ALIN: AN ACTIVE LEARNING FRAMEWORK FOR INCOMPLETE NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Significant progression has been made in active learning algorithms for graph networks in various tasks. However real-world applications frequently involve incomplete graphs with missing links, which pose the challenge that existing approaches might not adequately address. This paper presents an active learning approach tailored specifically for handling incomplete graphs, termed ALIN. Our algorithm employs graph neural networks (GNN) to generate node embeddings and calculates losses for both node classification and link prediction tasks. The losses are combined with appropriate weights and iteratively updating the GNN, ALIN efficiently queries nodes in batches, thereby achieving a balance between training feedbacks and resource utilization. Our empirical experiments have shown ALIN can surpass state-of-the-art baselines on Cora, Citeseer, Pubmed, and Coauthor-CS datasets.

Keywords— Active Learning, Incomplete Graphs, Graph Neural Networks, Node Classification, Link Prediction, Semi-Supervised Learning, Joint Loss Function

1 INTRODUCTION

The concept of graphs (or networks) has become pervasive across numerous domains, such as citation graphs and social graphs. Similar to other forms of data, graph data are undergoing rapid expansion, presently attaining substantial magnitudes. Consequently, the expanding dimensions of these graphs pose formidable challenges in attempting to analyze such type of data comprehensively.

Graph embeddings, the technique that transforms a given graph into a lower-dimensional space while preserving its underlying structural attributes and other inherent characteristics, are now gaining considerable attention in research areas (Goyal & Ferrara, 2018, Wang et al., 2023). By generating node embeddings, a spectrum of graph analytical tasks, including but not limited to node classification, node clustering, and link prediction, can be executed with heightened efficiency, optimizing both temporal and spatial considerations Ou et al. (2016). The semi-supervised graph embedding algorithms typically assume the training labeled data are given, which may not be always true in real practice Song et al. (2022). Given a labeling budget, the strategic selection of training labeled nodes to maximize eventual performance is thus of great importance. Addressing this concern, the concept of Active Learning (AL) has been introduced as a solution Xie et al. (2022). AL strategies offer a highly efficient mechanism for enhancing the process of data annotation by prioritizing the identification and labeling of the most informative instances. This, in turn, serves to optimize the efficiency and overall performance of machine learning models. Significantly, the domain of graph-based tasks, including many applications such as social network analysis, recommendation systems, and biological network inference, has benefited greatly from these developments (Deng, 2022, Vatter et al., 2023).

Recent AL-based approaches on graphs often assume the underlying network is fully known (Ma et al., 2022). However, this assumption tends to be overly simplistic as the underlying network cannot be fully observed in many real-world applications of network analyses (Valente & Pumpuang, 2007, Rice et al., 2012). While, in theory, it is conceivable to allocate additional resources towards the exhaustive exploration of the entire network, the endeavor to acquire a comprehensive network structure frequently proves to be prohibitively costly, demanding in terms of labor, or entirely unfeasible in practice (Valente & Pumpuang, 2007). For example, network data extracted from social media platforms bear privacy concern limitations as a substantial 52.6% of Facebook users took measures to conceal their friends' connections during a demographic analysis of Facebook in New York City in June 2011.[1] Consequently, when working with graph data, one should assume a more practical case that only a part of the network structure is available in practice (Hou et al., 2022, Teji et al., 2022,

[1]We refer to Dey et al. (2012) for the statistics.

Tran et al., 2021). This raises a critical challenge: How do we adapt AL methods to effectively operate on such incomplete graphs?

To tackle this pressing challenge, we introduce a new active learning framework explicitly tuned to handle incomplete networks: ALIN (**A**ctive **L**earning for **I**ncomplete **N**etworks). We propose a framework that incorporates an edge-based scoring mechanism into the AL framework. Conventionally, AL approaches in graphs have prioritized node-centric objectives, such as optimizing node classification accuracy, which is no longer sufficient and there is an inherent need to strategically select nodes that contribute to graph completeness. However, simply introducing edge scores can compromise the primary goal of node classification, leading to reduced overall accuracy. To strike a balance between enhancing graph completeness and preserving node classification accuracy, we introduce a two-phase training. In the initial epochs, we focus on link prediction as an auxiliary task. This early phase aims to establish an effective synergy between node scores and edge scores, facilitating the creation of informative edges within the incomplete graph. In the subsequent epochs, our approach seamlessly transitions towards prioritizing the core task of node classification, ensuring that the final objective is met with high accuracy. By combining the objectives of improving graph completeness and enhancing node classification accuracy, our proposed AL framework addresses the unique challenges posed by incomplete graphs. This innovative approach not only extends the applicability of AL techniques to real-world scenarios but also opens doors to more comprehensive and accurate graph-based data analysis.

In this paper, we present a comprehensive set of contributions, each addressing a distinct facet of the active learning problem in the context of incomplete graphs:

- We introduce the novel Active Learning on Incomplete Graphs (ALIN) framework that is meticulously designed to tackle the unique challenges posed by incomplete graph structures, offering a robust end-to-end solution.
- We extend the conventional node scoring approach by introducing edge scores. This innovation caters specifically to the optimization needs of incomplete graphs, allowing for more effective query node selection.
- We propose a novel joint loss function that seamlessly combines node classification and link prediction. This integration ensures that the interplay between these two critical components is optimized. Furthermore, we introduce a method to harmonize these two losses, thereby achieving superior results in the ultimate task of node classification.
- Our contributions are substantiated through an extensive series of experiments conducted on datasets. These experiments not only establish the superior performance of ALIN when compared to conventional active learning methods on benchmark graphs but also underscore the robustness of our approach across various datasets and with different GNN backbones.

2 Related work

The framework that we proposed is related to the following three research lines.

Active Learning. Traditional active learning algorithms operate by querying individual samples for labeling in a sequential manner. However, such an approach proves to be suboptimal when applied to deep learning models as it frequently retrains but updates little, and it is prone to overfitting Ren et al. (2021). Therefore, in deep active learning, the batch-mode setting, where a diverse set of instances are sampled and queried, is more often considered. In recent years, the optimal experimental design principle (Pukelsheim, 2006, Allen-Zhu et al., 2021) motivated the machine learning community to minimize the use of training resources and avoid tuning on a validation set. Combining the settings of one-shot learning and batch-mode active learning, several recent studies (Contardo et al., 2017, Wu et al., 2019) adopted a one-step batch-mode active learning setting.

Based on the query strategy, the majority of work can be divided into three categories Aggarwal et al. (2014): heterogeneity-based, performance-based, and representativeness-based. Heterogeneity-based Zhang et al. (2017) labeled the instances that are most different from the current known model. Performance-based Guo & Greiner (2007) minimize labeled uncertainty of the remaining unlabelled instance. Representativeness-based Li & Guo (2013) labeled the instance that can represent the underlying distribution of training instances

Active Learning on Graphs. The majority of work can be divided into four categories, including: EER, Heuristics, Uncertainty, and GraphPart. EER (Expected Error Reduction) (Xiaojin Zhu, 2003, Macskassy, 2009, Gu & Han, 2012) is a criterion in active learning that selects instances with the highest expected reduction in classification error, aiming to improve model performance efficiently. Heuristics (Macskassy, 2009, Hongyun Cai, 2017) are rule-of-thumb strategies used in active learning to guide the selection of informative data points for labeling, often based on measures like uncertainty, diversity, or disagreement among models. Uncertainty sampling (Ma & Yifei, 2013, Hongyun Cai, 2017, Wu et al., 2019, Ma et al., 2022) is an active learning method that selects instances for labeling based on the uncertainty of their predicted class probabilities, targeting instances where the model is least confident in its predictions. Recently, GraphPart (Ma et al.,

2022) first splits the graph into disjoint partitions and then selects representative nodes within each partition to query. It is worth noting that all prior work operated under the assumption of complete graphs, which diverges from reality given the incomplete nature characteristic of most real-world graphs. In Table 1, we summarize the aforementioned AL methods for the node classification task.

Experiment	Method				Incomplete Network	Adaptive
	EER	Heuristics	Uncertainty	GraphPart		
Xiaojin Zhu, 2003	x				No	No
Macskassy, 2009	x	x			No	Yes
Gu & Han, 2012	x				No	No
Ma & Yifei, 2013			x		No	No
Hongyun Cai, 2017		x	x		No	Yes
Wu et al., 2019			x		No	Yes
Ma et al., 2022			x	x	No	Yes
ALIN (ours)			x	x	Yes	Yes

Table 1: Summary of active learning techniques for node classification on graphs. Here, the Adaptive column indicates that the active learner is updated based on the newly labeled instances.

Link Prediction is a fundamental problem that attempts to estimate the likelihood of the existence of a link between two nodes Lu & Zhou (2010). This process enhances our comprehension of the connection between specific nodes and the evolution of the entire network. Link prediction has been widely applied to a variety of fields such as biology Lei & Ruan (2013) and social networks Liben-Nowell & Kleinberg (2007); Bonchi et al. (2011). A multitude of methodologies exist for the prediction of links within networks. Heemakshi Malhi (2016) provided an extensive survey that encompasses diverse link prediction algorithms, with a particular emphasis on scrutinizing the limitations inherent in such methods. Lei & Ruan (2013) presented an excellent survey by summarizing different approaches, introducing typical applications, and outlining future challenges of link prediction algorithms. Building upon this foundation, Martínez et al. (2016) furnished a more contemporary perspective by incorporating recent methodologies and conducting a meticulous comparative analysis of similarity-based techniques. Since it is difficult to identify a method that has the best performance in all complex networks, which strongly depends on the structural properties of the network, the authors in (Wu et al., 2022) categorized various link prediction strategies, including common neighbors-based, paths-based, probabilistic and statistical models based, classifier based, and network embedding based techniques.

3 Problem Formulation

In this section, we describe a formal definition of the problem of active learning on an incomplete graph under iterative batch-mode settings and introduce a uniform set of notations.

Let us denote an underlying network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with N nodes $v_i \in \mathcal{V}$, edges $(v_i, v_j) \in \mathcal{E}$, its node feature matrix $\mathbf{X} \in \mathbb{R}^{N \times F}$ (i.e., F dimensional feature vector for N nodes), label matrix for labeled nodes $\mathbf{Y} \in \mathbb{R}^{N \times C}$ (C is the number of node classes) (i.e., $\mathbf{Y}_{ij} = 1$ indicates node i has label j), an oracle to label a query node and its associated edges, and a labeling budget B.

In this study, we follow the iterative batch-mode setting Wu et al., 2019. In this setting, for each iteration, we deplete a predetermined resource budget to select a batch of nodes for labeling, streamlining the querying process to minimize redundant retraining. We entail segmenting a given budget B into K equitably sized partitions. During each iteration k, $b = [B/K]$ nodes are selected for querying, which form a set of selected nodes, denoted as $\mathcal{Q}_b^{(k)}$. The primary objective underlying this approach is to harness the informative feedback derived from the training process, while simultaneously safeguarding against excessive resource consumption. This contrasts with the fundamental AL setup, where only a solitary node is chosen at a time, potentially imposing considerable training overhead.

Since we tackle a setting where graph data is incomplete, for each iteration $k = \{0, \cdots, K\}$, we are given an incomplete graph $\tilde{\mathcal{G}}^{(k)} = (\mathcal{V}, \tilde{\mathcal{E}}^{(k)})$ and an incomplete label set $\tilde{\mathbf{Y}}^{(k)}$, where $\tilde{\mathcal{E}}^{(k)} \subset \mathcal{E}$ and $\tilde{\mathbf{Y}}^{(k)} \subset \mathbf{Y}$ are the edge set and the set of updated node labels at k-th iteration, respectively. At k-th iteration, by querying b nodes, we obtain $\tilde{\mathbf{Y}}_q^{(k)}$ and $\tilde{\mathcal{E}}_q^{(k)}$ as the sets of newly obtained node labels and edges after querying, respectively. Additionally, we denote $\tilde{\mathbf{Y}}_u^{(k)}$ and $\tilde{\mathcal{E}}_u^{(k)}$ as the sets of updated node labels and updated edges set at the k-th iteration. Thus, we have $\tilde{\mathbf{Y}}_u^{(k)} = \tilde{\mathbf{Y}}^{(k)} \cup \tilde{\mathbf{Y}}_q^{(k)}$ and $\tilde{\mathcal{E}}_u^{(k)} = \tilde{\mathcal{E}}^{(k)} \cup \tilde{\mathcal{E}}_q^{(k)}$; and the budget b is the maximum number of updated node labels. In this setting, we assume that the node feature matrix $\mathbf{X}$ is fully observable.

We aim to train a GNN-based classification model by iteratively updating its parameters $\theta^{(k)}$. The GNN model maps ($\tilde{\mathcal{E}}_u^{(k)}$, $\mathbf{X}$) to prediction vectors $\hat{\mathbf{Y}}^{(k)}$ and $\hat{\mathcal{E}}^{(k)}$. From the prediction and the observation, we compute node

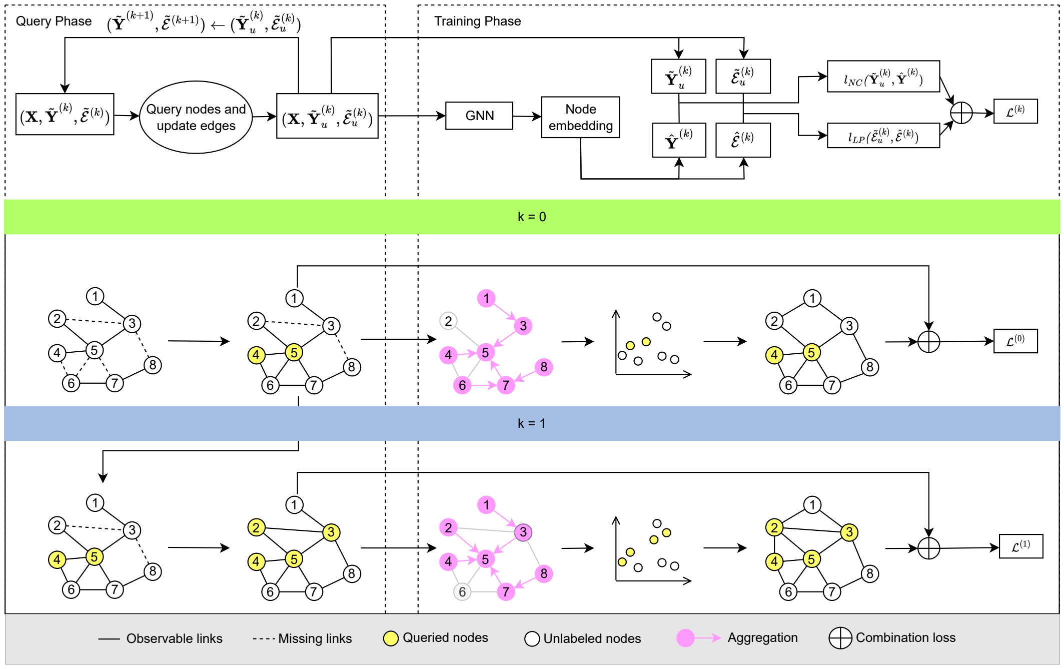
Query Phase $(\tilde{\mathbf{Y}}^{(k+1)}, \tilde{\mathcal{E}}^{(k+1)}) \leftarrow (\tilde{\mathbf{Y}}_u^{(k)}, \tilde{\mathcal{E}}_u^{(k)})$
Training Phase
$(\mathbf{X}, \tilde{\mathbf{Y}}^{(k)}, \tilde{\mathcal{E}}^{(k)})$
Query nodes and update edges
$(\mathbf{X}, \tilde{\mathbf{Y}}_u^{(k)}, \tilde{\mathcal{E}}_u^{(k)})$
GNN
Node embedding
$\tilde{\mathbf{Y}}_u^{(k)}$
$\tilde{\mathcal{E}}_u^{(k)}$
$\hat{\mathbf{Y}}^{(k)}$
$\hat{\mathcal{E}}^{(k)}$
$l_{NC}(\tilde{\mathbf{Y}}_u^{(k)}, \hat{\mathbf{Y}}^{(k)})$
$l_{LP}(\tilde{\mathcal{E}}_u^{(k)}, \hat{\mathcal{E}}^{(k)})$
$\mathcal{L}^{(k)}$
k = 0
$\mathcal{L}^{(0)}$
k = 1
$\mathcal{L}^{(1)}$
Observable links
Missing links
Queried nodes
Unlabeled nodes
Aggregation
Combination loss

Figure 1: A schematic overview of our proposed ALIN framework.

classification loss $l_{NC}(\tilde{\mathbf{Y}}_u^{(k)}, \hat{\mathbf{Y}}^{(k)})$ and link prediction loss $l_{LP}(\tilde{\mathcal{E}}_u^{(k)}, \hat{\mathcal{E}}^{(k)})$. To combine both losses, we sum l_{NC} and l_{LP} with a hyperparameter weight β, denoted as $\mathcal{L}^{(k)}$.

Our goal is to minimize the loss under a given budget b at the k-th iteration:

$$\min_{\theta^{(k)}, \mathcal{Q}_b} \mathcal{L}^{(k)} \tag{1}$$

4 ALIN Framework

In this section, we represent ALIN, a comprehensive solution designed to address the challenge of active learning within the context of an incomplete graph. The process underlying our framework is illustrated in Algorithm 1, which can be dissected into two principal components: the query phase and the training phase, both composed of distinct functions as follows:

- Query Phase: This phase encompasses node selection and subsequent updates, detailed in lines 3–15 of Algorithm 1. During the initial node selection, we utilize the InitNodes function (line 4). In subsequent iterations, we calculate node scores (line 8), edge scores (line 9), and combine them (line 10) to identify the most informative node (line 11). We then update the selected nodes and the lost edges associated with them (lines 13 –15).
- Training Phase: In this phase, we focus on the core of our methodology: a unified loss function that combines node classification and link loss prediction, as described in lines 16–19.

Example 1. A schematic overview of our proposed ALIN framework is visualized in Fig. 1, where we depict the initial two iterations of the ALIN framework. In first iteration, $(\mathbf{X}, \tilde{\mathbf{Y}}^{(0)}, \tilde{\mathcal{E}}^{(0)})$ contains eight unlabeled nodes and six missing edges. During the Query Phase, two nodes, namely 4 and 5 (highlighted by yellow circles), are chosen for labeling. Consequently, the three edges connected to these nodes are integrated, resulting in the updated $(\mathbf{X}, \tilde{\mathbf{Y}}_u^{(0)}, \tilde{\mathcal{E}}_u^{(0)})$. Subsequently, $(\mathbf{X}, \tilde{\mathbf{Y}}_u^{(0)}, \tilde{\mathcal{E}}_u^{(0)})$ undergoes the GNN model to generate node embeddings and to predict both $\hat{\mathbf{Y}}^{(0)}$ and $\hat{\mathcal{E}}^{(0)}$. From $(\tilde{\mathbf{Y}}_u^{(0)}, \tilde{\mathcal{E}}_u^{(0)})$ and $(\hat{\mathbf{Y}}^{(0)}, \hat{\mathcal{E}}^{(0)})$, we compute l_{NC} and l_{LP} and its amalgamation $\mathcal{L}^{(0)}$, which is subsequently utilized in the backpropagation process into the GNN. Moving on to the second iteration, we obtain $(\tilde{\mathbf{Y}}^{(1)}, \tilde{\mathcal{E}}^{(1)}) \leftarrow (\tilde{\mathbf{Y}}_u^{(0)}, \tilde{\mathcal{E}}_u^{(0)})$ and two additional nodes, namely 2 and 3 (highlighted by yellow circles), are selected for labeling. As a result, the two associated edges are incorporated, leading to the update of $(\mathbf{X}, \tilde{\mathbf{Y}}_u^{(1)}, \tilde{\mathcal{E}}_u^{(1)})$.

The technical details of the two phases are described in the following.

Algorithm 1 ALIN Algorithm

```
Require: $\mathbf{X}$, $\tilde{\mathcal{E}}^{(0)}$, $\tilde{\mathbf{Y}}^{(0)} = \emptyset$, Hyperparameters ($B$, $\alpha$, $\beta$), Trainable model parameters $\theta^{(0)}$, Training
    iterations $T$, iteration $k = 0$
Ensure: $\tilde{\mathcal{G}}^{(K)}$, $\theta^{(K)}$
 1: while $k < K$ do
 2:     $b \leftarrow [B/K]$
 3:     if $k = 0$ then                                                    ▷ Start Query Phase
 4:         $\mathcal{Q}_b^{(k)} \leftarrow$ InitNodes($\tilde{\mathcal{E}}^{(0)}, \mathbf{X}$)
 5:     else
 6:         $\tilde{\mathbf{Y}}^{(k)} \leftarrow \tilde{\mathbf{Y}}_u^{(k-1)}$
 7:         $\tilde{\mathcal{E}}^{(k)} \leftarrow \tilde{\mathcal{E}}_u^{(k-1)}$
 8:         $\phi_{NS}^{(k)} \leftarrow$ calculate Node Score for each node following (2)
 9:         $\phi_{ES}^{(k)} \leftarrow$ calculate Edge Score for each node following (3)
10:         $\phi_{CS}^{(k)} \leftarrow \alpha \cdot \phi_{NS}^{(k)} + (1 - \alpha) \cdot \phi_{ES}^{(k)}$
11:         $\mathcal{Q}_b^{(k)} \leftarrow$ Top $b$ nodes from $\phi_{CS}^{(k)}$
12:     end if
13:     From $\mathcal{Q}_b^{(k)}$ update selected nodes and associated edges from the oracle to $\tilde{\mathbf{Y}}_q^{(k)}$, $\tilde{\mathcal{E}}_q^{(k)}$
14:     $\tilde{\mathbf{Y}}_u^{(k)} \leftarrow \tilde{\mathbf{Y}}^{(k)} \cup \tilde{\mathbf{Y}}_q^{(k)}$
15:     $\tilde{\mathcal{E}}_u^{(k)} \leftarrow \tilde{\mathcal{E}}^{(k)} \cup \tilde{\mathcal{E}}_q^{(k)}$                            ▷ End Query Phase
16:     for $t = 1$ to $T$ do                                              ▷ Start Training Phase
17:         From $\theta^{(k)}$, calculate $\hat{\mathbf{Y}}^{(k)}$ and $\hat{\mathcal{E}}^{(k)}$
18:         $\mathcal{L}^{(k)} \leftarrow \beta \cdot l_{NC}(\tilde{\mathbf{Y}}_u^{(k)}, \hat{\mathbf{Y}}^{(k)}) + (1 - \beta) \cdot l_{LP}(\tilde{\mathcal{E}}_u^{(k)}, \hat{\mathcal{E}}^{(k)})$
19:         Backpropagation to $\theta^{(k)}$
20:     end for                                                            ▷ End Training Phase
21:     $k \leftarrow k + 1$
22: end while
```

4.1 Query Phase

4.1.1 InitNodes

As our framework relies on hidden representations of nodes or the predicted class distribution from the initial model, we operate within iterative settings, necessitating an initial model trained with the seed set. Consequently, we require the 'InitNodes' function to select the initial set of nodes. This function allows us to employ various selection strategies, such as random selection or employing recent methods such as FeatProp (Wu et al., 2019), Centrality (Hongyun Cai, 2017), GraphPart (Ma et al., 2022). For the implementation of ALIN, we employ GraphPart as the 'InitNodes' function. GraphPart works by dividing the graph into separate partitions and then selecting representative nodes within each partition for active learning with GNN. To mitigate interference across partitions in GraphPart, Ma et al., 2022 also introduce GraphPartFar, a method that penalizes selecting nodes close to medoids chosen in prior partitions, thereby promoting diversity among the returned nodes. According to two variants of GraphPart, we also present another variant of ALIN, termed ALINFar. The distinction between ALIN and ALINFar lies in their respective utilization of the 'InitNodes' function, i.e., ALIN employs the GraphPart function for this purpose, whereas ALINFar opts for the utilization of GraphPartFar as the initialization function.

4.1.2 Combine Score

In the context of querying the constituents of the graph, given a prescribed number of b queries, the query function is constructed based on an equilibrium criterion encompassing the informational value of nodes into node score $\phi_{NS}^{(k)}$ and the informational value of edges into edge score $\phi_{ES}^{(k)}$. The amalgamation of $\phi_{NS}^{(k)}$ and $\phi_{ES}^{(k)}$ with a weight parameter denoted as α yields the composite score $\phi_{CS}^{(k)}$.

Node Score. The use of entropy as a scoring metric provides valuable insights into the confidence of the GNN model's predictions for individual nodes. Higher entropy values indicate greater uncertainty, suggesting that a node's classification is less certain and may require further exploration or refinement in subsequent iterations of the process. In contrast, lower entropy values signify a higher level of confidence in the node's classification, making it less likely to be selected for additional query iterations. Following by Hongyun Cai (2017), ϕ_{NS} of candidate node v_i at the k-th iteration is calculated as follows:

$$\phi_{NS}^{(k)}(v_i) = -\sum_{c=1}^{C} \mathbb{M}_{ic}^{(k)} \log \mathbb{M}_{ic}^{(k)}, \tag{2}$$

where $\mathbb{M}_{ic}^{(k)} = \mathbb{P}(\tilde{\mathbf{Y}}_{ic}^{(k)} = 1|\tilde{\mathcal{G}}^{(k)}, \tilde{\mathbf{Y}}^{(k)}, \mathbf{X})$ is the probability of node v_i belonging to class c predicted by GNN at the k-th iteration; $\tilde{\mathbf{Y}}_{ic}^{(k)} = 1$ indicates node i has label c. Furthermore, the efficiency of our entropy-based node scoring approach plays a crucial role in accelerating the overall query process. With reduced computational overhead, our framework facilitates faster exploration of the graph and enhances the overall efficiency of the active learning framework.

Edge Score. In scenarios where the graph is incomplete, we aim to not only select nodes with high entropy but also nodes to allow the model to learn on a more complete graph. Intuitively, nodes with a larger difference in observable and predicted degrees are prioritized for inclusion in the active learning process since they offer the potential for improving the overall graph representation and classification performance. Thus, ϕ_{ES} of node v_i at the k-th iteration is calculated as follows:

$$\phi_{ES}^{(k)}(v_i) = \sum_{n=1}^{N} \mathbb{P}(\tilde{\mathcal{E}}_{in}^{(k)} = 1|\tilde{\mathcal{G}}^{(k)}, \tilde{\mathcal{E}}^{(k)}, \mathbf{X}) - D_{v_i}(\tilde{\mathcal{E}}^{(k)}), \tag{3}$$

where $\sum_{n=1}^{N} \mathbb{P}(\tilde{\mathcal{E}}_{in}^{(k)} = 1|\tilde{\mathcal{G}}^{(k)}, \tilde{\mathcal{E}}^{(k)}, \mathbf{X})$ is the probability of node v_i has connect to node v_n and $D_{v_i}(\tilde{\mathcal{E}}^{(k)})$ is the degree of node v_i in $\tilde{\mathcal{E}}^{(k)}$; $\tilde{\mathcal{E}}_{in}^{(k)} = 1$ indicates node i has a connection to node n at the k-th. Intuitively, $phi_{ES}^{(k)}(v_i)$ can be interpreted as the residual degree of a node v_i. By incorporating the $\phi_{ES}^{(k)}$ alongside the $\phi_{NS}^{(k)}$, our active learning framework ensures the selection of nodes that not only exhibit uncertainty but also contribute to enhancing the graph's completeness and discriminative power.

The combination of $\phi_{NS}^{(k)}$ and $\phi_{ES}^{(k)}$ represents a promising direction for active learning in graph-based settings. By leveraging uncertainty and graph completeness, our approach strikes a balance between exploration and exploitation, thereby achieving efficient and reliable active learning in real-world scenarios. The versatility of our approach makes it well-suited for a wide range of applications, including social networks, recommendation systems, and bioinformatics, among others

4.2 Training Phase

GNN model training: In our active learning framework, we utilize two loss functions during the training of the Graph Neural Network (GNN) model. For node classification, we employ the cross entropy loss function, and for link prediction, we utilize the binary cross entropy with logits loss function. The rationale behind using a combined loss is that both $\phi_{NS}^{(k)}$ and $\phi_{ES}^{(k)}$ are influenced by the predictive power of the GNN model. By simultaneously retraining both the node classifier and the edge classifier models, we ensure that the combined score captures valuable information about both node attribute information and the structural information surrounding each node. This concurrent retraining of the edge classifier is motivated by the objective to query nodes that offer a high value of information for both aspects, thereby enhancing the overall representation of the graph. During the model retraining phase, we update the GNN model based on the newly acquired information from the queried nodes. This iterative improvement of the GNN model helps refine the parameters $\theta^{(k)}$ with each query round, resulting in an increasingly accurate and informative model.

5 Experiments

5.1 Experiment Setup

Dataset. To compare against state-of-the-art methods, we experiment on 4 benchmark datasets, including citation networks Citeseer, Cora, and Pubmed (Sen et al., 2008) and co-authorship networks (Shchur et al., 2018). The summary statistics of the datasets are provided in Table 2. The homophily ratio is defined following (Zhu et al., 2020). In the experimental range with simulated conditions, we remove 30% of the total edges to create incomplete graphs.

GNN Models. We perform experiments over three popular GNN models, including a 3-layer GCN (Kipf & Welling, 2016) with hidden neurons are 128 and 64, respectively, a 3-layer GraphSAGE (Hamilton et al., 2017) with hidden neurons are 128 and 64, respectively, and a 8 attention head-GAT (Hamilton et al., 2017) with 2 hidden layers of size 16 and 8, respectively. To train each model, we use an Adam optimizer with an initial learning rate of 1×10^{-2} and weight decay of 5×10^{-4}. As in the active learning setup, there should not be

Dataset	#Nodes	#Edges	#Features	#Classes (C)	Homophily
Cora	2,708	5,278	1,433	7	0.810
Citeseer	3,327	4,552	3,703	6	0.736
Pubmed	19,717	44,324	500	3	0.802
Coauthor-CS	18,333	81,894	6,805	6	0.808

Table 2: Summary statistics of datasets.

enough labeled samples to be used as a validation set, we train the GNN model with fixed 200 epochs in all the experiments and evaluate over the full graph.

Competitive methods. We compare active learning methods that can be applied to the iterative setting, divided into two categories: 1) general-purpose methods that are agnostic to the graph structure, namely Random, Density, Uncertainty, and CoreSet; and 2) methods tailored for graph-structured data, including Centrality, AGE, FeatProp, GraphPartFar, ALINFar.

- **Random**: Randomly chooses nodes without any specific criteria.
- **Density** (Hongyun Cai, 2017): Initially applies clustering to the hidden representations of nodes. It then selects nodes with the highest density score, which is roughly inversely related to the l_2-distance between each node and its respective cluster center.
- **Uncertainty** (Settles & Craven, 2008): Selects nodes with the highest entropy in their predicted class distribution.
- **CoreSet** (Sener & Savarese, 2017): Utilizes K-Center clustering on the hidden representations of nodes. Given the scalability issues of the MIP optimized version, a time-efficient greedy approximation, as described in the original work, is used.
- **Centrality**: Chooses nodes with the highest values in graph centrality metrics. Notably, this approach only considers the graph structure and does not take into account node features. Empirical evidence from (Hongyun Cai, 2017) suggests that **Degree** centrality and **PageRank** centrality tend to outperform other metrics and thus we employ **Degree** and **PageRank** as two baselines for comparison.
- **AGE** (Hongyun Cai, 2017): Quantifies the informativeness of nodes by linearly combining three metrics: centrality, density, and uncertainty. It then selects nodes with the highest combined scores.
- **FeatProp** (Wu et al., 2019): First conducts K-Means clustering on the aggregated node features and subsequently selects nodes that are closest to the cluster centers.
- **GraphPart** and **GraphPartFar** (Ma et al., 2022): First obtains a K-partition of a graph using the Clauset-Newman-Moore greedy modularity maximization (Clauset et al., 2004) method. In each part, cluster on the aggregated node features and then choose the nodes closest to the cluster centers.

Following (Wu et al., 2019), we evaluate each baseline with a series of label budgets and report the Macro-F1 performance for node classification over the full graph. We note that the results are the average of repeated experiments with 3 random seeds

5.2 Experiment Results on GCN

The performance comparison between all competitive methods is presented in Table 3. Note worthy findings are summarized as follows:

- Our proposed ALIN and ALINFar, substantially outperform over baseline methods across various budget constraints. Notably, these improvements persist until performance plateaus.
- On smaller datasets like Cora and Citeseer, where the number of nodes and edges is relatively modest, our proposed framework exhibits remarkable superiority, surpassing baseline methods by a notable margin, typically around 1-1.5%. This enhanced performance can be attributed to the framework's adept utilization of feedback from the training process.
- For datasets with more extensive node and edge counts, such as Pubmed and Coauthor-CS, ALIN demonstrates clear advantages over baseline methods, particularly outperforming GraphPart, the state-of-the-art method. Notably, on larger datasets like Pubmed and Coauthor-CS, ALIN proves most effective with smaller budgets, typically around 200-230. However, with a more substantial budget (260), AGE edges slightly ahead of ALIN by approximately 0.2-0.5%, as larger budgets tend to lead to performance saturation.
- GraphPartFar exhibits commendable performance on Cora, trailing ALINFar by a mere 1-2%. However, as datasets expand in size, the loss of numerous edges affects partitioning significantly. Consequently, on Pubmed and Coauthor-CS, GraphPartFar lags behind ALINFar by approximately 4-5%.

Baselines	**Cora**			**Citeseer**		
Buget	200	230	260	200	230	260
Random	76.6 ± 0.8	78.6 ± 2.2	79.4 ± 0.5	61.8 ± 0.3	61.0 ± 0.4	63.1 ± 1.4
Density	73.1 ± 1.1	74.5 ± 1.6	76.4 ± 1.9	61.6 ± 0.6	60.4 ± 1.3	58.4 ± 1.4
Uncertainty	78.7 ± 0.7	$\underline{80.6}$ ± 0.9	80.5 ± 1.2	63.3 ± 1.1	$\underline{64.2}$ ± 1.3	64.5 ± 0.7
CoreSet	77.9 ± 1.8	79.1 ± 0.8	79.9 ± 0.2	61.2 ± 0.2	61.7 ± 1.4	65.8 ± 0.6
Degree	72.2 ± 0.4	73.5 ± 0.8	75.7 ± 1.2	53.9 ± 1.8	54.9 ± 1.1	56.4 ± 1.7
Pagerank	77.8 ± 0.5	78.5 ± 0.1	79.5 ± 0.6	63.1 ± 0.2	63.6 ± 0.1	64.1 ± 0.2
AGE	77.6 ± 0.9	77.7 ± 1.2	79.6 ± 0.2	63.2 ± 0.2	64.1 ± 0.9	66.0 ± 1.5
FeatProp	72.1 ± 1.9	73.1 ± 0.8	74.5 ± 0.6	50.6 ± 1.3	55.8 ± 0.5	57.2 ± 2.2
GraphPart	72.8 ± 1.8	73.7 ± 1.2	75.1 ± 0.6	54.1 ± 0.4	54.8 ± 1.1	57.2 ± 1.7
GraphPartFar	77.8 ± 0.3	78.4 ± 0.5	78.4 ± 0.6	60.9 ± 2.0	61.4 ± 2.1	62.3 ± 1.7
ALIN	**79.8** ± 1.3	80.4 ± 0.6	**81.6** ± 1.0	$\underline{63.4}$ ± 1.1	63.9 ± 1.2	$\underline{66.1}$ ± 0.9
ALINFar	$\underline{78.9}$ ± 1.2	**81.2** ± 1.1	$\underline{81.4}$ ± 1.7	**64.4** ± 1.3	**64.7** ± 1.1	**66.4** ± 0.6
Baselines	**Pubmed**			**Coauthor-CS**		
Buget	200	230	260	200	230	260
Random	77.1 ± 0.5	77.5 ± 1.3	78.8 ± 1.0	77.8 ± 0.4	82.6 ± 2.6	81.3 ± 3.1
Density	77.2 ± 0.4	76.9 ± 1.2	77.6 ± 0.9	79.3 ± 1.5	75.0 ± 0.8	79.3 ± 4.0
Uncertainty	77.3 ± 0.4	78.5 ± 1.5	79.0 ± 0.3	84.3 ± 1.0	85.4 ± 1.5	85.6 ± 0.4
CoreSet	76.4 ± 0.9	77.3 ± 0.8	77.1 ± 0.7	57.2 ± 3.1	62.7 ± 5.2	64.1 ± 3.8
Degree	76.1 ± 1.2	75.7 ± 0.5	75.7 ± 0.5	60.0 ± 0.2	59.9 ± 0.1	60.2 ± 0.5
Pagerank	75.8 ± 0.8	76.2 ± 0.2	77.5 ± 0.4	84.4 ± 0.7	84.0 ± 1.6	84.3 ± 1.0
AGE	$\underline{78.3}$ ± 1.0	$\underline{79.0}$ ± 0.2	$\underline{79.6}$ ± 0.6	$\underline{85.1}$ ± 0.3	$\underline{85.5}$ ± 1.1	**86.6** ± 0.3
FeatProp	74.0 ± 0.8	73.4 ± 0.7	73.6 ± 1.4	72.7 ± 4.7	77.7 ± 1.8	78.3 ± 1.0
GraphPart	73.8 ± 0.6	74.5 ± 0.8	74.2 ± 0.9	77.0 ± 3.6	79.0 ± 2.6	80.9 ± 2.4
GraphPartFar	75.1 ± 0.5	75.1 ± 0.6	74.8 ± 0.7	80.2 ± 1.7	84.9 ± 1.3	85.0 ± 0.8
ALIN	**79.0** ± 0.8	**79.3** ± 0.5	**80.1** ± 0.6	**85.4** ± 1.2	**86.1** ± 0.7	$\underline{85.8}$ ± 0.5
ALINFar	75.0 ± 1.6	77.7 ± 1.7	78.0 ± 0.9	84.3 ± 1.1	84.8 ± 0.3	85.6 ± 0.5

Table 3: Summary of the performance of GCN on each benchmark. The **bold** marker denotes the best performance and the $\underline{\text{underlined}}$ marker denotes the second-best performance.

- In the context of the second-best performing methods, both ALINFar and Uncertainty shine on Cora and Citeseer. However, on Pubmed and Coauthor-CS, Uncertainty's performance is lackluster, with AGE emerging as the second-best performer. The discrepancy arises because Uncertainty operates independently of the graph structure, which becomes problematic for larger datasets where edge loss entails more significant information loss. In contrast, AGE amalgamates centrality, density, and uncertainty scores, proving advantageous in scenarios where data lacks edges in extensive graphs. Consequently, AGE consistently outperforms Uncertainty in these settings.

Overall, our experimental results showcase the robustness and versatility of the ALIN framework and its extensions, shedding light on their adaptability across diverse graph datasets and budget constraints.

5.3 Generalizability to Other GNNs

We further present experiment results of all competitive methods on other GNN architectures in Table 4, in which the GCN backbone is replaced by GAT and GraphSAGE accordingly. Our proposed framework consistently demonstrates its capacity to enhance the accuracy of node classification tasks, even when transitioning GAT and GraphSAGE. This substantiates the framework's robust applicability across diverse GNN models.

Notably, among the benchmark methods including Random, Density, CoreSet, Pagerank, GraphPartFar, and ALINFar, CoreSet exhibits exceptional performance on GAT, surpassing GraphSAGE by approximately 4%. In contrast, the other baseline methods exhibit a narrower performance gap of around 1-2% when comparing GAT to GraphSAGE. This superiority of CoreSet on GAT underscores GAT's effectiveness in capturing pertinent information from neighboring nodes, even in scenarios with missing connections, courtesy of its adaptive attention mechanism that judiciously weighs the significance of various neighbors.

Conversely, on GraphSAGE, Uncertainty, Degree, and FeatProp consistently outperform GAT, achieving improvements of roughly 0.5-1%. Notably, GraphSAGE is favored for its computational efficiency and scalability, making it well-suited for large graph datasets. Its resilience to the absence of edges between nodes is attributed to its neighbor sampling strategy.

Baselines	**GAT**			**GraphSAGE**		
Buget	200	230	260	200	230	260
Random	78.4 ± 0.6	78.0 ± 1.1	79.5 ± 0.3	76.2 ± 0.5	78.2 ± 0.7	78.7 ± 0.4
Density	77.2 ± 0.9	80.1 ± 0.3	79.9 ± 0.4	77.4 ± 0.4	80.6 ± 1.5	81.7 ± 1.6
Uncertainty	$\underline{79.8} \pm 1.0$	81.3 ± 1.4	81.5 ± 1.0	$\underline{79.4} \pm 0.7$	$\underline{80.8} \pm 1.3$	81.3 ± 0.7
CoreSet	66.5 ± 1.4	65.7 ± 2.7	66.1 ± 3.7	61.0 ± 4.2	62.5 ± 2.4	62.1 ± 3.2
Degree	74.7 ± 1.1	75.3 ± 1.1	76.8 ± 0.4	76.4 ± 0.7	76.7 ± 0.6	79.1 ± 0.6
Pagerank	77.7 ± 0.5	78.9 ± 0.3	80.3 ± 0.6	77.6 ± 0.3	77.9 ± 0.5	80.1 ± 0.7
AGE	78.8 ± 0.1	79.9 ± 0.5	80.7 ± 0.1	77.3 ± 1.6	80.0 ± 0.4	80.8 ± 0.4
FeatProp	72.2 ± 0.7	73.8 ± 0.4	75.9 ± 0.4	73.1 ± 0.6	75.0 ± 0.8	76.1 ± 1.2
GraphPart	72.8 ± 0.9	74.3 ± 0.9	75.1 ± 0.3	74.4 ± 0.7	74.7 ± 1.0	75.0 ± 0.7
GraphPartFar	77.9 ± 0.4	76.9 ± 0.6	78.6 ± 0.8	76.7 ± 0.9	76.8 ± 0.5	77.6 ± 0.1
ALIN	$\mathbf{80.5} \pm 0.9$	$\underline{81.4} \pm 1.2$	$\underline{81.8} \pm 0.7$	79.0 ± 0.5	79.9 ± 1.8	$\underline{81.6} \pm 1.4$
ALINFar	79.7 ± 0.9	$\mathbf{82.1} \pm 0.2$	$\mathbf{82.7} \pm 0.8$	$\mathbf{79.9} \pm 0.8$	$\mathbf{81.7} \pm 0.3$	$\mathbf{82.4} \pm 0.4$

Table 4: Summary of the performance of others GNN on Cora dataset. The **bold** marker denotes the best performance and the underlined marker denotes the second-best performance.

It is noteworthy that both CoreSet and FeatProp rely on K-means clustering based on hidden node representations, which introduces sensitivity to different runs, particularly when applied to a limited number of labels. Consequently, these two baseline methods exhibit a considerably higher standard deviation compared to the other baselines.

5.4 Hyperparameter Sensitivity

We further the experiment results on tuning hyperparameters, and carry out experiments as follows in Fig. 2, we investigate the impact of hyperparameters α, which adjusts the balance between two terms for the query node selection at line 10 of the algorithm 1, on the performance of both ALIN and ALINFar when $B = 230$ and $K = 8$ on Cora dataset. In these experiments, we only show the results from the GCN model since those from the other GNN models follow a similar trend. We observe the following:

- The case of $\alpha = 0.5$ exhibits the best performance almost consistently.
- When $\alpha = 1$, ALIN and ALINFar only depend on the node score, resulting low performance.
- Setting α to 0.3 lead to a much lower performance as this setting overemphasizes the edge score, detracting from our ultimate goal of node classification.

These findings emphasize the delicate interplay between node uncertainty, edge information, and the overarching goal of accurate node classification within the ALIN framework.

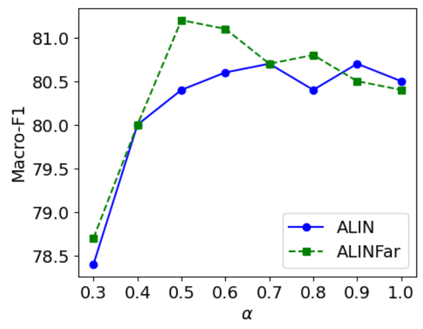

Figure 2: The expected influence by ALIN and ALINFar according to different values hyperparameters α

6 Conclusion and Discussion

In this work, we embarked on a comprehensive exploration of the active learning paradigm tailored specifically for Graph Neural Networks (GNNs) operating on incomplete graphs. Drawing inspiration from the synergy between node and edge information, we introduced a novel framework designed to harness the unique potential inherent in incomplete graph structures. Our experiments yielded compelling evidence of the efficacy of our proposed framework. We demonstrated that it not only outperforms existing state-of-the-art baseline active learning methods but does so consistently across a variety of real-world datasets and scenarios. These findings underscore the pivotal role of our approach in advancing active learning strategies in the context of incomplete graph data.

Several avenues for future research may include: 1) Investigating dynamic edge scoring mechanisms that adapt to the evolving graph structure could be fruitful and 2) Exploring the synergy between our active learning framework and graph generative models could open doors to novel applications.

REFERENCES

Charu C. Aggarwal, Xiangnan Kong, Quanquan Gu, Jiawei Han, and Philip S. Yu. Active learning: A survey. In *Data Classification: Algorithms and Applications*, 2014.

Zeyuan Allen-Zhu, Yuanzhi Li, Aarti Singh, and Yining Wang. Near-optimal discrete optimization for experimental design: A regret minimization approach. *Math. Program.*, 186(1–2):439–478, mar 2021. ISSN 0025-5610. doi: 10.1007/s10107-019-01464-2. URL `https://doi.org/10.1007/s10107-019-01464-2`.

Francesco Bonchi, Carlos Castillo, Aristides Gionis, and Alejandro Jaimes. Social network analysis and mining for business applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):1–37, 2011.

Gabriella Contardo, Ludovic Denoyer, and Thierry Artieres. A meta-learning approach to one-step active learning, 2017.

Yue Deng. Recommender systems based on graph embedding techniques: A review. *IEEE Access*, 10:51587–51633, 2022. doi: 10.1109/ACCESS.2022.3174197.

Ratan Dey, Zubin Jelveh, and Keith Ross. Facebook users have become much more private: A large-scale study. In *2012 IEEE International Conference on Pervasive Computing and Communications Workshops*, pp. 346–352. IEEE, 2012.

Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94, 2018. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2018.03.022. URL `https://www.sciencedirect.com/science/article/pii/S0950705118301540`.

Quanquan Gu and Jiawei Han. Towards active learning on graphs: An error bound minimization approach. *2012 IEEE 12th International Conference on Data Mining*, pp. 882–887, 2012.

Yuhong Guo and Russell Greiner. Optimistic active-learning using mutual information. In *IJCAI*, volume 7, pp. 823–829, 2007.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

MA Heemakshi Malhi. A survey of various link prediction algorithms in complex networks. *Int J Comput Sci Mob Comput*, 5(6):244–250, 2016.

Kevin Chen-Chuan Chang Hongyun Cai, Vincent W. Zheng. Active learning for graph embedding, 2017.

Yu Hou, Cong Tran, and Won-Yong Shin. Meta-code: Community detection via exploratory learning in topologically unknown networks. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pp. 4034–4038, 2022.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Chengwei Lei and Jianhua Ruan. A novel link prediction algorithm for reconstructing protein-protein interaction networks by topological similarity. *Bioinformatics*, 29 3:355–64, 2013.

Xin Li and Yuhong Guo. Active learning with multi-label svm classification. In *IjCAI*, volume 13, pp. 1479–1485. Citeseer, 2013.

David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. journal of the association for information science and technology (2007). *Google Scholar Google Scholar Digital Library Digital Library*, 2007.

Linyuan Lu and Tao Zhou. Link prediction in complex networks: A survey. *ArXiv*, abs/1010.0725, 2010.

Ma and Roman Yifei, Garnett. Sigma-optimality for active learning on gaussian random fields. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL `https://proceedings.neurips.cc/paper_files/paper/2013/file/7810ccd41bf26faaa2c4e1f20db70a71-Paper.pdf`.

Jiaqi Ma, Ziqiao Ma, Joyce Chai, and Qiaozhu Mei. Partition-based active learning for graph neural networks. *arXiv preprint arXiv:2201.09391*, 2022.

Sofus A. Macskassy. Using graph-based metrics with empirical risk minimization to speed up active learning on networked data. In *Knowledge Discovery and Data Mining*, 2009.

Víctor Martínez, Fernando Berzal, and Juan-Carlos Cubero. A survey of link prediction in complex networks. *ACM computing surveys (CSUR)*, 49(4):1–33, 2016.

Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1105–1114, 2016.

Friedrich Pukelsheim. *Optimal design of experiments*. SIAM, 2006.

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B Gupta, Xiaojiang Chen, and Xin Wang. A survey of deep active learning. *ACM computing surveys (CSUR)*, 54(9):1–40, 2021.

Eric Rice, Eve Tulbert, Julie Cederbaum, Anamika Barman Adhikari, and Norweeta G Milburn. Mobilizing homeless youth for hiv prevention: a social network analysis of the acceptability of a face-to-face and online social networking intervention. *Health education research*, 27(2):226–236, 2012.

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.

Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*, 2017.

Burr Settles and Mark Craven. An analysis of active learning strategies for sequence labeling tasks. In *proceedings of the 2008 conference on empirical methods in natural language processing*, pp. 1070–1079, 2008.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

Zixing Song, Xiangli Yang, Zenglin Xu, and Irwin King. Graph-based semi-supervised learning: A comprehensive review. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

Binon Teji, Jayanta K Das, Swarup Roy, and Dinabandhu Bhandari. Predicting missing links in gene regulatory networks using network embeddings: A qualitative assessment of selective embedding techniques. In *Intelligent Systems: Proceedings of ICMIB 2021*, pp. 143–154. Springer, 2022.

Cong Tran, Won-Yong Shin, and Andreas Spitz. Community detection in partially observable social networks. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 16(2):1–24, 2021.

Thomas W Valente and Patchareeya Pumpuang. Identifying opinion leaders to promote behavior change. *Health education & behavior*, 34(6):881–896, 2007.

Jana Vatter, Ruben Mayer, and Hans-Arno Jacobsen. The evolution of distributed systems for graph neural networks and their origin in graph processing and deep learning: A survey. *ACM Computing Surveys*, 2023.

Xiao Wang, Deyu Bo, Chuan Shi, Shaohua Fan, Yanfang Ye, and Philip S. Yu. A survey on heterogeneous graph embedding: Methods, techniques, applications and sources. *IEEE Transactions on Big Data*, 9(2): 415–436, 2023. doi: 10.1109/TBDATA.2022.3177455.

Haixia Wu, Chunyao Song, Yao Ge, and Tingjian Ge. Link prediction on complex networks: an experimental survey. *Data Science and Engineering*, 7(3):253–278, 2022.

Yuexin Wu, Yichong Xu, Aarti Singh, Yiming Yang, and Artur Dubrawski. Active learning for graph neural networks via node feature propagation. *arXiv preprint arXiv:1910.07567*, 2019.

Zoubin Ghahramani Xiaojin Zhu, John D. Lafferty. Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *International Conference on Machine Learning*, 2003.

Yu Xie, Shengze Lv, Yuhua Qian, Chao Wen, and Jiye Liang. Active and semi-supervised graph neural networks for graph classification. *IEEE Transactions on Big Data*, 8(4):920–932, 2022. doi: 10.1109/TBDATA.2021.3140205.

Ye Zhang, Matthew Lease, and Byron Wallace. Active discriminative text representation learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in neural information processing systems*, 33:7793–7804, 2020.

A WEIGHT GROWTH OF β

In this section, we delve into the intricate dynamics of β and its evolution across epochs, achieved through the utilization of a growth function. Recall that β is the weight of combined loss at line 18 of Algorithm 1, which plays a pivotal role in balancing the trade-off between two essential tasks: optimal link prediction and the ultimate goal of node classification.

At the outset of training, during the initial epochs, we set β to a value of 0.05. This choice steers the model's focus primarily towards solving the optimal link prediction problem. In contrast, as we approach the final epochs, our objective is to set β to 1, emphasizing the model's commitment to the ultimate task of node classification.

Achieving this gradual transition in β necessitates the implementation of a suitable growth function. We explored various weight growth functions to identify the most effective approach, weight growth functions shown in Fig. 3. The results of these experiments are summarized in Table 5, revealing distinct performance characteristics among different growth functions.

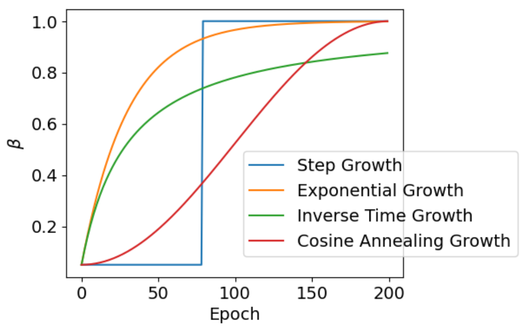

Figure 3: Weight growth functions

Notably, the Cosine Annealing, Step, and Exponential growth functions emerge as superior choices when compared to the Inverse Time growth function. Both the Cosine Annealing and Step growth functions exhibit an advantageous pattern of gradually increasing the β parameter during the middle epochs. This characteristic aligns seamlessly with the requirements of the Link Prediction task, which thrives on sustained training over multiple epochs, rather than experiencing a premature reduction in emphasis. Furthermore, the Exponential growth function proves notable for its ability to swiftly approach a β value close to 1 during the latter epochs. This rapid convergence to a higher β value positions the Exponential growth function as a compelling choice, outperforming the Cosine Annealing growth function in terms of accuracy.

In summary, our approach to the weight growth of β involves a thoughtful selection of growth functions, ultimately tailored to strike the right balance between optimizing link prediction and achieving robust node classification. The choice of growth function is a critical aspect of our framework, as it ensures that the model evolves and adapts its focus in a manner that aligns with the evolving requirements of the tasks at hand.

Weight Growth Function	Baselines	Budget		
		200	230	260
Step Growth	ALIN	**79.8** $\pm$ 1.3	<u>80.4</u> $\pm$ 0.6	**81.6** $\pm$ 1.0
	ALINFar	<u>78.9</u> $\pm$ 1.2	**81.2** $\pm$ 1.1	<u>81.4</u> $\pm$ 1.7
Inverse Time Growth	ALIN	63.3 $\pm$ 1.7	62.3 $\pm$ 2.3	65.2 $\pm$ 0.6
	ALINFar	63.0 $\pm$ 2.1	64.3 $\pm$ 1.7	65.6 $\pm$ 0.4
Exponential Growth	ALIN	78.2 $\pm$ 0.5	77.6 $\pm$ 1.5	79.7 $\pm$ 0.6
	ALINFar	78.6 $\pm$ 1.5	77.7 $\pm$ 1.1	80.1 $\pm$ 0.5
Cosine Annealing Growth	ALIN	76.0 $\pm$ 0.6	78.1 $\pm$ 0.8	76.8 $\pm$ 2.0
	ALINFar	74.7 $\pm$ 1.8	75.8 $\pm$ 0.4	76.7 $\pm$ 0.7

Table 5: Summary of the performance of weight growth functions using GNN on Cora dataset. The numerical values represent the average Macro-F1 score of 3 independent trials. The **bold** marker denotes the best performance and the <u>underlined</u> marker denotes the second-best performance.