal{L}(f(\mathcal{M}_{\theta_D}(u))) < \sum_{v \in T'} \mathcal{L}(f(\mathcal{M}_{\theta_{D'}}(v)))$.*

The proof of Theorem 2 is given in Appendix C. Theorem 2 extends the result of Theorem 1 and gives a structural relation of the performance of GNN on different pairs of training set and test set with respect to their graph distances. We can draw several implications from the theorems. *First*, an insight is obtained that to enable better learning quality of GNN, the training set should consist of vertexes within close proximity to the rest of the input graph. Fig. 3 gives an illustration and we will further validate it in Sec. 4.3. *Second*, the topology awareness of GNN not only is important for improving its expressive power, but also can be exploited to make other informed decisions in the learning process. We use our result in investigating the initial data labelling problem in Sec. 5, and briefly discuss the potential of applying it in other related problems such as vertex sampling for batch training of GNN in Sec. 6. *Third*, Theorem 1 and Theorem 2 give us a glimpse on the fundamental difference between the learning process in GNN and in other ML settings.

4.3 Experiments

We next conduct experiments to verify our structural relation derived and some of the assumptions. We train four representative GNN models: (1) GCN (Kipf & Welling, 2017); (2) GAT (Veličković et al., 2017); (3) GraphSAGE (Hamilton et al., 2018); and (4) Position-aware Graph Neural Network (PGNN) (You et al., 2019) with a deliberately designed mechanism (described in Sec. 3.2) to enhance GNN's capability to capture inter-vertex distances. We adopt popular graph data sets including Cora, Citeseer and PubMed (Sen et al., 2008; Li et al., 2018). Due to space limit, we mainly show experimental results using Cora (and GCN) in the main body of the paper, and present

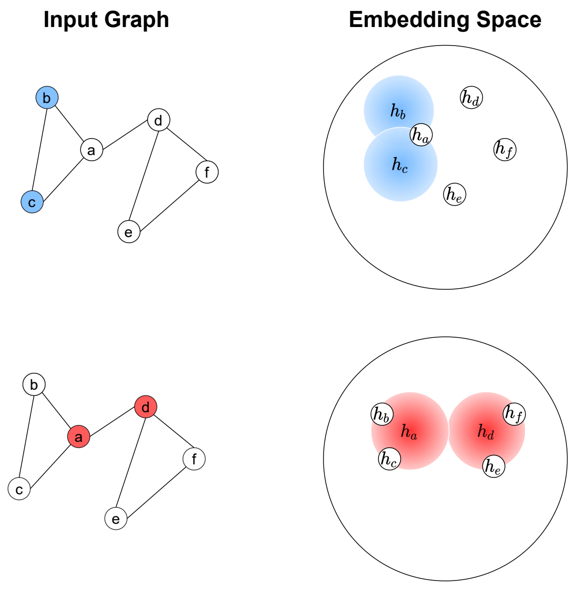

Figure 3: The loss landscape in the embedding space induced by different training sets. When vertexes a and d are selected as the training set instead of b and c, the test losses are smaller as the average distance between the training set and the other vertexes is smaller (the neighborhoods of h_a and h_d cover embeddings of more vertexes in the embedding space).

the remaining experimental results on other data sets, as well as GNN architecture details and training hyperparameters, in Appendix F. For experiments on Cora, we consider the largest connected component (2485 out of the total 2708 vertexes) and use a training set of size 35 (5 vertexes for each class) that can bring larger variance on the distance of training set to the rest of the vertexes. The experimental results demonstrate that our assumptions and the derived structural relation are indeed common and persistent among GNNs on data sets with network homophily.

We first validate that GNNs preserve distances (with a small distortion rate) locally by evaluating the relation between the graph distances of vertexes to the training set and the embedding distances of vertex representations to the representations of the training set, namely the relation between $d(v, D)$ and $d(\mathcal{M}_{\theta_D}(v), \mathcal{M}_{\theta_D}(D))$. In this experiment, we compute the graph distance and the embedding distance on the rest of the vertexes after training. As shown in Fig. 4, there exists a strong correlation between the graph distance and the embedding distance. Especially, when the graph distance is relatively small, the relation is close to linear and the relative orders are mostly preserved. This indicates that the distortion α is indeed small (at least within the first few hops) with different GNNs. This implies that the common aggregation mechanism in GNNs can well capture local graph structure information (graph distances), which in turn indicates that the premise on the distortion rate which we used to prove our theoretical results is not far from reality.

Next, we study the relation between the graph distance and the prediction loss following the same settings as in the previous experiment. As illustrated in Fig. 5, vertexes corresponding to smaller embedding distances (to representations of the training set in the embedding space) achieve smaller loss (the GNN is more certain and more likely to be correct with the prediction). This validates Proposition 1.

Combining the two experiments above, we can derive the positive relation between graph distance (to the training set) and prediction loss. A smaller graph distance (to the training set) translates into a smaller embedding distance (to representations of the training set) and then a smaller loss. Fig. 6 gives an illustration and validates the result in Theorem 1.

Further, we validate if the distance between different pairs of training set and test set is related to the prediction performance of GNN. As shown in Fig. 7, Table 1 and Table 2, this is indeed the case, which validates our result in Theoreom 2 that better performance (test accuracy/loss) is achieved when the distance between the training set and the test set is smaller.

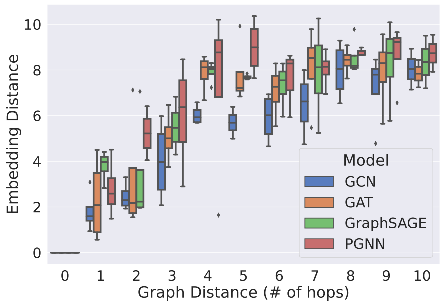

Figure 4: Graph distance vs. embedding distance. We randomly sample vertexes with different distances from the training set. We obtain their representations by feeding them into the trained GNN.

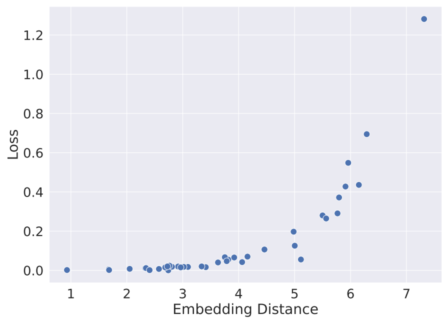

Figure 5: Embedding distance vs. loss: GCN on Cora data set. We randomly sample vertexes that are not in the training set, and derive their losses by feeding them into the trained GCN.

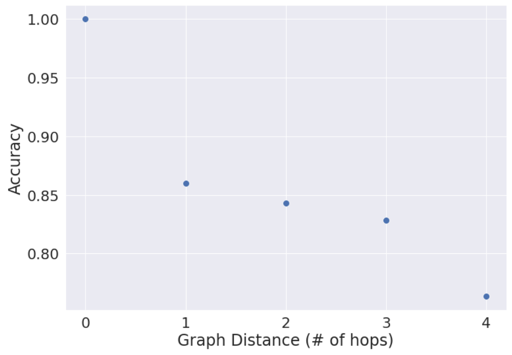

Figure 6: Graph distance vs. accuracy: GCN on Cora data set. We group vertexes that are not in the training set into different groups based on their distances to the training set. Then, we compute the average accuracy of the vertexes in different groups. Vertexes with 0 hop are the training vertexes.

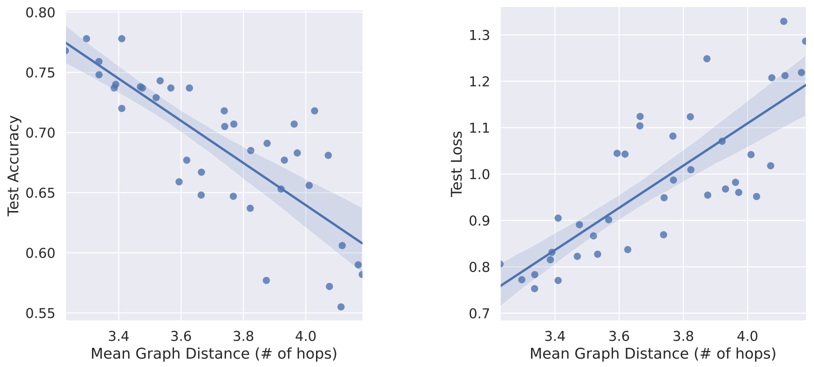

Figure 7: *Left:* test accuracy vs. average graph distance (to the training set). *Right:* test loss vs. average graph distance (to the training set). GCN on Cora. Mean graph distance is computed by $\frac{\sum_{u \in D_{test}} d(u, D)}{|D_{test}|}$.

Table 1: GCN on Cora. Same experimental setting as in the previous experiment. We randomly partition the labelled nodes into training set and test set for 100 times. Different partitions are ranked in increasing order of the distance between training set and test set. Metrics are evaluated in three cases: (1) "Random": all 100 ways of partitions are used; (2) Top 50 %: top half of the ranked partitions are used; (3) Top 30 %: top 30 % of the ranked partitions are used. Mean distance is the average graph distance between training set and test set. Values in the bracket indicate the variance.

	Test Accuracy	Test Loss	Mean Distance
Random	69.34 (5.03)	0.9629 (0.1326)	3.74 (0.18)
Top 50%	71.56 (4.23)	0.9343 (0.1229)	3.24 (0.07)
Top 30%	72.40 (3.59)	0.9152 (0.0970)	3.19 (0.06)

Table 2: Test accuracy of different models on Cora. Same experimental setting as in Table 1.

	GCN	GAT	GraphSAGE
Random	69.34 (5.03)	68.03 (4.62)	63.61 (5.44)
Top 50%	71.56 (4.23)	69.94 (3.86)	64.54 (4.84)
Top 30%	72.40 (3.59)	70.52 (3.53)	65.60 (4.27)

5 Application on Initial Data Labelling

We next present an application of our derived results on the initial data labelling problem in active learning of GNNs. Active learning deals with selecting samples in the data set to label and the goal is to maximize the gain on model performance while labelling the fewest samples possible. One common scheme of active learning is to first select an initial set D to label and have the model trained on D, and then select the data samples to be labelled next with "most uncertainty" (largest loss) using the trained model (Ren et al., 2020; Settles, 2009).

Unsatisfactory initial set selection may lead to slow learning progress (Grimova et al., 2018). How to select the most "economic/proper" initial labelled data set is referred to as the "cold start" problem, as the model itself is not ready to be used for selecting data samples yet. One natural objective for this selection is to find an initial set D from the complete data set V that could maximize the generalization performance of the model trained on D: $\arg\min_{D \subset V} \sum_{d \in V \setminus D} \mathcal{L}(d|D)$.

For a GNN model, we have shown that how well it can learn from the data set is closely related to the coverage of the training set. A smaller mean distance of other vertexes to the training set leads to better GNN performance. The distance function used in our theoretical results is defined with the class label of the vertexes, which is unavailable in the initial labelling problem. Nevertheless, the general principle is that we want the labelled data to have good coverage on the rest of the data. This can be formulated into the following optimization problem.

$$\begin{aligned} &\max_{D \subset V} \sum_{u \in V \setminus D} I_r(u|D) \\ &s.t. \quad |D| \leq k \end{aligned} \tag{1}$$

where k is the given initial set size and r is the given neighborhood radius. $I_r(u|D)$ is an indicator function: $I_r(u|D) = 1$ if there exists a vertex $v \in D$ such that $u \in N_r(v)$, and $I_r(u|D) = 0$, otherwise. The problem can be reduced to the maximal coverage problem (Hochbaum, 1996), as given in Appendix D. The maximal coverage problem is NP-hard. There exist efficient greedy algorithms to achieve an approximation ratio of $1 - \frac{1}{e}$, e.g., by selecting a vertex whose neighbor has the most uncovered vertexes at each stage (Hochbaum, 1996). We evaluate the initial set selected by solving (1) using the greedy algorithm in (Hochbaum, 1996), and those chosen with common approaches of uniformly random selection (effective for active learning of DNN under the I.I.D data assumption) and importance-based selection (vertexes are sampled based on some graph properties distribution). In Table 3, we observe that our locality-based selection achieves a significant improvement of model performance over the other strategies, giving a better kick-start to the active learning process of GNNs.

Table 3: Model accuracy of different models trained on Cora. An initial labelled set (k=10) is selected by different strategies, and test set includes the rest of labelled vertexes. We run each strategy 10 times on each model and compute average test accuracy and variance (in brackets). 'Cover' is our strategy that solves (1) with $r = 2$.

	GCN	GraphSAGE	GAT
Random	30.33 (1.45)	45.75 (1.73)	49.36 (0.52)
Importance:Degree	32.11 (1.01)	46.65 (1.23)	54.08 (0.24)
Importance:Random Walk	29.20 (0.72)	44.33 (1.18)	48.38 (0.19)
Importance:Centrality	33.27 (0.92)	47.01 (1.08)	56.29 (0.28)
Importance:Cluster Coefficient	28.11 (1.32)	43.83 (1.12)	49.30 (0.32)
Cover	<u>36.46 (0.56)</u>	<u>49.05 (0.88)</u>	<u>62.76 (0.18)</u>

In addition to finding an initial set of given size k, similar principle can be adopted to decide the minimal size of the initial labelled data set, for certain guarantee on the performance of the trained model. A lower bound of the initial labelled set size can be computed by solving the following problem:

$$\begin{aligned} &\min_{D \subset V} |D| \\ &s.t. \sum_{u \in V - D} I_r(u|D) \geq g|V| \end{aligned} \tag{2}$$

where $g \in [0, 1]$ and $I_r(u|D)$ is the same indicator function as defined earlier. For given r and g, problem (2) finds the minimum cover which covers at least g portion of all the vertexes within its radius-r neighborhoods. By adjusting the ratio g and radius r, one can control the performance guarantee resulted from the initial set selection. (2) can be reduced to a minimal partial covering problem which is NP-hard with efficient approximation algorithms available (Gandhi et al., 2004). We provide a detailed reduction in Appendix E.

6 Concluding Discussions

In this paper, we investigate the role of the input graph in GNN learning, and show that data dependence has significant effects on GNN learning quality. We formally and experimentally show that there exists a structural relation between the coverage of the training set and the performance of the GNN on the test set. Using the obtained result, we investigate its application in tackling the "cold start" problem in the active learning of GNN. As our first attempt into in-depth understanding of GNN's learning process, a number of interesting directions are opened up for further investigation.

Combining with other topological features and network principles. Besides the distance preserving property, the ideas in this work may be extended to combine other (useful) topology-preserving properties of GNN (e.g., group and community structures (Zhou et al., 2020)) and network principles (e.g., structure equivalence). Furthermore, similar ideas/analyses can also be extended to graph edge-related tasks by adapting the topological features from edges.

Size of the local neighborhood of the training set. It has been studied on other deep learning models that the curvature of these local neighborhoods (Keskar et al., 2016) is closely related to the performance of the deep learning models. Preliminary factors that affect the curvature of local minima have been studied in (Jastrzebski et al., 2018). Comparing with other deep learning models, GNNs allow for more rigorous analysis based on the underlying graph structure and the distance-preserving property. It is interesting to extend our analysis to formally unwrap the factors that affect the size/curvature of the local neighborhood, and how the curvature at the local minima could further affect the model performance.

Other applications of the results. Other than initial data labelling, we may apply our structural results in other data selection problems in GNN training, such as batch sampling for stochastic training (Chen et al., 2017a) and neighbor subsampling in the information aggregation process (Ying et al., 2018).

REFERENCES

Jean Bourgain. On lipschitz embedding of finite metric spaces in hilbert space. *Israel Journal of Mathematics*, 52(1-2):46–52, 1985.

Rickard Brüel-Gabrielsson. Universal function approximation on graphs, 2020.

Jianfei Chen, Jun Zhu, and Le Song. Stochastic training of graph convolutional networks with variance reduction. *arXiv preprint arXiv:1710.10568*, 2017a.

Jianfei Chen, Jun Zhu, and Le Song. Stochastic training of graph convolutional networks with variance reduction. *arXiv preprint arXiv:1710.10568*, 2017b.

Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: fast learning with graph convolutional networks via importance sampling. *arXiv preprint arXiv:1801.10247*, 2018.

David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. *arXiv preprint arXiv:1509.09292*, 2015.

Rajiv Gandhi, Samir Khuller, and Aravind Srinivasan. Approximation algorithms for partial covering problems. *Journal of Algorithms*, 53(1):55–84, 2004.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry, 2017.

Nela Grimova, Martin Macas, and Vaclav Gerla. Addressing the cold start problem in active learning approach used for semi-automated sleep stages classification. In *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pp. 2249–2253. IEEE, 2018.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs, 2018.

Dorit S Hochbaum. Approximating covering and packing problems: set cover, vertex cover, independent set, and related problems. In *Approximation algorithms for NP-hard problems*, pp. 94–143. 1996.

Stanislaw Jastrzebski, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. Three factors influencing minima in sgd, 2018.

Nicolas Keriven and Gabriel Peyré. Universal invariant and equivariant graph neural networks, 2019.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017.

AA Leman and B Weisfeiler. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsiya*, 2(9):12–16, 1968.

Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. Distance encoding: Design provably more powerful neural networks for graph representation learning, 2020.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI conference on artificial intelligence*, 2018.

Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*, 2017a.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks, 2017b.

Jiaqi Ma, Junwei Deng, and Qiaozhu Mei. Subgroup generalization and fairness of graph neural networks. *arXiv preprint arXiv:2106.15535*, 2021.

Haggai Maron, Ethan Fetaya, Nimrod Segol, and Yaron Lipman. On the universality of invariant networks. In *International conference on machine learning*, pp. 4363–4371. PMLR, 2019.

Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, 27(1):415–444, 2001.

Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks, 2020.

Sunil Nishad, Shubhangi Agarwal, Arnab Bhattacharya, and Sayan Ranu. Graphreach: Position-aware graph neural network using reachability estimations, 2021.

Samet Oymak and Mahdi Soltanolkotabi. Towards moderate overparameterization: global convergence guarantees for training shallow neural networks, 2019.

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. A survey of deep active learning, 2020.

Ryoma Sato. A survey on the expressive power of graph neural networks. *arXiv preprint arXiv:2003.04078*, 2020.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation, 2019.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Jul 2018. doi: 10.1145/3219819.3219890. URL `http://dx.doi.org/10.1145/3219819.3219890`.

Jiaxuan You, Rex Ying, and Jure Leskovec. Position-aware graph neural networks, 2019.

Kaixiong Zhou, Xiao Huang, Yuening Li, Daochen Zha, Rui Chen, and Xia Hu. Towards deeper graph neural networks with differentiable group normalization, 2020.

Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. *arXiv preprint arXiv:2006.11468*, 2020.

Qi Zhu, Natalia Ponomareva, Jiawei Han, and Bryan Perozzi. Shift-robust gnns: Overcoming the limitations of localized graph training data, 2021.

A PROOF OF PROPOSITION 1

In this appendix, we provide a proof for Proposition 1.

Proof. Let θ_D be the set of parameters learnt by the GNN model that satisfy properties given in Assumption 2. Namely, for each vertex $v \in D$ and any $\epsilon > 0$, we have that

$$\mathcal{L}_{\theta_D}(f(h_v)) < \epsilon,$$

where $h_v = \mathcal{M}(v)$. This means that $\mathcal{L}_{\theta_D}(f(h_v)) \mapsto 0$ and we know that the $\mathcal{L}$ function is a continuous function of range $\mathbb{R}_+$. $\mathcal{L}_{\theta_D}(f(h_v))$ achieves the global minimum of the loss function in the embedding space.

By Assumption 1, we know that $\frac{d}{dh}\mathcal{L}(f(h_v))$ and $\frac{d^2}{d^2h}\mathcal{L}(f(h_v))$ exist. This implies that

$$\frac{d}{dh}\mathcal{L}(f(h_v)) = 0,$$

as this is the necessary condition for h_v to achieve a local minimum. Furthermore, as $\mathcal{L}_{\theta_D}(f(h_v))$ also achieves the global minimum, this implies that

$$\frac{d^2}{d^2h}\mathcal{L}(f(h_v)) \geq 0.$$

This means that there must exist a $r_v > 0$ such that $\forall h \in N_{r_v}(h_v) \subset \mathcal{H}$, we have

$$\frac{d}{dh}\mathcal{L}(f(h_v)) \geq 0.$$

For $d(h, h_v) \leq d(h', h_v) \leq r_v$, we can rewrite $h' = h + d$. Then we have,

$$\begin{aligned}\mathcal{L}(f(h')) &= \mathcal{L}(f(h+d)) \\ &\geq \mathcal{L}(f(h)) + \frac{d}{dh}\mathcal{L}(f(h))\|d\|\end{aligned}$$

Because $\frac{d}{dh}\mathcal{L}(f(h)) \geq 0$ and $\|d\| > 0$, we have that

$$\mathcal{L}(f(h')) > \mathcal{L}(f(h)).$$

□

B PROOF OF THEOREM 1

Before diving into the detailed proof, we present an outline of the structure of the proof and prove a lemma which we use in the proof of the theorem.

Outline of the proof for the thereom

1. Suppose we are given T and T', two test sets which satisfy the premise of the theorem;
2. Then, we can approximate and bound the loss of each vertex in the test set based on the nearest vertex in the training set by extending the result from Proposition 1;
3. Based on smoothness of curvature assumed in Assumption 1, we can bound the behavior of vertexes in the test set and derive the desired result.

Lemma 1. *Let D be a given training set. $\mathcal{M}$ and f are the GNN model and prediction function which have parameter θ and satisfy the property in Assumption 2. Let u, v be two arbitrary vertexes in D with representations $h_u = \mathcal{M}(u)$ and $h_v = \mathcal{M}(v)$. Let $N_{r_v}(h_v)$ and $N_{r_u}(h_u)$ be the neighborhood given in Proposition 1. Furthermore, let $N_{r_v,+}(h_v) := \{h \in N_{r_v}(h_v) | \mathcal{L}(f(h)) > 0\}$, i.e.,* *the set whose elements have positive loss values.* $N_{r_u,+}(h_u)$ *is Similarly defined. Then, we have that*

$$N_{r_u,+}(h_u) \cap N_{r_v,+}(h_v) = \emptyset.$$

Proof. Suppose the opposite: there exists h which belongs to $N_{r_u,+}(h_u)$ and $N_{r_v,+}(h_v)$ at the same time. In other words,

$$h \in N_{r_v,+}(h_v) \cap N_{r_u,+}(h_u).$$

By their definition, $N_{r_v,+}(h_v)$ and $N_{r_u,+}(h_u)$ are open sets. By definition of an open set, there exist $r_{h,v}$ and $r_{h,u}$ such that the neighborhood $N_{r_{h,v}}(h) \subseteq N_{r_v}(h_v)$ and the neighborhood $N_{r_{h,u}}(h) \subseteq N_{r_u}(h_u)$

Let $r_h = \min\{r_{h,v}, r_{h,u}\}$. We have that neighborhood $N_{r_h}(h)$ are in both $N_{r_u}(h_u)$ and $N_{r_v}(h_v)$.

Consider $h' \in N_{r_h}(h)$ such that

$$d(h, h_v) < d(h', h_v),$$

and

$$d(h, h_u) > d(h', h_u).$$

Such h' exists because h_v and h_u are two distinct points in the embedding space. Next, let's consider the loss value of h, h' from the perspective of h_v. Because

$$d(h, h_v) < d(h', h_v),$$

by Proposition 1, we have that

$$\mathcal{L}(f(h)) < \mathcal{L}(f(h')).$$

Similarly, if consider the loss value of h, h' from the perspective of h_u, we have that

$$\mathcal{L}(f(h)) > \mathcal{L}(f(h')),$$

because

$$d(h, h_u) > d(h', h_u).$$

We reach a contradiction. ☐

Lemma 1 implies that each vertex can be in at most one of these neighborhoods at a time. Next, we provide the proof for Theorem 1.

Proof. Let $G = (V, E)$ be the input graph with node feature vector X_v for all $v \in V$. Let $D \subset V$ be the training sets. Let $\mathcal{M}$ be a given GNN model. Let f be the prediction function that maps the output of $\mathcal{M}$ to the class representation. Let $\mathcal{L}$ be the loss function with range $\mathbb{R}_+$. Let θ_D be the sets of model parameters learnt using training set D, which satisfy Assumption 2. In other words, we have,

$$\sum_{u \in D} \mathcal{L}(f(\mathcal{M}_{\theta_D}(u))) = 0 \tag{3}$$

Let T, T' be the two test sets satisfy the premise of the theorem. In other words, we have a one-to-one mapping $g : T \mapsto T'$ with $\alpha rd(u, D) < rd(g(u), D), \forall u \in T$. Now, let's consider the loss on the test set T with the model $\mathcal{M}_D$, which can be written as:

$$\sum_{u \in T} \mathcal{L}(f(\mathcal{M}_{\theta_D}(u))) \tag{4}$$

By premise, we have that for each $u \in T$, there exists at least one $v \in D$ such that $\mathcal{M}(u) \in N_{r_v}(\mathcal{M}(v))$ where $N_{r_v}(\mathcal{M}(v))$ satisfies properties of Proposition 1. In other words, the loss function is monotonically increasing with respect to the embedding distance in $N_{r_v}(\mathcal{M}(v))$.

By Lemma 1, we know that u can be in only one of this neighborhoods. Let $Q : T \mapsto D$ be the mapping that maps vertex $u \in T$ to its corresponding simple neighborhood that is centered at a vertex $Q(u) \in D$. For simplicity, we denote the vertex as $q_u = Q(u)$. Due to Proposition 1 and

Assumption 1, we can do a quadratic approximation of u around q_u and have the following upper bound:

$$\begin{aligned}\mathcal{L}(f(\mathcal{M}_{\theta_D}(u))) \leq \mathcal{L}(f(\mathcal{M}_{\theta_D}(q_u))) + \langle \mathcal{M}_{\theta_D}(u) - \mathcal{M}_{\theta_D}(q_u), D\mathcal{M}_{\theta_D}(q_u)\rangle \\ + \frac{M_u}{2}\|\mathcal{M}_{\theta_D}(u) - \mathcal{M}_{\theta_D}(q_u)\|\end{aligned} \tag{5}$$

where

$$M_u = \sup\{D^2\mathcal{L}(f(h)) | h \in N_{r_{q_u}}(q_u)\},$$

and

$$r_{q_u} = \|q_u - u\|.$$

Let

$$M_D^* = \max\{\|M_u\| | u \in T\} \tag{6}$$

M_D^* exists since T is finite. Then, we obtain the following upper bound that is universal for all vertexes in T:

$$\sum_{u\in T} \mathcal{L}(f(\mathcal{M}_{\theta_D}(u))) \leq \sum_{u\in T} L_1(u) + \frac{M_D^*}{2}\|\mathcal{M}_{\theta_u}(u) - \mathcal{M}_{\theta_D}(q_u)\| \tag{7}$$

where

$$L_1(u) = \mathcal{L}(f(\mathcal{M}_{\theta_D}(q_u))) + \langle \mathcal{M}_{\theta_u}(u) - \mathcal{M}_{\theta_D}(q_u), D\mathcal{M}_{\theta_D}(q_u)\rangle.$$

Similarly, let $Q' : T' \mapsto D$ be the mapping that maps vertex $u \in T'$ to its corresponding simple neighborhood that is centered at a vertex $Q'(u) \in D$. For simplicity, we denote the vertex as $q'_u = Q'(u)$. Again, due to Proposition 1 and Assumption 1, we can do a quadratic approximation of u around q'_u and have the following lower bound:

$$\begin{aligned}\mathcal{L}(f(\mathcal{M}_{\theta_D}(u))) \geq \mathcal{L}(f(\mathcal{M}_{\theta_D}(q'_u))) + \langle \mathcal{M}_{\theta_D}(u) - \mathcal{M}_{\theta_D}(q'_u), D\mathcal{M}_{\theta_D}(q'_u)\rangle \\ + \frac{L_u}{2}\|\mathcal{M}_{\theta_D}(u) - \mathcal{M}_{\theta_D}(q'_u)\|\end{aligned} \tag{8}$$

where

$$L_u = \inf\{D^2\mathcal{L}(f(h)) | h \in N_{r_{q'_u}}(q'_u)\},$$

and

$$r_{q'_u} = \|q'_u - u\|.$$

Let

$$L_D^* = \min\{\|L_u\| | u \in T'\} \tag{9}$$

L_D^* exists since T' is finite. Then, we obtain the following lower bound that is universal for all the vertexes in T':

$$\sum_{u\in T'} \mathcal{L}(f(\mathcal{M}_{\theta_D}(u))) \geq \sum_{u\in T'} L'_1(u) + \frac{L_D^*}{2}\|\mathcal{M}_{\theta_D}(u) - \mathcal{M}_{\theta_D}(q'_u)\| \tag{10}$$

where

$$L'_1(u) = \mathcal{L}(f(\mathcal{M}_{\theta_D}(q'_u))) + \langle \mathcal{M}_{\theta_D}(u) - \mathcal{M}_{\theta_D}(q'_u), D\mathcal{M}_{\theta_D}(q'_u)\rangle.$$

For simplicity, suppose the cardinality of T and T' are k, and let $I = \{1, ..., k\}$. Let's index and arrange the vertexes in T and T' such that,

$$T = \{u_1, ..., u_k\},$$

and

$$T' = \{v_1, ..., v_k\},$$

such that

$$v_i = g(u_i).$$

Let's consider the difference between equation 25 and equation 22:

$$\begin{aligned}
&\sum_{u\in T'} \mathcal{L}(f(\mathcal{M}_{\theta_D} u))) - \sum_{u\in T} \mathcal{L}(f(\mathcal{M}_{\theta_D}(u))) \\
&= \sum_{i\in I} \mathcal{L}(f(\mathcal{M}_{\theta_D}(v_i))) - \mathcal{L}(f(\mathcal{M}_{\theta_D}(u_i))) \\
&> \sum_{i\in I} L'_1(v_i) + \frac{L^*_D}{2}\|\mathcal{M}_{\theta_D}(v_i) - \mathcal{M}_{\theta_D}(q'_{v_i})\| - L_1(u_i) - \frac{M^*_D}{2}\|\mathcal{M}_{\theta_D}(u_i) - \mathcal{M}_{\theta_D}(q_{u_i})\| \\
&= \sum_{i\in I}(L'_1(v_i) - L_1(u_i)) + \sum_{i\in I}\Big[\frac{L^*_D}{2}\|\mathcal{M}_{\theta_D}(v_i) - \mathcal{M}_{\theta_D}(q'_{v_i})\| - \frac{M^*_D}{2}\|\mathcal{M}_{\theta_D}(u_i) - \mathcal{M}_{\theta_D}(q_{u_i})\|\Big] \\
&= \sum_{i\in I}(L'_1(v_i) - L_1(u_i)) + \\
&\frac{L^*_D}{2}\sum_{i\in I}\big[\|\mathcal{M}_{\theta_D}(v_i) - \mathcal{M}_{\theta_D}(q'_{v_i})\|\big] - \frac{M^*_D}{2}\sum_{i\in I}\big[\|\mathcal{M}_{\theta_D}(u_i) - \mathcal{M}_{\theta_D}(q_{u_i})\|\big]
\end{aligned} \tag{11}$$

Since q_{u_i} and q'_{v_i} are local minima of the loss function, we get that

$$L'_1(v_i) = L_1(u_i).$$

This leads to that

$$\begin{aligned}
&\sum_{i\in I} \mathcal{L}(f(\mathcal{M}_{\theta_D}(v_i))) - \mathcal{L}(f(\mathcal{M}_{\theta_D}(u_i))) \\
&> \frac{L^*_D}{2}\sum_{i\in I}\big[\|\mathcal{M}_{\theta_D}(v_i) - \mathcal{M}_{\theta_D}(q'_{v_i})\|\big] - \frac{M^*_D}{2}\sum_{i\in I}\big[\|\mathcal{M}_{\theta_D}(u_i) - \mathcal{M}_{\theta_D}(q_{u_i})\|\big]
\end{aligned} \tag{12}$$

To simplify the notation, let's denote

$$S = \sum_{i\in I}\big[\|\mathcal{M}_{\theta_D}(u_i) - \mathcal{M}_{\theta_D}(q_{u_i})\|\big],$$
$$S' = \sum_{i\in I}\big[\|\mathcal{M}_{\theta_D}(v_i) - \mathcal{M}_{\theta_D}(q'_{v_i})\|\big].$$

and

$$W = \sum_{i\in I} d(u_i, D),$$
$$W' = \sum_{i\in I} d(v_i, D)$$

By Definition 1, we have that

$$rW \le S \le \alpha r W,$$

and

$$rW' \le S' \le \alpha r W'.$$

Then, we can rewrite equation 27 as follows:

$$\begin{aligned}
&\sum_{i\in I} \mathcal{L}(f(\mathcal{M}_{\theta_D}(v_i))) - \mathcal{L}(f(\mathcal{M}_{\theta_D}(u_i))) \\
&> \frac{L^*_D}{2}S' - \frac{M^*_D}{2}S.
\end{aligned} \tag{13}$$

We know that $M_D^* > L_D^* > 0$. This means that there exists a constant $\beta > 1$ such that

$$M_D^* = \beta L_D^*.$$

Similarly, By the premise of the theorem, we have that,

$$S' \geq rW' > \alpha r W \geq S.$$

There exists a constant $\alpha > 1$ such that

$$S' = \gamma S.$$

Furthermore, we can obtain the following bound on the constant γ,

$$\begin{aligned} \gamma &= \frac{S'}{S} \\ &\geq \frac{rW'}{\alpha r W} \\ &\geq \frac{1}{\alpha}\frac{W'}{W} \\ &\geq \frac{1}{\alpha}\delta \end{aligned} \tag{14}$$

Substituting the above results into equation 28, we get that

$$\begin{aligned} &\frac{1}{2}\left[L_D^* S' - M_D^* S\right] \\ &= \frac{1}{2}\left[L_D^* \gamma S - \beta L_D^* S\right] \\ &= \frac{1}{2} L_D^* S(\gamma - \beta) \end{aligned} \tag{15}$$

Let's take any $\delta > \alpha\beta$ and have that

$$\begin{aligned} \gamma - \beta &= \frac{S'}{S} - \beta, \\ &> \frac{1}{\alpha}\delta - \beta, \\ &> \frac{1}{\alpha}\alpha\beta - \beta, \\ &= 0. \end{aligned} \tag{16}$$

Therefore, if $\delta > \alpha\beta$, then we have

$$\sum_{u \in V_{test}} \mathcal{L}(f(\mathcal{M}_{\theta_D}(u))) - \mathcal{L}(f(\mathcal{M}_{\theta_D}(u))) > 0 \tag{17}$$

This completes the proof for Theorem 1.

□

C Proof for Theorem 2

In this appendix, we provide a proof for Theorem 2. The proof is similar to the one presented for Theorem 1.

Proof. Let $G = (V, E)$ be the input graph with node feature vector X_v for all $v \in V$. Let $D, D' \subset V$ be the training sets. Let $\mathcal{M}$ be a given GNN model. Let f be the prediction function that maps the output of $\mathcal{M}$ to the class representation. Let $\mathcal{L}$ be the loss function with range $\mathbb{R}_+$. Let $\theta_D, \theta_{D'}$ be the sets of model parameters learnt using training set D, D', which satisfy Assumption 2. In other words, we have,

$$\sum_{u \in D} \mathcal{L}(f(\mathcal{M}_{\theta_D}(u))) = \sum_{v \in D'} \mathcal{L}(f(\mathcal{M}_{\theta_{D'}}(v))) = 0 \tag{18}$$

Let T, T' be the two test sets satisfy the premise of the theorem. In other words, we have a one-to-one mapping $g : T \mapsto T'$ with $\alpha rd(u, D) < rd(g(u), D'), \forall u \in T$. Now, let's consider the loss on the test set T with the model $\mathcal{M}_D$, which can be written as:

$$\sum_{u \in T} \mathcal{L}(f(\mathcal{M}_{\theta_D}(u))) \tag{19}$$

By premise, we have that for each $u \in T$, there exists at least one $v \in D$ such that $\mathcal{M}(u) \in N_{r_v}(\mathcal{M}(v))$ where $N_{r_v}(\mathcal{M}(v))$ satisfies properties of Proposition 1. In other words, the loss function is monotonically increasing with respect to the embedding distance in $N_{r_v}(\mathcal{M}(v))$.

By Lemma 1, we know that u can be in only one of this neighborhoods. Let $Q : T \mapsto D$ be the mapping that maps vertex $u \in T$ to its corresponding simple neighborhood that is centered at a vertex $Q(u) \in D$. For simplicity, we denote the vertex as $q_u = Q(u)$. Due to Proposition 1 and Assumption 1, we can do a quadratic approximation of u around q_u and have the following upper bound:

$$\begin{aligned}\mathcal{L}(f(\mathcal{M}_{\theta_D}(u))) \leq \mathcal{L}(f(\mathcal{M}_{\theta_D}(q_u))) &+ \langle \mathcal{M}_{\theta_D}(u) - \mathcal{M}_{\theta_D}(q_u), D\mathcal{M}_{\theta_D}(q_u) \rangle \\ &+ \frac{M_u}{2} \|\mathcal{M}_{\theta_D}(u) - \mathcal{M}_{\theta_D}(q_u)\|\end{aligned} \tag{20}$$

where
$$M_u = \sup\{D^2 \mathcal{L}(f(h)) | h \in N_{r_{q_u}}(q_u)\},$$
and
$$r_{q_u} = \|q_u - u\|.$$

Let
$$M_D^* = \max\{\|M_u\| | u \in T\} \tag{21}$$

M_D^* exists since T is finite. Then, we obtain the following upper bound that is universal for all vertexes in T:

$$\sum_{u \in T} \mathcal{L}(f(\mathcal{M}_{\theta_D}(u))) \leq \sum_{u \in T} L_1(u) + \frac{M_D^*}{2} \|\mathcal{M}_{\theta_u}(u) - \mathcal{M}_{\theta_D}(q_u)\| \tag{22}$$

where
$$L_1(u) = \mathcal{L}(f(\mathcal{M}_{\theta_D}(q_u))) + \langle \mathcal{M}_{\theta_u}(u) - \mathcal{M}_{\theta_D}(q_u), D\mathcal{M}_{\theta_D}(q_u) \rangle.$$

Similarly, let $Q' : T' \mapsto D'$ be the mapping that maps vertex $u \in T'$ to its corresponding simple neighborhood that is centered at a vertex $Q'(u) \in D'$. For simplicity, we denote the vertex as $q'_u = Q'(u)$. Again, due to Proposition 1 and Assumption 1, we can do a quadratic approximation of u around q'_u and have the following lower bound:

$$\begin{aligned}\mathcal{L}(f(\mathcal{M}_{\theta_{D'}}(u))) \geq \mathcal{L}(f(\mathcal{M}_{\theta_{D'}}(q'_u))) &+ \langle \mathcal{M}_{\theta_{D'}}(u) - \mathcal{M}_{\theta_{D'}}(q'_u), D\mathcal{M}_{\theta_{D'}}(q'_u) \rangle \\ &+ \frac{L_u}{2} \|\mathcal{M}_{\theta_{D'}}(u) - \mathcal{M}_{\theta_{D'}}(q'_u)\|\end{aligned} \tag{23}$$

where

$$L_u = \inf\{D^2\mathcal{L}(f(h)) | h \in N_{r_{q'_u}}(q'_u)\},$$

and

$$r_{q'_u} = \|q'_u - u\|.$$

Let

$$L^*_{D'} = \min\{\|L_u\| | u \in T'\} \tag{24}$$

$L^*_{D'}$ exists since T' is finite. Then, we obtain the following lower bound that is universal for all the vertexes in T':

$$\sum_{u \in T'} \mathcal{L}(f(\mathcal{M}_{\theta_{D'}}(u))) \geq \sum_{u \in T'} L'_1(u) + \frac{L^*_{D'}}{2} \|\mathcal{M}_{\theta_{D'}}(u) - \mathcal{M}_{\theta_{D'}}(q'_u)\| \tag{25}$$

where

$$L'_1(u) = \mathcal{L}(f(\mathcal{M}_{\theta_{D'}}(q'_u))) + \langle \mathcal{M}_{\theta_{D'}}(u) - \mathcal{M}_{\theta_{D'}}(q'_u), D\mathcal{M}_{\theta_{D'}}(q'_u)\rangle.$$

For simplicity, suppose the cardinality of T and T' are k, and let $I = \{1, ..., k\}$. Let's index and arrange the vertexes in T and T' such that,

$$T = \{u_1, ..., u_k\},$$

and

$$T' = \{v_1, ..., v_k\},$$

such that

$$v_i = g(u_i).$$

Let's consider the difference between equation 25 and equation 22:

$$\begin{aligned}
&\sum_{u \in T'} \mathcal{L}(f(\mathcal{M}_{\theta_{D'}} u))) - \sum_{u \in T} \mathcal{L}(f(\mathcal{M}_{\theta_D}(u))) \\
&= \sum_{i \in I} \mathcal{L}(f(\mathcal{M}_{\theta_{D'}}(v_i))) - \mathcal{L}(f(\mathcal{M}_{\theta_D}(u_i))) \\
&> \sum_{i \in I} L'_1(v_i) + \frac{L^*_{D'}}{2} \|\mathcal{M}_{\theta_{D'}}(v_i) - \mathcal{M}_{\theta_{D'}}(q'_{v_i})\| - L_1(u_i) - \frac{M^*_D}{2} \|\mathcal{M}_{\theta_D}(u_i) - \mathcal{M}_{\theta_D}(q_{u_i})\| \\
&= \sum_{i \in I} (L'_1(v_i) - L_1(u_i)) + \sum_{i \in I} \Big[\frac{L^*_{D'}}{2} \|\mathcal{M}_{\theta_{D'}}(v_i) - \mathcal{M}_{\theta_{D'}}(q'_{v_i})\| - \frac{M^*_D}{2} \|\mathcal{M}_{\theta_D}(u_i) - \mathcal{M}_{\theta_D}(q_{u_i})\|\Big] \\
&= \sum_{i \in I} (L'_1(v_i) - L_1(u_i)) + \\
&\frac{L^*_{D'}}{2} \sum_{i \in I} \big[\|\mathcal{M}_{\theta_{D'}}(v_i) - \mathcal{M}_{\theta_{D'}}(q'_{v_i})\|\big] - \frac{M^*_D}{2} \sum_{i \in I} \big[\|\mathcal{M}_{\theta_D}(u_i) - \mathcal{M}_{\theta_D}(q_{u_i})\|\big]
\end{aligned} \tag{26}$$

Since q_{u_i} and q'_{v_i} are local minima of the loss function, we get that

$$L'_1(v_i) = L_1(u_i).$$

This leads to that

$$\begin{aligned}
&\sum_{i \in I} \mathcal{L}(f(\mathcal{M}_{\theta_{D'}}(v_i))) - \mathcal{L}(f(\mathcal{M}_{\theta_D}(u_i))) \\
&> \frac{L^*_{D'}}{2} \sum_{i \in I} \big[\|\mathcal{M}_{\theta_{D'}}(v_i) - \mathcal{M}_{\theta_{D'}}(q'_{v_i})\|\big] - \frac{M^*_D}{2} \sum_{i \in I} \big[\|\mathcal{M}_{\theta_D}(u_i) - \mathcal{M}_{\theta_D}(q_{u_i})\|\big]
\end{aligned} \tag{27}$$

To simplify the notation, let's denote

$$S = \sum_{i \in I} \left[\| \mathcal{M}_{\theta_D}(u_i) - \mathcal{M}_{\theta_D}(q_{u_i}) \| \right],$$

$$S' = \sum_{i \in I} \left[\| \mathcal{M}_{\theta_{D'}}(v_i) - \mathcal{M}_{\theta_{D'}}(q'_{v_i}) \| \right].$$

and

$$W = \sum_{i \in I} d(u_i, D),$$

$$W' = \sum_{i \in I} d(v_i, D')$$

By Definition 1, we have that

$$rW \leq S \leq \alpha r W,$$

and

$$rW' \leq S' \leq \alpha r W'.$$

Then, we can rewrite equation 27 as follows:

$$\begin{aligned} &\sum_{i \in I} \mathcal{L}(f(\mathcal{M}_{\theta_{D'}}(v_i))) - \mathcal{L}(f(\mathcal{M}_{\theta_D}(u_i))) \\ &> \frac{L^*_{D'}}{2} S' - \frac{M^*_D}{2} S. \end{aligned} \tag{28}$$

Without loss of generality, we may assume $M^*_D > L^*_{D'} > 0$. Otherwise, the proof is completed. This means that there exists a constant $\beta > 1$ such that

$$M^*_D = \beta L^*_{D'}.$$

Similarly, By the premise of the theorem, we have that,

$$S' \geq rW' > \alpha r W \geq S.$$

There exists a constant $\alpha > 1$ such that

$$S' = \gamma S.$$

Furthermore, we can obtain the following bound on the constant γ,

$$\begin{aligned} \gamma &= \frac{S'}{S} \\ &\geq \frac{rW'}{\alpha r W} \\ &\geq \frac{1}{\alpha} \frac{W'}{W} \\ &\geq \frac{1}{\alpha} \delta \end{aligned} \tag{29}$$

Substituting the above results into equation 28, we get that

$$\begin{aligned} &\frac{1}{2} \left[L^*_{D'} S' - M^*_D S \right] \\ &= \frac{1}{2} \left[L^*_{D'} \gamma S - \beta L^*_D S \right] \\ &= \frac{1}{2} L^*_{D'} S (\gamma - \beta) \end{aligned} \tag{30}$$

Let's take any $\delta' > \alpha\beta$ and have that

$$\begin{aligned} \gamma - \beta &= \frac{S'}{S} - \beta, \\ &> \frac{1}{\alpha}\delta' - \beta, \\ &> \frac{1}{\alpha}\alpha\beta - \beta, \\ &= 0. \end{aligned} \tag{31}$$

Therefore, if $\delta' > \alpha\beta$, then we have

$$\sum_{u \in V_{test}} \mathcal{L}(f(\mathcal{M}_{\theta_D}(u))) - \mathcal{L}(f(\mathcal{M}_{\theta_D}(u))) > 0 \tag{32}$$

This completes the proof for Theorem 1.

□

D Maximum Cover

In this appendix we provide a detailed reduction of optimization (1) to the maximal coverage problem.

Let's create binary variables X_v and y_v for each vertex $v \in V$ such that X_v indicates whether vertex v is selected in the cover and y_v indicates whether vertex v is covered by some neighborhood of vertex in the selected set. We can rewrite optimization (1) as follows:

$$\begin{aligned} &\max \sum_{v \in V} y_v \\ s.t. \quad &\sum_{v \in V} X_v \leq k \\ &\sum_{u \in N_r(v)} X_u \geq y_v, \forall v \in V \\ &y_v \in \{0, 1\} \\ &X_v \in \{0, 1\} \end{aligned} \tag{33}$$

$\sum_{v \in V} X_v \leq k$ ensures no more than k vertexes are selected into the set. $\sum_{u \in N_r(v)} X_u \geq y_v, \forall v \in V$ ensures that y_v is 1 if and only if vertex v is covered by the neighborhood of some vertex u selected in the cover. The objective function is to maximize the number of vertexes covered. Optimization (33) has the standard form of maximal coverage problem as presented in (Hochbaum, 1996). Therefore, we can directly employ the heuristic algorithm and its performance guarantee in (Hochbaum, 1996).

E Partial Cover

In this appendix we provide a detailed reduction of optimization (2) to the partial coverage problem.

Let's create binary variables X_v and y_v for each vertex $v \in V$ suc that X_v indicates whether vertex v is selected in the cover and y_v indicates whether vertex v is covered by neighborhood of some vertex in the selected set. We can rewrite optimization (2) above as follows:

$$
\begin{aligned}
&\min \sum_{v\in V} X_v \\
s.t. &\sum_{u\in N_r(v)} X_u \geq y_v, \forall v \in V \\
&\sum_{v\in V} y_v \geq g|V| \\
&y_v \in \{0,1\} \\
&X_v \in \{0,1\}
\end{aligned} \tag{34}
$$

$\sum_{u\in N_r(v)} X_u \geq y_v, \forall v \in V$ ensures that y_v is 1 if and only if vertex v is covered by the neighborhood of some vertex u selected in the cover. $\sum_{v\in V} y_v \geq g|V|$ ensures that there are at least $g|V|$ number of vertexes covered. The objective function is to minimize the number of vertex selected in the cover set. Optimization (34) has the standard form of partial coverage problem as presented in (Gandhi et al., 2004) and therefore, we can directly employ the heuristic algorithm and its performance guarantee in (Gandhi et al., 2004).

F Additional Experiments

In this appendix, we provide additional experimental results and include detailed set-up of the experiments for reproducibility.

F.1 Random Partition

In the experiments related to Fig. 1, all three models, GCN, GAT and GraphSage, have 3 layers with hidden dimensions of 16 and Relu as the activation function. We use 2 attention heads for GAT. We trained each model with Adam optimizer with a learning rate of 0.01, weight decay of 0.0005, and drop-out of 0. We keep all other hyperparameters as default. For each training, we randomly select a k (30 for Citeseer and 60 for PubMed) number of labelled vertexes as the training set and use the rest of labelled vertexes to evaluate the performance. We trained each model for 400 epochs and record the performance. Additional experimental results on Citeseer and PubMed are provided in Fig. 8 and Fig. 9.

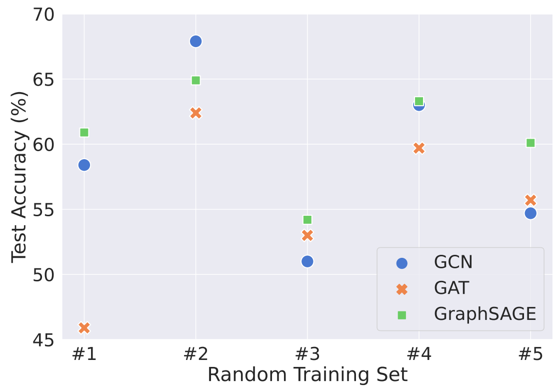

Figure 8: Performance of GCN, GraphSAGE and GAT on Citeseer dataset with different random partitions of training set and test set over the labelled data set.

Remark. From all these experiments, we can see that the variance on the performance of GNN induced by different training sets is indeed common and persistent across different data sets and GNN models.

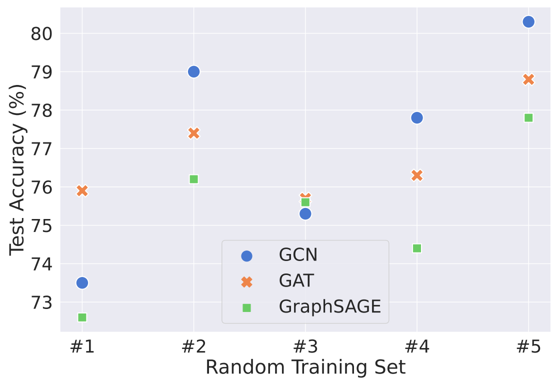

Figure 9: Performance of GCN, GraphSAGE and GAT on PubMed dataset with different random partitions of training set and test set over the labelled data set.

F.2 Graph Distance vs Embedding Distance

In the set of experiments related to Fig. 4, all three models, GCN, GAT and GraphSage, have 3 layers with hidden dimensions of 16 and Relu as the activation function. We trained each model with Adam optimizer with a learning rate of 0.01, weight decay of 0.0005, and drop-out of 0. We keep all other hyperparameters as default. For each training, we consider the largest connected component and randomly select k (30 for Citeseer and 60 for PubMed) number of labelled vertexes as the training set. We trained the model in 400 epochs. We randomly sample a set of vertexes from the graph that are not in the training set. We compute their graph/embedding distance (to the training set) and use them for the plots. The additional experimental results on Citeseer and PubMed datasets are provided in Fig. 10 and Fig. 11. In addition to the Euclidean distance metric, we also conducted similar experiments with the embedding distance computed by the cosine similarity. The results are provided in Fig. 12, Fig. 13 and Fig. 14.

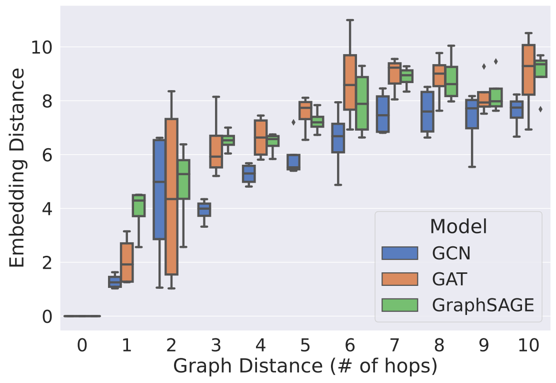

Figure 10: Graph distance vs. embedding distance on Citeseer. We randomly sample vertexes with different distances from the training set. We obtain their representations by feeding them into the trained GNN.

Remark. Comparing to Cora and Citeseer, we can see that the distance preserving phenomenon on PubMed is significantly weaker and this could be due to the fact that PubMed data set does not have a strong network homophily property as the other two data sets. However, as we will see in later experiments, it is still strong enough to works well with our initial data labelling strategy. In addition, the above experiments show that the relation between graph distance and embedding

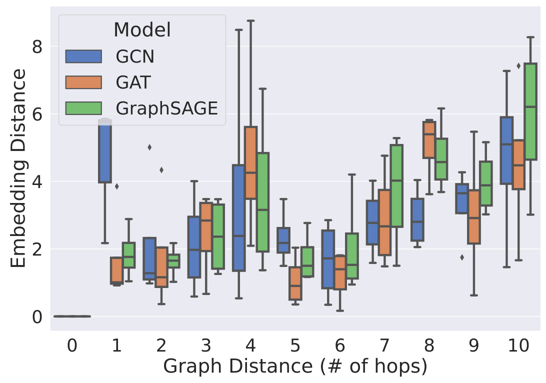

Figure 11: Graph distance vs. embedding distance on PubMed. We randomly sample vertexes with different distances from the training set. We obtain their representations by feeding them into the trained GNN.

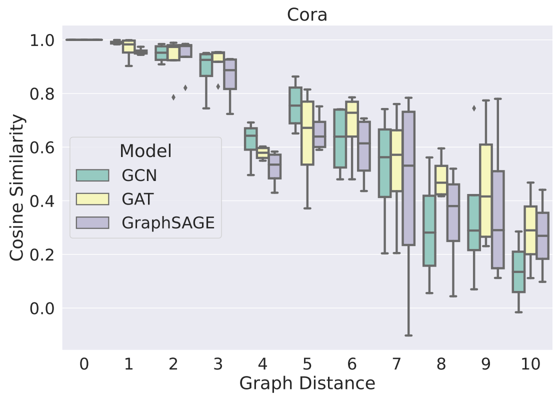

Figure 12: Graph distance vs. embedding distance on Cora with cosine similarity metric. We randomly sample vertexes with different distances from the training set. We obtain their representations by feeding them into the trained GNN.

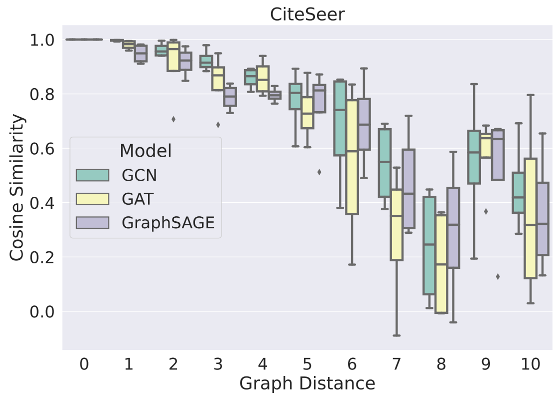

Figure 13: Graph distance vs. embedding distance on Citeseer with cosine similarity metric. We randomly sample vertexes with different distances from the training set. We obtain their representations by feeding them into the trained GNN.

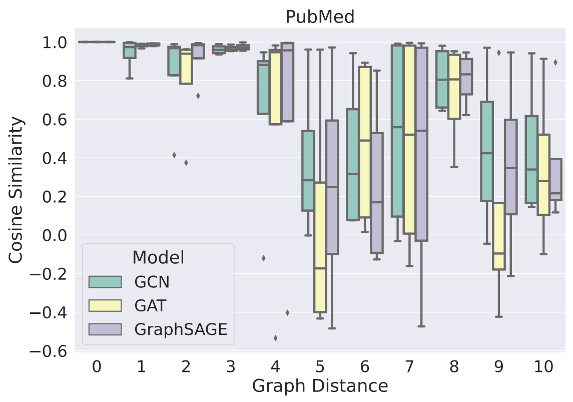

Figure 14: Graph distance vs. embedding distance on PubMed with cosine similarity metric. We randomly sample vertexes with different distances from the training set. We obtain their representations by feeding them into the trained GNN.

distance is persistent regardless of the metrics used for the computation (Euclidean and Cosine similarity).

F.3 Training Set Coverage and Model Performance

In this set of experiments related to Table 2, all three models, GCN, GAT and GraphSage, have 3 layers with hidden dimensions of 16 and Relu as the activation function. We use 2 attention heads for GAT. We trained the model with Adam optimizer with a learning rate of 0.01, weight decay of 0.0005 and drop-out of 0. We keep all other hyperparameters as default. For Citeseer data set, we consider the largest connected component and randomly partition the labelled nodes into training set (of size 30) and test set for 100 times. Different partitions are ranked in increasing order of the distance between training set and test set. Metrics are evaluated in three cases: (1) "Random": all 100 ways of partitions are used; (2) Top 50 %: top half of the ranked partitions are used; (3) Top 30 %: top 30 % of the ranked partitions are used. Mean distance is the average graph distance between training set and test set. For each training, we trained the model in 400 epochs and record the performance from each training. The additional experimental results on Citeseer are provided in Table 4.

Table 4: Test accuracy of different models on Citeseer

	GCN	GAT	GraphSAGE
Random	56.72 (5.41)	53.78 (4.73)	56.52 (4.85)
Top 50%	61.48 (5.15)	54.33 (4.24)	61.78 (3.60)
Top 30%	<u>63.03 (4.14)</u>	<u>54.63 (3.65)</u>	<u>62.38 (3.47)</u>

F.4 Initial Data Labelling

In this set of experiments related to Table 3, all three models, GCN, GAT and GraphSage, have 3 layers with hidden dimensions of 16 and Relu as the activation function. We use 2 attention heads for GAT. We trained the model with Adam optimizer with a learning rate of 0.01, weight decay of 0.0005 and drop-out of 0. We keep all other hyperparameters as default. For each training, we trained the model in 400 epochs and record the best performance from each training. For the "cover" strategy following the greedy algorithm in (Hochbaum, 1996), we greedily select the next vertexes to be the vertexes whose r-hop neighborhood has the most uncovered vertexes. If there are more than one candidates, we randomly pick one from the candidate set, and then we update and repeat the selection procedure. The additional experimental results on Citeseer and PubMed are provided in Table 5 and Table 6.

Table 5: Model accuracy (%) of different models trained on Citeseer data set. An initial labelled set (k=20) is selected by different strategies, and the test set includes the rest of lablled vertexes. We run each strategy 20 times on each model and compute its average performance and variance. 'Cover' is our strategy that solves (1) with $r = 2$.

	GCN	GraphSAGE	GAT
Random	31.27 (1.09)	33.34 (1.07)	45.14 (1.48)
Importance	38.84 (0.62)	36.72 (1.41)	49.27 (0.65)
Cover	42.58 (0.34)	48.93 (0.75)	64.13 (0.04)

Table 6: Model accuracy of different models trained on PubMed data set. An initial labelled set (k=20) is selected by different strategies, and the test set includes the rest of lablled vertexes. We run each strategy 20 times on each model and compute its average performance and variance. 'Cover' is our strategy that solves (1) with $r = 2$.

	GCN	GraphSAGE	GAT
Random	42.35 (0.84)	56.48 (0.87)	62.65 (0.55)
Importance	47.64 (0.73)	60.67 (0.54)	64.57 (0.19)
Cover	48.87 (0.43)	62.48 (0.41)	71.69 (0.01)

Remark. Based on the experimental results, GAT models significantly outperform (in terms of accuracy and variance) the other two models when the size of the training set is small. This makes an interesting research question: why GAT is able to generalize much better than GCN and GraphSAGE? This could be interesting to explore in the future.