# Accurate Link Prediction for Edge-Incomplete Graphs via PU Learning

**Junghun Kim**[1], **Ka Hyun Park**[1], **Hoyoung Yoon**[1], **U Kang**[1]

[1]Seoul National University, South Korea
{bandalg97, kahyunpark, crazy8597, ukang}@snu.ac.kr,

## Abstract

Given an edge-incomplete graph, how can we accurately find its missing links? The problem aims to discover the missing relations between entities when their relationships are represented as a graph. Edge-incomplete graphs are prevalent in real-world due to practical limitations, such as not checking all users when adding friends in a social network. Addressing the problem is crucial for various tasks, including recommending friends in social networks and finding references in citation networks. However, previous approaches rely heavily on the given edge-incomplete (observed) graph, making it challenging to consider the missing (unobserved) links.

In this paper, we propose PULL, an accurate link prediction method based on the positive-unlabeled (PU) learning. PULL treats the observed edges in the training graph as positive examples, and the unconnected node pairs as unlabeled ones. PULL prevents the model from blindly trusting the observed graph by proposing latent variables for unconnected node pairs, and leveraging the expected graph structure with respect to these variables. Extensive experiments on real-world datasets show that PULL consistently outperforms the baselines for predicting links in edge-incomplete graphs.

**Extended version** — https://github.com/snudatalab/PULL

## Introduction

*Given an edge-incomplete graph, how can we accurately find the missing links among the unconnected node pairs?* Edge-incomplete graphs are easily encountered in real-world networks. In social networks, connections between users can be missing since we do not check every user when adding friends. In the context of citation networks, there may be missing citations as we do not review all published papers for citation. In these scenarios, the objective is to find uncited references in citation networks (Shibata, Kajikawa, and Sakata 2012; Liu et al. 2019) or to recommend friends in social networks (Wang et al. 2015; Daud et al. 2020).

The main limitation of previous works (Zhang et al. 2021; Zhu et al. 2021; Chamberlain et al. 2023; Ucar 2023) for link prediction is that they assume the unconnected edges in the given graph as true negative examples. In link prediction, the complete set of true unconnected edges are not given. The given graph contains only a subset of ground-truth edges, while the other node pairs remain unlabeled; i.e., true unconnected edges and the unobserved ground-truth edges are mixed without labels. This misconception limits the model's ability to propagate information through unconnected node pairs, which may potentially form edges, resulting in overreliance of a link predictor to the edge-incomplete graph. Thus, it is important to consider the uncertainties of the given graph to obtain accurate linking probabilities.

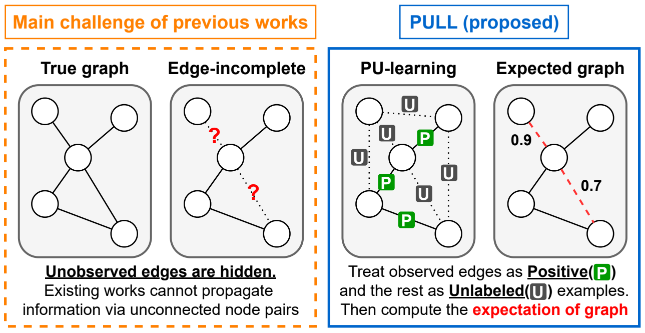
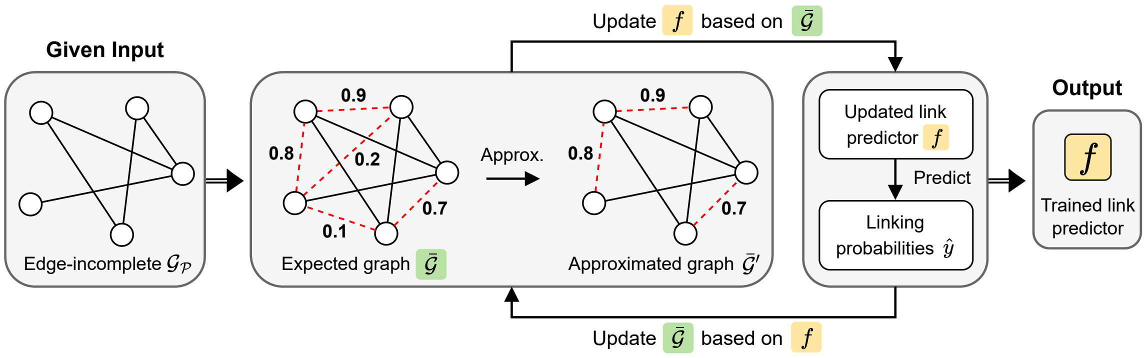

Main challenge of previous works
True graph
Edge-incomplete
?
?
Unobserved edges are hidden.
Existing works cannot propagate information via unconnected node pairs
PULL (proposed)
PU-learning
U
P
Expected graph
0.9
0.7
Treat observed edges as Positive(P) and the rest as Unlabeled(U) examples. Then compute the expectation of graph

Figure 1: Main challenge of previous works. They cannot consider the hidden unobserved edges in the given graph. PULL treats the unconnected node pairs as unlabeled examples, and utilizes the expectation of graph structure.

In this work, we propose PULL (PU-LEARNING-BASED LINK PREDICTOR), an accurate link prediction method in edge-incomplete graphs. To account for the uncertainties in the given graph while training a link predictor, PULL exploits PU (Positive-Unlabeled) learning (see Related Works for details). We treat the observed edges in the given graph as positive examples and the unconnected node pairs, which may contain hidden edges, as unlabeled examples. We then construct an expected graph while proposing latent variables for the unlabeled (unconnected) node pairs to consider the hidden edges among them. This enables us to effectively propagate information through the unconnected edges, improving the prediction accuracy. The main challenge of previous works and our approach is depicted in Figure 1.

Our contributions are summarized as follows:

- **Method.** We propose PULL, an accurate link prediction method in graphs. PULL effectively overcomes the pri-



Given Input
Edge-incomplete $\mathcal{G}_{\mathcal{P}}$
0.9
0.8
0.2
0.7
0.1
Expected graph $\bar{\mathcal{G}}$
Approx.
0.9
0.8
0.7
Approximated graph $\bar{\mathcal{G}}'$
Update $f$ based on $\bar{\mathcal{G}}$
Updated link predictor $f$
Predict
Linking probabilities $\hat{y}$
Update $\bar{\mathcal{G}}$ based on $f$
Output
$f$
Trained link predictor

Figure 2: Overall structure of PULL. Given an edge-incomplete graph $\mathcal{G}_{\mathcal{P}}$ with a set $\mathcal{P}$ of observed edges, PULL first computes the expected graph structure $\bar{\mathcal{G}}$ by proposing latent variables for the edges. Then PULL utilizes $\bar{\mathcal{G}}$ to update the link predictor $f$. The marginal linking probabilities $\hat{y}$ obtained by the updated $f$ are used to compute $\bar{\mathcal{G}}$ in the next iteration.

mary limitation of previous methods, specifically their heavy reliance on the provided edge-incomplete graph.

- **Theory.** We theoretically analyze PULL, including the relationship with the EM algorithm and the complexity.
- **Experiments.** Extensive experiments on real-world datasets show that PULL achieves the best performance.

## Related Works

### Link Prediction in Graphs

Many graph convolutional networks have been proposed to perform various tasks in graph-structured data including graph classification (Yoo, Shim, and Kang 2022), node feature estimation (Yoo et al. 2022a), graph generation (Jung, Park, and Kang 2020), and model compression (Kim, Jung, and Kang 2021). Recently, link prediction has garnered significant attention due to its successful application in various domains including social networks (Daud et al. 2020), recommendation systems (Afoudi, Lazaar, and Hmaidi 2023), and biological networks (Long et al. 2022). Previous link prediction approaches are categorized into two groups: autoencoder-based and embedding-based approaches.

Autoencoder methods use autoencoder structure (Hinton and Salakhutdinov 2006) for the link predictor. GAE and VGAE (2017) are the first autoencoder-based unsupervised framework for link prediction. ARGA and ARGVA (2018) exploit adversarial training strategy to improve the performance of GAE and VGAE, respectively. VGNAE (2021) finds that autoencoder-based methods produce embeddings that converge to zero for isolated nodes. They utilize L2-normalization to get better embeddings for these isolated nodes. MVGAE (2023) is a multi-view representation model which considers both global and local topologies. However, those autoencoder-based methods have limitations in that they cannot consider the missing edges during training.

Embedding-based approaches create compact representations of edges through information aggregation or subgraph extraction. GCN (2017), GraphSAGE (2017), and GAT (2017) aggregate information from neighbors to learn the edge embeddings, assuming adjacent nodes are similar. 1D-GNN (2021) formalizes link prediction as a conditional node classification, incorporating the identity of the source node. Neo-GNN (2022) propagates information through the original graph and concatenates structural features such as the count of common neighbors. SEAL (2018) and BUDDY (2023) extend the link prediction problem to a subgraph classification problem. PS2 (2023) automatically identifies optimal subgraphs for different edges. NESS (2023) utilizes static subgraphs during training and aggregates their representations at test time to learn embeddings in a transductive setting. However, the primary limitation of those methods is that they assume the edges of the given graph are fully observed. This misconception restricts the model's ability to propagate information between unconnected node pairs that might form edges, leading to over-reliance of the link predictor to the edge-incomplete graph.

### Graph-based PU Learning

The objective of PU (Positive-Unlabeled) learning is to train a binary classifier that effectively distinguishes positive and negative instances when only positive and unlabeled examples are available (Kiryo et al. 2017; Zhao et al. 2022). Recently, many graph-based PU learning approaches have been studied (Ma and Zhang 2017; Zhang et al. 2019; Wu et al. 2019). GRAB (2021; 2022b) is the first approach to solve the graph-based PU learning problem without knowing the class prior in advance. PULNS (2021) uses reinforcement learning to design an effective negative sample selector. PU-GNN (2023) introduces distance-aware PU loss to achieve more accurate supervision. However, those methods cannot be directly used in the link prediction problem since they aim to classify nodes, not edges, while considering the edges of the given graph as fully observed ones. PU-AUC (2021) and Bagging-PU (2022) proposed PU learning frameworks for link prediction considering the given edges as observed positive examples. However, their performance is constrained by the propagation of information through the edge-incomplete graph for obtaining node and edge representations.

## Proposed Method

We propose PULL, an accurate method for link prediction. We illustrate the overall process of PULL in Figure 2 and

Algorithm 1. The main challenges and our approaches are:

**C1. How can we consider the missing links?** We treat the given edges as observed positive examples, and the rest (unconnected edges) as unlabeled ones. We then propagate information through *expectation of graph structure* by proposing latent variables to the unconnected edges.

**C2. How can we effectively model the expected graph?** Naive computation of the expected graph is intractable since there are $2^{|\mathcal{E}_\mathcal{U}|}$ possible structures where $\mathcal{E}_\mathcal{U}$ is the set of unconnected edges. We compute the expectation of graph by carefully designing the probabilities of graphs.

**C3. How can we gradually improve the performance of the link predictor?** PULL iteratively improves the quality of the expected graph structure, which is the evidence for training the link predictor.

**Algorithm 1:** Overall process of PULL.

```
Input  : Edge-incomplete graph $\mathcal{G}_\mathcal{P} = (\mathcal{V}, \mathcal{E}_\mathcal{P})$, feature
         matrix $\mathbf{X}$, set $\mathcal{E}_\mathcal{U}$ of unconnected edges,
         hyperparameter $r$, and link predictor $f_\theta$
Output : Best parameters $\theta$ of the link predictor $f_\theta$
1 Randomly initialize $\theta^{\text{new}}$ , and initialize $K$ as $|\mathcal{E}_\mathcal{P}|$;
2 repeat
3     $\theta \leftarrow \theta^{\text{new}}$;
4     $\bar{\mathcal{G}} \leftarrow \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z}|\mathbf{X},\mathcal{E}_\mathcal{P},\theta)}[\mathbf{A}(\mathbf{z})] = \mathbf{A}^{\bar{\mathcal{G}}}$ ; // Equations (3, 4)
5     Approximate $\bar{\mathcal{G}}$ to $\bar{\mathcal{G}}'$ by selecting $K$ confident edges
        from a set of candidate edges;
6     $K \leftarrow K + |\mathcal{E}_\mathcal{P}| * r$ ;
7     $\theta^{\text{new}} \leftarrow \arg\min_\theta \mathcal{L}(\theta; \bar{\mathcal{G}}', \mathbf{X})$;        // Equations (5, 6)
8 until the maximum number of iterations is reached or the
    early stopping condition is met;
```

## Modeling Missing Links (C1)

In a link prediction problem, we are given a feature matrix $\mathbf{X}$ and an edge-incomplete graph $\mathcal{G}_\mathcal{P}$ consisting of two sets of edges, $\mathcal{E}_\mathcal{P}$ and $\mathcal{E}_\mathcal{U}$. The set $\mathcal{E}_\mathcal{P}$ contains observed edges, while $\mathcal{E}_\mathcal{U}$ consists of unconnected node pairs; $\mathcal{E}_\mathcal{P} \cup \mathcal{E}_\mathcal{U}$ is a set of all possible node pairs. Then we aim to find unobserved connected edges among $\mathcal{E}_\mathcal{U}$ accurately. Existing link prediction methods treat $\mathcal{E}_\mathcal{U}$ as true negative examples, which restricts the model's ability to propagate information through unconnected node pairs that could potentially form edges. This misleads $f$ into fitting the edge-incomplete graph, thereby degrading the prediction performance.

PULL addresses this misconception of previous methods by modeling the given graph based on PU-learning. Since there are hidden connections in $\mathcal{E}_\mathcal{U}$, we treat the unconnected edges in $\mathcal{E}_\mathcal{U}$ as unlabeled examples, and the observed edges in $\mathcal{E}_\mathcal{P}$ as positive ones. Then we propose a latent variable $z_{ij} \in \{1, 0\}$ for every edge $e_{ij}$, indicating whether there is a link between nodes $i$ and $j$ to consider the hidden connections; $z_{ij} = 1$ for every $e_{ij} \in \mathcal{E}_\mathcal{P}$, but not always $z_{ij} = 0$ for $e_{ij} \in \mathcal{E}_\mathcal{U}$. We denote the graph $\mathcal{G}_\mathcal{P}$ with latent variable $\mathbf{z} = \{z_{ij} \text{ for } e_{ij} \in (\mathcal{E}_\mathcal{P} \cup \mathcal{E}_\mathcal{U})\}$ as $\mathcal{G}_\mathcal{P}(\mathbf{z})$.

A main challenge is that we cannot propagate information through the *variablized graph* $\mathcal{G}_\mathcal{P}(\mathbf{z})$ since every edge $e_{ij} \in \mathcal{E}_\mathcal{U}$ of $\mathcal{G}_\mathcal{P}(\mathbf{z})$ is probabilistically connected. Instead, PULL exploits the expectation $\bar{\mathcal{G}}$ of graph $\mathcal{G}_\mathcal{P}(\mathbf{z})$ over the latent variables $\mathbf{z}$. This enables us to train a link predictor $f$ accurately, considering the hidden connections in $\mathcal{E}_\mathcal{U}$. Given that the estimated linking probabilities of $f$ provide prior knowledge for constructing the expected graph $\bar{\mathcal{G}}$, improved link predictor $f$ enhances the quality of $\bar{\mathcal{G}}$. Thus, PULL employs an iterative learning approach with two-steps to achieve a repeated improvement of the link predictor: a) constructing an expected graph $\bar{\mathcal{G}}$ based on the predicted linking probabilities from $f$, and b) updating $f$ utilizing $\bar{\mathcal{G}}$. The updated $f$ is used to refine $\bar{\mathcal{G}}$ in the subsequent iteration.

## Expectation of Graph Structure (C2)

During training of a link predictor $f_\theta$, PULL propagates information through the expected graph $\bar{\mathcal{G}}$ of $\mathcal{G}_\mathcal{P}(\mathbf{z})$ over the latent variable $p(\mathbf{z} \mid \mathbf{X}, \mathcal{E}_\mathcal{P}, \theta)$ where $\theta$ is the learnable model parameter. Note that $\bar{\mathcal{G}}$ requires computing the joint probabilities $p(\mathbf{z} \mid \mathbf{X}, \mathcal{E}_\mathcal{P}, \theta)$ for all possible graph structures $\mathcal{G}_\mathcal{P}(\mathbf{z})$. This is intractable since there are $2^{|\mathcal{E}_\mathcal{U}|}$ possible states of $\mathbf{z}$ in $\mathcal{G}_\mathcal{P}(\mathbf{z})$. Instead, PULL efficiently computes the expectation of graph structure by carefully designing the joint probability $p(\mathbf{z} \mid \mathbf{X}, \mathcal{E}_\mathcal{P}, \theta)$. In the following, we describe how we design the joint probability and compute the expectation of graph.

**Designing the joint probability.** We convert the graph $\mathcal{G}_\mathcal{P}(\mathbf{z})$ into a line graph $L(\mathcal{G}_\mathcal{P}(\mathbf{z})) = (\mathcal{V}_L, \mathcal{E}_L)$ where nodes in $L(\mathcal{G}_\mathcal{P}(\mathbf{z}))$ represent the edges of $\mathcal{G}_\mathcal{P}(\mathbf{z})$, and two nodes in $L(\mathcal{G}_\mathcal{P}(\mathbf{z}))$ are connected if their corresponding edges in $\mathcal{G}_\mathcal{P}(\mathbf{z})$ are adjacent. $\mathcal{V}_L$ contains both $\mathcal{E}_\mathcal{P}$ and $\mathcal{E}_\mathcal{U}$ of $\mathcal{G}_\mathcal{P}(\mathbf{z})$ since every node pair $(i, j)$ in $\mathcal{G}_\mathcal{P}(\mathbf{z})$ is correlated with variable $z_{ij}$. We then consider the line graph as a pairwise Markov network, which assumes that any two random variables in the network are conditionally independent of each other given the rest of the variables if they are not directly connected (Koller and Friedman 2009). This simplifies the probabilistic modeling on graph-structured random variables, and effectively marginalizes the joint distribution of nodes in $L(\mathcal{G}_\mathcal{P}(\mathbf{z}))$, which corresponds to the distribution $p(\mathbf{z} \mid \mathbf{X}, \mathcal{E}_\mathcal{P}, \theta)$ of edges in the original graph $\mathcal{G}_\mathcal{P}(\mathbf{z})$.

With the Markov property, the distribution $p(\mathbf{z} \mid \mathbf{X}, \mathcal{E}_\mathcal{P}, \theta)$ is computed by the multiplication of all the node and edge potentials in the line graph $L(\mathcal{G}_\mathcal{P}(\mathbf{z}))$:

$$p(\mathbf{z} \mid \mathbf{X}, \mathcal{E}_\mathcal{P}, \theta) = \frac{1}{F} \prod_{ij \in \mathcal{V}_L} \phi_{ij}(z_{ij} \mid \mathbf{X}, \theta) \prod_{(ij,jk) \in \mathcal{E}_L} \psi_{ij,jk}(z_{ij}, z_{jk} \mid \mathbf{X}, \theta) \quad (1)$$

where $\phi_{ij}$ is the node potential for a transformed node $ij$, and $\psi_{ij,jk}$ is the edge potential for a transformed edge $(ij, jk)$. The node potential $\phi_{ij}$ represents the unnormalized marginal linking probability between nodes $i$ and $j$ in the original graph $\mathcal{G}_\mathcal{P}(\mathbf{z})$. The edge potential $\psi_{ij,jk}$ denotes a degree of homophily between the edges containing a common node in $\mathcal{G}_\mathcal{P}(\mathbf{z})$. $F$ is the normalizing factor that ensures the distribution adds up to one. For simplicity, we omit $\mathbf{X}$ in $\phi_{ij}$ and $\psi_{ij,jk}$ in the rest of the paper.

We define the node potential $\phi_{ij}$ of $L(\mathcal{G}_\mathcal{P}(\mathbf{z}))$ as follows to make nodes in $\mathcal{G}_\mathcal{P}(\mathbf{z})$ with similar hidden representations have a higher likelihood of connection:

$$\phi_{ij}(z_{ij}=1 \mid \theta) = \begin{cases} 1 & \text{if } e_{ij} \in \mathcal{E}_{\mathcal{P}} \\ f_\theta(i,j) & \text{otherwise} \end{cases}$$

where $\phi_{ij}(z_{ij}=0 \mid \theta) = 1 - \phi_{ij}(z_{ij}=1 \mid \theta)$ and $f_\theta(i,j)$ is the predicted marginal linking probability between nodes $i$ and $j$. We set $\phi_{ij}(z_{ij}=1 \mid \theta) = 1$ for $e_{ij} \in \mathcal{E}_{\mathcal{P}}$ since the linking probability of an observed edge is 1. We use a GCN followed by a sigmoid function as the link predictor: $f_\theta(i,j) = \sigma(h_i \cdot h_j)$ where $h_i$ is the hidden representation of node $i$ computed by the GCN embedding function with graph $\bar{\mathcal{G}}$. In the first iteration, we initialize $\bar{\mathcal{G}}$ with $\mathcal{G}_{\mathcal{P}}$. Other graph-based models can also be used instead of GCN (see Q2 of Experiments section). We define $\psi_{ij,jk}$ as a constant $c$ to make the joint distribution focus on the marginal linking probabilities. Then the normalizing constant $F$ in Equation (1) becomes $c^{|\mathcal{E}_L|}$ as $\sum_{\mathbf{z}} \prod_{ij \in \mathcal{V}_L} \phi_{ij}(z_{ij} \mid \theta) = 1$ (see Lemma 1 in Appendix for proof).

As a result, the probability $p(\mathbf{z} \mid \mathbf{X}, \mathcal{E}_{\mathcal{P}}, \theta)$ is expressed by the multiplication of node potentials $\phi_{ij}$ for $e_{ij} \in \mathcal{E}_{\mathcal{U}}$:

$$\begin{aligned} p(\mathbf{z} \mid \mathbf{X}, \mathcal{E}_{\mathcal{P}}, \theta) &= \prod_{ij \in \mathcal{V}_L} \phi_{ij}(z_{ij} \mid \theta) \\ &= \prod_{e_{ij} \in (\mathcal{E}_{\mathcal{P}} \cup \mathcal{E}_{\mathcal{U}})} \phi_{ij}(z_{ij} \mid \theta) = \prod_{e_{ij} \in \mathcal{E}_{\mathcal{U}}} \phi_{ij}(z_{ij} \mid \theta). \end{aligned} \tag{2}$$

**Computing the expectation of graph.** Let $\mathbf{A}(\mathbf{z})$ be the adjacency matrix representing the state $\mathbf{z}$ where the $(i,j)$-th component of $\mathbf{A}(\mathbf{z})$, which we denote as $\mathbf{A}(\mathbf{z})_{ij}$, is $z_{ij} \in \{1, 0\}$. Then the corresponding weighted adjacency matrix $\mathbf{A}^{\bar{\mathcal{G}}}$ of the expected graph $\bar{\mathcal{G}}$ is computed as follows:

$$\begin{aligned} \mathbf{A}^{\bar{\mathcal{G}}} = \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z} \mid \mathcal{E}_{\mathcal{P}}, \theta)}[\mathbf{A}(\mathbf{z})] &= \sum_{\mathbf{z}} p(\mathbf{z} \mid \mathbf{X}, \mathcal{E}_{\mathcal{P}}, \theta) \mathbf{A}(\mathbf{z}) \\ &= \sum_{\mathbf{z}} \prod_{e_{ij} \in \mathcal{E}_{\mathcal{U}}} \phi_{ij}(z_{ij} \mid \theta) \mathbf{A}(\mathbf{z}). \end{aligned} \tag{3}$$

The $(i,j)$-th component $\mathbf{A}^{\bar{\mathcal{G}}}_{ij}$ of $\mathbf{A}^{\bar{\mathcal{G}}}$ is expressed as

$$\begin{aligned} &\phi_{ij}(z_{ij}=1 \mid \theta) \sum_{\mathbf{z} \mid z_{ij}=1} \prod_{e_{kl} \in \mathcal{E}_{\mathcal{U}} \setminus \{e_{ij}\}} \phi_{kl}(z_{kl} \mid \theta) \mathbf{A}(\mathbf{z})_{ij} \\ &= \phi_{ij}(z_{ij}=1 \mid \theta) \end{aligned} \tag{4}$$

where the equality holds since $\mathbf{A}(\mathbf{z})_{ij} = 1$ for $z_{ij} = 1$, and $\sum_{\mathbf{z} \mid z_{ij}=1} \prod_{e_{kl} \in \mathcal{E}_{\mathcal{U}} \setminus \{e_{ij}\}} \phi_{kl}(z_{kl} \mid \theta) = 1$ (see Lemma 1 in Appendix for proof). As a result, we simply express the expected graph $\bar{\mathcal{G}}$ by an weighted adjacency matrix $\mathbf{A}^{\bar{\mathcal{G}}}$ where $\mathbf{A}^{\bar{\mathcal{G}}}_{ij} = \phi_{ij}(z_{ij}=1 \mid \theta)$.

**Approximating the expected graph.** Propagating information through $\bar{\mathcal{G}}$ directly may lead to oversmoothing problem, as $\bar{\mathcal{G}}$ is a fully connected graph represented by $\mathbf{A}^{\bar{\mathcal{G}}}$. Moreover, the training time increases exponentially with the number of nodes. To address these challenges, PULL utilizes an approximated one of $\bar{\mathcal{G}}$, which contains edges with high confidence. Specifically, we keep the top-$K$ edges with the largest weights while removing the rest. We denote this approximated one as $\bar{\mathcal{G}}'$, and its adjacency matrix as $\mathbf{A}^{\bar{\mathcal{G}}'}$. From the perspective of PU learning, selecting edges in $\bar{\mathcal{G}}$ can be viewed as selecting relatively connected edges among the unlabeled ones, while treating the rest as relatively unconnected edges. We gradually increase the number $K$ of selected edges in proportion to that of observed edges through the outer iteration of Algorithm 1, which is expressed by $K \leftarrow K + r|\mathcal{E}_{\mathcal{P}}|$, giving more trust in the expected graph $\bar{\mathcal{G}}$. This is because the quality of $\bar{\mathcal{G}}$ improves through the iterations (Figure 3). We set $r = 0.05$ in our experiments.

Another challenge lies in the need to compute weights for every node pair in each outer iteration to acquire the top-$K$ edges, which results in computational inefficiency. To address this, we define a set of candidate edges determined by the node degrees. This stems from the observation that nodes with higher degrees exhibit a greater likelihood of forming new connections in real-world networks (Barabási and Albert 1999). The candidate edge set consists of node pairs where at least one node has top-$M$ degree among all the nodes. We set $M = 100$ in our experiments. PULL selects top-$K$ edges among the candidate edge set instead of all node pairs to approximate $\bar{\mathcal{G}}$.

## Iterative Learning of Link Predictor (C3)

At each iteration, PULL constructs the expected graph $\bar{\mathcal{G}}$ given a trained link predictor $f_\theta$ with current parameter $\theta$. Then PULL propagates information through $\bar{\mathcal{G}}'$, which is the approximated version of $\bar{\mathcal{G}}$, to train a new link predictor $f_{\theta^{\text{new}}}$ with new parameter $\theta^{\text{new}}$. This effectively prevents the link predictor from blindly trusting the given edge-incomplete $\mathcal{G}_{\mathcal{P}}$.

To optimize the new parameter $\theta^{\text{new}}$, we propose the binary cross entropy (BCE) loss $\mathcal{L}_E$ in Equation (5) by treating the given edges in $\mathcal{E}_{\mathcal{P}}$ and the unconnected edges in $\mathcal{E}_{\mathcal{U}}$ as positive and unlabeled (PU) examples, respectively. For the unconnected edges, we use the expected linking probabilities $\mathbf{A}^{\bar{\mathcal{G}}'}_{ij}$ as pseudo labels for $e_{ij}$:

$$\begin{aligned} \mathcal{L}_E = &- \sum_{e_{ij} \in \mathcal{E}_{\mathcal{P}}} \log \hat{y}_{ij} - \sum_{e_{ij} \in \mathcal{E}^r_{\mathcal{U}}} \log(1 - \hat{y}_{ij}) \\ &- \sum_{e_{ij} \in \mathcal{E}^r_{\mathcal{P}}} \left( \mathbf{A}^{\bar{\mathcal{G}}'}_{ij} \log \hat{y}_{ij} + (1 - \mathbf{A}^{\bar{\mathcal{G}}'}_{ij}) \log(1 - \hat{y}_{ij}) \right) \end{aligned} \tag{5}$$

where $\hat{y}_{ij} = f_{\theta^{\text{new}}}(i,j)$. $\mathcal{E}^r_{\mathcal{P}}$ is the set of relatively connected edges selected from $\mathcal{E}_{\mathcal{U}}$ when approximating the expected graph structure $\bar{\mathcal{G}}$ by $\bar{\mathcal{G}}'$, and $\mathcal{E}^r_{\mathcal{U}} = \mathcal{E}_{\mathcal{U}} \setminus \mathcal{E}^r_{\mathcal{P}}$.

However, in real-world graphs, there is a severe imbalance between the numbers of connected edges and unconnected ones. We balance them by randomly sampling $|\mathcal{E}_{\mathcal{P}} \cup \mathcal{E}^r_{\mathcal{P}}|$ unconnected edges among $\mathcal{E}^r_{\mathcal{U}}$ for every epoch. We denote the loss $\mathcal{L}_E$ with a set $\mathcal{E}'_{\mathcal{U}}$ of randomly sampled edges among $\mathcal{E}^r_{\mathcal{U}}$ with size $|\mathcal{E}'_{\mathcal{U}}| = |\mathcal{E}_{\mathcal{P}} \cup \mathcal{E}^r_{\mathcal{P}}|$ as $\mathcal{L}'_E$.

If the current parameter $\theta$ of the link predictor is inaccurate, the quality of the expected graph structure deteriorates, leading to the next iteration's parameter $\theta^{\text{new}}$ becoming even more inaccurate. Thus, we propose another loss term $\mathcal{L}_C$ for correction, which measures the BCE for all observed edges and randomly sampled unconnected edges from $\mathcal{E}^r_{\mathcal{U}}$:

$$\mathcal{L}_C = - \sum_{e_{ij} \in \mathcal{E}_{\mathcal{P}}} \log \tilde{y}_{ij} - \sum_{e_{kl} \in \mathcal{E}''_{\mathcal{U}}} \log(1 - \tilde{y}_{ij}) \tag{6}$$

where $\mathcal{E}''_{\mathcal{U}}$ is the set of randomly sampled node pairs from $\mathcal{E}^r_{\mathcal{U}}$ with size $|\mathcal{E}''_{\mathcal{U}}| = |\mathcal{E}_{\mathcal{P}}|$. $\tilde{y}_{ij}$ is $f_{\theta^{\text{new}}}(i,j)$ computed with the given graph $\mathcal{G}_{\mathcal{P}}$. $\mathcal{L}_C$ effectively prevents excessive self-reinforcement in the link predictor of PULL (Figure 3).

As a result, PULL finds the best parameter $\theta^{\text{new}}$ for each iteration by minimizing the sum of the two loss terms $\mathcal{L}'_E$ and $\mathcal{L}_C$. We denote the final loss function as $\mathcal{L}(\theta^{\text{new}}; \bar{\mathcal{G}}', \mathbf{X}) = \mathcal{L}'_E + \mathcal{L}_C$. The new parameter $\theta^{\text{new}}$ is used as the current $\theta$ for the next iteration. The iterations stop if the maximum number of iterations is reached or an early stopping condition is met.

## Theoretical Analysis

We theoretically analyze PULL in terms of its connection to the EM algorithm, and the time complexity.

**Relation of PULL to EM algorithm.** EM (Expectation-Maximization) (Dempster, Laird, and Rubin 1977) is an iterative method used for estimating model parameter $\theta$ when there are unobserved data. It assigns latent variables $\mathbf{z}$ to the unobserved data, and maximizes the expectation of the log likelihood $\log p(\mathbf{y}, \mathbf{z} \mid \mathbf{X}, \theta)$ in terms of $\mathbf{z}$ to optimize $\theta$ where $\mathbf{y}$ and $\mathbf{X}$ are target and input variables, respectively.

In our problem, the expectation of the log likelihood given the current parameter $\theta$ is written as follows:

$$Q(\theta^{\text{new}} \mid \theta) = \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z} \mid \mathbf{X}, \mathcal{E}_{\mathcal{P}}, \theta)}[\log p(\mathcal{E}_{\mathcal{P}}, \mathbf{z} \mid \mathbf{X}, \theta^{\text{new}})] \quad (7)$$

where $\theta^{\text{new}}$ is the new parameter. The EM algorithm finds $\theta^{\text{new}}$ that maximizes $Q(\theta^{\text{new}} \mid \theta)$, and they are used as $\theta$ in the next iteration. The algorithm is widely used in situations involving latent variables as it always improves the likelihood $Q$ through the iterations (Murphy 2012).

PULL iteratively optimizes the parameter $\theta$ of a link predictor by minimizing both $\mathcal{L}'_E$ and $\mathcal{L}_C$ where $\mathcal{L}'_E$ is the approximation of $\mathcal{L}_E$ in Equation (5). We compare Equations (5) and (7) to show the similarity between the iterative minimization of $\mathcal{L}_E$ in PULL and the iterative maximization of $Q(\theta^{\text{new}} \mid \theta)$ in the EM algorithm.

PULL expresses $p(\mathbf{z} \mid \mathbf{X}, \mathcal{E}_{\mathcal{P}}, \theta)$ in Equation (7) by the multiplication of node potentials as in Equation (2). For the joint probability $p(\mathcal{E}_{\mathcal{P}}, \mathbf{z} \mid \mathbf{X}, \theta^{\text{new}})$ in Equation (7), we approximate it using a link predictor $f_{\theta^{\text{new}}}$ with new parameter $\theta^{\text{new}}$. We consider $f_{\theta_{\text{new}}}$ as a marginalization function that gives marginal linking probabilities for each node pair. We also assume that the marginal distributions obtained by $f_{\theta_{\text{new}}}$ are mutually independent. Then the joint probability is approximated as follows:

$$p(\mathcal{E}_{\mathcal{P}}, \mathbf{z} \mid \mathbf{X}, \theta^{\text{new}}) \approx \prod_{e_{ij} \in \mathcal{E}_{\mathcal{P}}} \hat{y}_{ij} \prod_{e_{ij} \in \mathcal{E}_{\mathcal{U}}} \big(z_{ij}\hat{y}_{ij} + (1-z_{ij})(1-\hat{y}_{ij})\big) \quad (8)$$

where $\hat{y}_{ij} = f_{\theta^{\text{new}}}(i,j)$, and $z_{ij} \in \{1, 0\}$ represents the connectivity between nodes $i$ and $j$.

Using Equations (2) and (8), we derive Theorem 1 that shows the similarity between the iterative minimization of $\mathcal{L}_E$ in PULL and the iterative maximization of $Q$ in the EM.

**Theorem 1.** *Given the assumption in Equation (8), the likelihood $Q(\theta^{\text{new}} \mid \theta)$ of the EM algorithm reduces to the negative of the loss $\mathcal{L}_E$ of* PULL *with the expected graph $\bar{\mathcal{G}}$:*

$$Q(\theta^{\text{new}} \mid \theta) \approx \sum_{e_{ij} \in \mathcal{E}_{\mathcal{P}}} \log \hat{y}_{ij} + \sum_{e_{ij} \in \mathcal{E}_{\mathcal{U}}} \big(\mathbf{A}^{\bar{\mathcal{G}}}_{ij} \log \hat{y}_{ij} + (1 - \mathbf{A}^{\bar{\mathcal{G}}}_{ij}) \log(1 - \hat{y}_{ij})\big) \quad (9)$$

*where $\hat{y}_{ij}$ is the estimated linking probability between nodes $i$ and $j$ by $f_{\theta^{\text{new}}}$, and $\mathbf{A}^{\bar{\mathcal{G}}}$ is the corresponding adjacency matrix of $\bar{\mathcal{G}}$ (see Appendix for proof).*

**Complexity of PULL.** PULL is scalable to large graphs due to its linear scalability with the number of nodes and edges. Let $n_o$ and $n_i$ be the number of outer and inner iterations in Algorithm 1, respectively. For simplicity, we assume PULL has $m$ layers where the number $d$ of features for each node is the same in all layers.

**Theorem 2.** *The time complexity of* PULL *(Algorithm 1) is*

$$O\left(n_o d((n_i m + r n_o n_i m)|\mathcal{E}_{\mathcal{P}}| + (n_i m d + M)|\mathcal{V}|)\right),$$

*which is linear to the numbers $|\mathcal{V}|$ and $|\mathcal{E}_{\mathcal{P}}|$ of nodes and edges in $\mathcal{G}_{\mathcal{P}}$, respectively (see Appendix for proof).*

## Experiments

We conduct experiments to answer the following questions:

**Q1. Link prediction performance.** How accurate is PULL compared to the baselines for predicting links in edge-incomplete graphs?
**Q2. Applying PULL to other baselines.** Does applying PULL to other methods improve the accuracy?
**Q3. Effect of iterative learning.** How does the accuracy of PULL change over iterations?
**Q4. Effect of additional loss.** How does the additional loss term $\mathcal{L}_C$ of PULL contribute to the performance?
**Q5. Scalability.** How does the running time of PULL change as the graph size grows?

### Experimental Settings

**Datasets.** We use seven real-world datasets from various domains which are summarized in Table 4 (in Appendix). PubMed and Cora-full are citation networks where nodes correspond to scientific publications and edges signify citation relationships between the papers. Each node has binary bag-of-words features. Chameleon and Crocodile are Wikipedia networks, with nodes representing web pages and edges representing hyperlinks between them. Node features include keywords or informative nouns extracted from the pages. Facebook is a social network where nodes represent users, and edges indicate mutual likes. Node features represent user-specific information such as age and gender.

**Baselines.** We compare PULL with previous approaches for link prediction. GCN+CE, GAT+CE, and SAGE+CE use GCN (2017), GAT (2017), and GraphSAGE (2017) for computing linking probabilities, respectively. Cross entropy (CE) loss is used and $|\mathcal{E}_{\mathcal{P}}|$ non-edges are sampled randomly from $\mathcal{E}_{\mathcal{U}}$ every epoch to balance the ratio between edge and non-edge examples. GAE and VGAE (2016) are

Table 1: The link prediction accuracy of PULL and baselines in terms of AUROC and AUPRC. Bold numbers denote the best performance, and underlined ones represent the second-best accuracy. PULL outperforms all the baselines in most of the cases.

| Missing ratio $r_m$ = 0.1 | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Model** | **PubMed** | | **Cora-full** | | **Chameleon** | | **Crocodile** | | **Facebook** | |
| | AUROC | AUPRC | AUROC | AUPRC | AUROC | AUPRC | AUROC | AUPRC | AUROC | AUPRC |
| GCN+CE | $96.45 \pm 0.23$ | $96.58 \pm 0.21$ | $95.77 \pm 0.65$ | $95.77 \pm 0.74$ | $96.77 \pm 0.35$ | $96.67 \pm 0.40$ | $96.91 \pm 0.46$ | $97.22 \pm 0.45$ | $97.06 \pm 0.18$ | $97.33 \pm 0.19$ |
| GAT+CE | $90.99 \pm 0.40$ | $89.64 \pm 0.49$ | $94.27 \pm 0.38$ | $93.74 \pm 0.43$ | $91.55 \pm 1.82$ | $91.39 \pm 1.72$ | $90.65 \pm 1.83$ | $91.67 \pm 1.40$ | $92.43 \pm 0.62$ | $92.04 \pm 0.77$ |
| SAGE+CE | $87.22 \pm 1.14$ | $88.34 \pm 0.99$ | $94.35 \pm 0.54$ | $94.77 \pm 0.60$ | $96.30 \pm 0.48$ | $95.87 \pm 0.63$ | $96.00 \pm 0.61$ | $96.55 \pm 0.55$ | $95.17 \pm 0.52$ | $95.34 \pm 0.54$ |
| GAE | $96.35 \pm 0.17$ | $96.46 \pm 0.15$ | $95.51 \pm 0.31$ | $95.52 \pm 0.32$ | $96.88 \pm 0.48$ | $96.80 \pm 0.54$ | $96.67 \pm 0.70$ | $96.78 \pm 1.17$ | $97.00 \pm 0.17$ | $97.27 \pm 0.13$ |
| VGAE | $94.61 \pm 1.01$ | $94.74 \pm 1.00$ | $92.37 \pm 3.89$ | $92.40 \pm 3.68$ | $96.32 \pm 0.27$ | $96.20 \pm 0.26$ | $95.29 \pm 0.40$ | $95.45 \pm 0.82$ | $96.29 \pm 0.27$ | $96.49 \pm 0.28$ |
| ARGA | $93.67 \pm 0.71$ | $93.35 \pm 0.73$ | $91.39 \pm 1.02$ | $90.72 \pm 1.15$ | $94.76 \pm 0.51$ | $94.37 \pm 0.71$ | $96.03 \pm 0.38$ | $95.65 \pm 0.35$ | $92.03 \pm 0.59$ | $92.19 \pm 0.48$ |
| ARGVA | $93.56 \pm 1.21$ | $93.80 \pm 1.11$ | $89.88 \pm 3.13$ | $89.59 \pm 2.88$ | $94.26 \pm 0.74$ | $94.32 \pm 0.70$ | $95.04 \pm 0.18$ | $94.32 \pm 0.59$ | $92.35 \pm 2.58$ | $92.76 \pm 2.36$ |
| VGNAE | $95.99 \pm 0.63$ | $95.99 \pm 0.55$ | $95.42 \pm 1.23$ | $95.14 \pm 1.34$ | $97.46 \pm 0.43$ | $97.17 \pm 0.53$ | $96.34 \pm 0.76$ | $95.29 \pm 1.87$ | $95.79 \pm 0.52$ | $95.89 \pm 0.54$ |
| Bagging-PU | $94.55 \pm 0.39$ | $94.88 \pm 0.37$ | $92.74 \pm 0.62$ | $94.20 \pm 0.77$ | $97.27 \pm 0.77$ | $97.14 \pm 0.89$ | $97.47 \pm 0.44$ | $97.75 \pm 0.39$ | $97.02 \pm 0.15$ | $97.38 \pm 0.14$ |
| NESS | $95.15 \pm 0.24$ | $95.01 \pm 0.23$ | $\mathbf{96.29 \pm 0.20}$ | $96.16 \pm 0.27$ | $96.86 \pm 0.12$ | $96.83 \pm 0.09$ | $95.88 \pm 0.46$ | $96.74 \pm 0.34$ | $95.75 \pm 0.21$ | $96.17 \pm 0.14$ |
| PULL (proposed) | $\mathbf{96.59 \pm 0.19}$ | $\mathbf{96.83 \pm 0.18}$ | $96.06 \pm 0.34$ | $\mathbf{96.25 \pm 0.35}$ | $\mathbf{97.87 \pm 0.33}$ | $\mathbf{97.83 \pm 0.33}$ | $\mathbf{98.31 \pm 0.20}$ | $\mathbf{98.36 \pm 0.22}$ | $\mathbf{97.41 \pm 0.11}$ | $\mathbf{97.67 \pm 0.09}$ |

Table 2: The performance improvement of baselines with the integration of PULL. The best performance is indicated in bold. Note that PULL enhances the performance of baseline models.

| Missing ratio $r_m$ = 0.1 | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Model** | **PubMed** | | **Cora-full** | | **Chameleon** | | **Crocodile** | | **Facebook** | |
| | AUROC | AUPRC | AUROC | AUPRC | AUROC | AUPRC | AUROC | AUPRC | AUROC | AUPRC |
| GAE | $96.35 \pm 0.17$ | $96.46 \pm 0.15$ | $95.51 \pm 0.31$ | $95.52 \pm 0.32$ | $96.88 \pm 0.48$ | $96.80 \pm 0.54$ | $96.67 \pm 0.70$ | $96.78 \pm 1.17$ | $97.00 \pm 0.17$ | $97.27 \pm 0.13$ |
| GAE+PULL | $\mathbf{96.64 \pm 0.22}$ | $\mathbf{96.86 \pm 0.21}$ | $\mathbf{96.00 \pm 0.48}$ | $\mathbf{96.12 \pm 0.58}$ | $\mathbf{98.04 \pm 0.18}$ | $\mathbf{98.05 \pm 0.15}$ | $\mathbf{98.22 \pm 0.18}$ | $\mathbf{98.31 \pm 0.17}$ | $\mathbf{97.41 \pm 0.14}$ | $\mathbf{97.67 \pm 0.11}$ |
| VGAE | $94.61 \pm 1.01$ | $94.74 \pm 1.00$ | $92.37 \pm 3.89$ | $92.40 \pm 3.68$ | $96.32 \pm 0.27$ | $96.20 \pm 0.26$ | $95.29 \pm 0.40$ | $95.45 \pm 0.82$ | $96.29 \pm 0.27$ | $96.49 \pm 0.28$ |
| VGAE+PULL | $\mathbf{95.81 \pm 0.51}$ | $\mathbf{95.92 \pm 0.50}$ | $\mathbf{93.75 \pm 3.17}$ | $\mathbf{93.85 \pm 3.01}$ | $\mathbf{97.24 \pm 0.47}$ | $\mathbf{97.29 \pm 0.49}$ | $\mathbf{97.17 \pm 0.73}$ | $\mathbf{97.33 \pm 0.64}$ | $\mathbf{96.56 \pm 0.25}$ | $\mathbf{96.72 \pm 0.26}$ |
| VGNAE | $95.99 \pm 0.63$ | $95.99 \pm 0.55$ | $95.42 \pm 1.23$ | $95.14 \pm 1.34$ | $97.46 \pm 0.43$ | $97.17 \pm 0.53$ | $96.34 \pm 0.76$ | $95.29 \pm 1.87$ | $95.79 \pm 0.52$ | $95.89 \pm 0.54$ |
| VGNAE+PULL | $\mathbf{96.22 \pm 0.37}$ | $\mathbf{96.23 \pm 0.37}$ | $\mathbf{96.02 \pm 0.64}$ | $\mathbf{95.75 \pm 0.70}$ | $\mathbf{97.75 \pm 0.36}$ | $\mathbf{97.46 \pm 0.43}$ | $\mathbf{97.23 \pm 0.73}$ | $\mathbf{96.91 \pm 0.79}$ | $\mathbf{95.83 \pm 0.46}$ | $\mathbf{95.91 \pm 0.44}$ |

autoencoder-based frameworks for link prediction. ARGA and ARGVA (2018) respectively improve the performance of GAE and VGAE by introducing adversarial training strategy. VGNAE (2021) utilizes L2-normalization to obtain better node embeddings for isolated nodes. Bagging-PU (2022) classifies node pairs into observed and unobserved, and approximates the linking probabilities using the ratio of observed links. NESS (2023) is the current state-of-the-art link prediction method. It utilizes multiple static subgraphs during training, and aggregates the learned node embeddings to obtain the linking probabilities at test time. More implementation details are described in the Appendix section.

**Evaluation and Settings.** We evaluate the performance of PULL and the baselines in classifying edges and non-edges correctly. We use AUROC (AUC) and AUPRC (AP) scores as the evaluation metric (Kipf and Welling 2017). The models are trained using graphs that lack some edges, while preserving all node attributes. The validation and test sets consist of the missing edges and an equal number of randomly sampled non-edges. We vary the ratio $r_m$ of test missing edges in $\{0.1, 0.2\}$. The ratio of valid missing edges are set to 0.1 through the experiments. The validation set is employed for early stopping with patience 20, and the number of maximum epochs is set to 2,000. The epochs are not halted until 500 since the accuracy oscillates in the earlier epochs. For PULL, we set the number of inner loops as 200, and the maximum number of iterations as 10. The iterations stop if the current validation AUROC is smaller than that of the previous iteration. We use Adam optimizer with a learning rate of 0.01, and set the numbers of layers and hidden dimensions as 2 and 16, respectively, following the original GCN paper (2017) for fair comparison between the methods. For the hyperparameters of the baselines, we use the default settings described in their papers. We repeat the experiments ten times with different random seeds and present the results in terms of both average and standard deviation.

## Link Prediction Performance (Q1)

We compare the link prediction performance of PULL with the baselines in Table 1 while setting the ratio $r_m$ of test missing edges as 0.1. We also report the performance with $r_m = 0.2$ in Appendix. Note that PULL achieves the highest AUROC and AUPRC scores among the methods in most of the cases. This highlights the significance of considering the uncertainty in the given edge-incomplete graph to enhance the prediction performance. It is also noteworthy that GCN+CE model, which propagates information through the edge-incomplete graph using GCN, shows consistently lower performance than PULL. This shows that the propagation of PULL with the expected graph structure effectively prevents $f$ from overtrusting the given graph structure, whereas the propagation of GCN+CE with the given graph leads to overfitting.

## Applying PULL to Other Methods (Q2)

PULL can be applied to other GCN-based methods including GAE, VGAE, and VGNAE that use GCN-based propagation scheme. To show that PULL improves the performance of existing models, we conduct experiments by applying PULL to them while setting the ratio $r_m$ of test missing edges as 0.1. We also report the experimental results with $r_m = 0.2$ in Appendix. It is not easy for PULL to be directly applied to other baselines such as GAT, GraphSAGE, ARGA, and ARGVA. This is because they use different propagation schemes instead of GCN, posing a challenge for PULL in propagating information through the expected

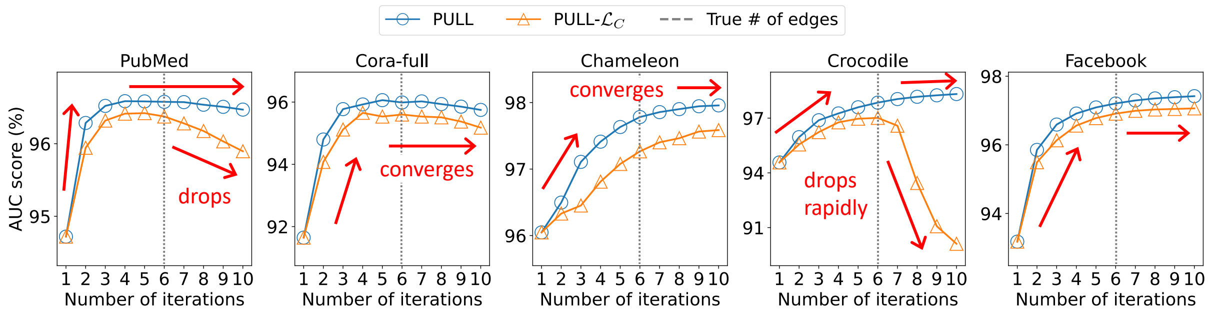


Figure 3: AUC score of PULL and PULL-$\mathcal{L}_C$ through the iterations. PULL-$\mathcal{L}_C$ represents PULL without $\mathcal{L}_C$. The dashed gray lines denote the ground-truth numbers of edges. The accuracy of PULL increases in early iterations, and converges or slightly increases as the number $K$ of sampled edges exceeds the ground-truth one. This shows that PULL improves the quality of the expected graph with each iteration. Moreover, PULL consistently shows superior performance than PULL-$\mathcal{L}_C$. In PubMed and Crocodile, the accuracy of PULL-$\mathcal{L}_C$ drops rapidly after exceeding the dashed gray lines. This demonstrates that $\mathcal{L}_C$ protects PULL from performance degradation when the expected graph structure has more edges than the actual graph.

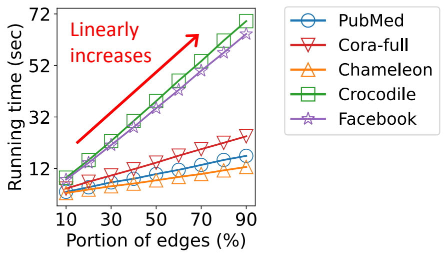


Figure 4: The running time of PULL on sampled subgraphs. The time increases linearly with the number of edges.

graph structure during training. Table 2 summarizes the results. Note that PULL improves the accuracy of the baselines in most of the cases, demonstrating its effectiveness across a range of different models.

## Effect of Iterative Learning (Q3)

For each iteration, PULL computes the expected graph $\bar{\mathcal{G}}$ utilizing the trained link predictor $f$ from the previous iteration. Then PULL retrains $f$ with $\bar{\mathcal{G}}$ to prevent the link predictor from blindly trusting the given graph. We study how the accuracy of PULL evolves as the iteration proceeds in Figure 3. PULL increases the number $K$ of selected edges for the approximation of $\bar{\mathcal{G}}$ as the iteration progresses. The dashed gray lines indicate the points at which $K$ becomes equal to the ground-truth number of edges for each dataset.

The AUC score of PULL in Figure 3 increases through the iterations, reaching its best performance when the number $K$ of selected edges closely matches the ground-truth one. This shows that PULL enhances the quality of the expected graph as the iterations progress, and eventually makes accurate predictions of the true graph structure. In Crocodile and Facebook, the prediction accuracy increases even with larger number of edges than the ground-truth one. This observation indicates that the ground-truth graph structures of Crocodile and Facebook inherently contain missing links.

## Effect of Additional Loss (Q4)

We study the effect of the additional loss term $\mathcal{L}_C$ of PULL on the link prediction performance in Figure 3. PULL-$\mathcal{L}_C$ represents PULL trained by minimizing only $\mathcal{L}'_E$. Note that PULL-$\mathcal{L}_C$ consistently shows lower accuracy than PULL, showing the importance of $\mathcal{L}_C$. In PubMed and Crocodile, the AUC scores of PULL-$\mathcal{L}_C$ drop rapidly after the fifth iteration, where the number $K$ of selected edges exceeds the ground-truth one. This indicates that $\mathcal{L}_C$ effectively safeguards PULL against performance degradation when the expected graph contains more edges than the actual one.

## Scalability (Q5)

We study the running time of PULL on subgraphs with different sizes to show its scalability to large graphs in Figure 4. We randomly sample edges from the original graphs with various portions $r_p \in \{0.1, ..., 0.9\}$. Thus, each induced subgraph has $r_p|\mathcal{E}|$ edges where $\mathcal{E}$ is the set of edges of the original graph. Figure 4 shows that the runtime of PULL exhibits a linear increase with the number of edges, showing its scalability to large graphs. This is because PULL effectively approximates the expected graph $\bar{\mathcal{G}}$ with $|\mathcal{V}|^2$ weighted edges by a graph $\bar{\mathcal{G}}'$ with $(1 + 0.05t)|\mathcal{E}_\mathcal{P}|$ edges where $\mathcal{V}$ and $\mathcal{E}_\mathcal{P}$ are sets of nodes and observed edges, respectively, and $t$ is the number of iterations.

## Conclusion

We propose PULL, an accurate method for link prediction in edge-incomplete graphs. PULL addresses the limitation of previous approaches, which is their heavy reliance on the observed graph, by iteratively predicting the true graph structure. PULL proposes latent variables for the unconnected edges in a graph, and propagates information through the expected graph structure. PULL then uses the expected linking probabilities of unconnected edges as their pseudo labels for training a link predictor. Extensive experiments on real-world datasets show that PULL shows superior performance than the baselines.

## References

Afoudi, Y.; Lazaar, M.; and Hmaidi, S. 2023. An enhanced recommender system based on heterogeneous graph link prediction. *Engineering Applications of Artificial Intelligence*, 124: 106553.

Ahn, S.; and Kim, M. H. 2021. Variational Graph Normalized AutoEncoders. In *Conference on Information and Knowledge Management (CIKM)*, 2827–2831. ACM.

Barabási, A.-L.; and Albert, R. 1999. Emergence of Scaling in Random Networks. *Science*.

Chamberlain, B. P.; Shirobokov, S.; Rossi, E.; Frasca, F.; Markovich, T.; Hammerla, N. Y.; Bronstein, M. M.; and Hansmire, M. 2023. Graph Neural Networks for Link Prediction with Subgraph Sketching. In *International Conference on Learning Representations (ICLR)*.

Daud, N. N.; Hamid, S. H. A.; Saadoon, M.; Sahran, F.; and Anuar, N. B. 2020. Applications of link prediction in social networks: A review. *Journal of Network and Computer Applications*, 166: 102716.

Dempster, A. P.; Laird, N. M.; and Rubin, D. B. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society: series B (methodological)*.

Gan, S.; Alshahrani, M.; and Liu, S. 2022. Positive-Unlabeled Learning for Network Link Prediction. *Mathematics*, 10(18): 3345.

Hamilton, W. L.; Ying, Z.; and Leskovec, J. 2017. Inductive Representation Learning on Large Graphs. In *Conference on Neural Information Processing Systems (NeurIPS)*, 1024–1034.

Hao, Y. 2021. *Learning node embedding from graph structure and node attributes*. Ph.D. thesis, UNSW Sydney.

Hinton, G. E.; and Salakhutdinov, R. R. 2006. Reducing the dimensionality of data with neural networks. *science*, 313(5786): 504–507.

Jung, J.; Park, H.; and Kang, U. 2020. BalanSiNG: Fast and Scalable Generation of Realistic Signed Networks. In *EDBT*.

Kim, J.; Jung, J.; and Kang, U. 2021. Compressing deep graph convolution network with multi-staged knowledge distillation. *Plos one*.

Kipf, T. N.; and Welling, M. 2016. Variational Graph Auto-Encoders. *CoRR*, abs/1611.07308.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR) (Poster)*. OpenReview.net.

Kiryo, R.; Niu, G.; du Plessis, M. C.; and Sugiyama, M. 2017. Positive-Unlabeled Learning with Non-Negative Risk Estimator. In *Conference on Neural Information Processing Systems (NeurIPS)*, 1675–1685.

Koller, D.; and Friedman, N. 2009. *Probabilistic graphical models: principles and techniques*. MIT press.

Li, J.; Lu, G.; Wu, Z.; and Ling, F. 2023. Multi-view representation model based on graph autoencoder. *Information Sciences*, 632: 439–453.

Liu, H.; Kou, H.; Yan, C.; and Qi, L. 2019. Link prediction in paper citation network to construct paper correlation graph. *European Association For Signal Processing (EURASIP) Journal on Wireless Communications and Networking*, 2019: 233.

Long, Y.; Wu, M.; Liu, Y.; Fang, Y.; Kwoh, C. K.; Chen, J.; Luo, J.; and Li, X. 2022. Pre-training graph neural networks for link prediction in biomedical networks. *Bioinformatics*, 38(8): 2254–2262.

Luo, C.; Zhao, P.; Chen, C.; Qiao, B.; Du, C.; Zhang, H.; Wu, W.; Cai, S.; He, B.; Rajmohan, S.; and Lin, Q. 2021. PULNS: Positive-Unlabeled Learning with Effective Negative Sample Selector. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 8784–8792. AAAI Press.

Ma, S.; and Zhang, R. 2017. In *2017 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, 537–542. IEEE.

Murphy, K. P. 2012. *Machine learning: a probabilistic perspective*. MIT press.

Pan, S.; Hu, R.; Long, G.; Jiang, J.; Yao, L.; and Zhang, C. 2018. Adversarially Regularized Graph Autoencoder for Graph Embedding. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2609–2615. ijcai.org.

Shibata, N.; Kajikawa, Y.; and Sakata, I. 2012. Link prediction in citation networks. *Journal of the Association for Information Science and Technology*, 63(1): 78–85.

Tan, Q.; Zhang, X.; Liu, N.; Zha, D.; Li, L.; Chen, R.; Choi, S.; and Hu, X. 2023. Bring Your Own View: Graph Neural Networks for Link Prediction with Personalized Subgraph Selection. In *International Conference on Web Search and Data Mining (WSDM)*.

Ucar, T. 2023. NESS: Learning Node Embeddings from Static SubGraphs. *CoRR*.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2017. Graph Attention Networks.

Wang, P.; Xu, B.; Wu, Y.; and Zhou, X. 2015. Link prediction in social networks: the state-of-the-art. *Science China Information Sciences*, 58(1): 1–38.

Wu, M.; Pan, S.; Du, L.; Tsang, I. W.; Zhu, X.; and Du, B. 2019. Long-short Distance Aggregation Networks for Positive Unlabeled Graph Learning. In *Conference on Information and Knowledge Management (CIKM)*, 2157–2160. ACM.

Yang, H.; Zhang, Y.; Yao, Q.; and Kwok, J. T. 2023. Positive-Unlabeled Node Classification with Structure-aware Graph Learning. In *Conference on Information and Knowledge Management (CIKM)*, 4390–4394. ACM.

Yoo, J.; Jeon, H.; Jung, J.; and Kang, U. 2022a. Accurate Node Feature Estimation with Structured Variational Graph Autoencoder. In *KDD*.

Yoo, J.; Kim, J.; Yoon, H.; Kim, G.; Jang, C.; and Kang, U. 2021. Accurate Graph-Based PU Learning without Class Prior. In *International Conference on Data Mining (ICDM)*, 827–836. IEEE.

Yoo, J.; Kim, J.; Yoon, H.; Kim, G.; Jang, C.; and Kang, U. 2022b. Graph-based PU learning for binary and multiclass classification without class prior. *Knowledge and Information Systems*.

Yoo, J.; Shim, S.; and Kang, U. 2022. Model-Agnostic Augmentation for Accurate Graph Classification. In *WWW*.

You, J.; Gomes-Selman, J. M.; Ying, R.; and Leskovec, J. 2021. Identity-aware graph neural networks. In *Association for the Advancement of Artificial Intelligence (AAAI)*, volume 35, 10737–10745.

Yun, S.; Kim, S.; Lee, J.; Kang, J.; and Kim, H. J. 2022. Neo-GNNs: Neighborhood Overlap-aware Graph Neural Networks for Link Prediction. *CoRR*, abs/2206.04216.

Zhang, C.; Ren, D.; Liu, T.; Yang, J.; and Gong, C. 2019. Positive and Unlabeled Learning with Label Disambiguation. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 4250–4256. ijcai.org.

Zhang, M.; and Chen, Y. 2018. Link Prediction Based on Graph Neural Networks. In *Conference on Neural Information Processing Systems (NeurIPS)*, 5171–5181.

Zhang, M.; Li, P.; Xia, Y.; Wang, K.; and Jin, L. 2021. Labeling Trick: A Theory of Using Graph Neural Networks for Multi-Node Representation Learning. In *Conference on Neural Information Processing Systems (NeurIPS)*, 9061–9073.

Zhao, Y.; Xu, Q.; Jiang, Y.; Wen, P.; and Huang, Q. 2022. Dist-PU: Positive-Unlabeled Learning from a Label Distribution Perspective. In *Computer Vision and Pattern Recognition Conference (CVPR)*, 14441–14450. IEEE.

Zhu, Z.; Zhang, Z.; Xhonneux, L. A. C.; and Tang, J. 2021. Neural Bellman-Ford Networks: A General Graph Neural Network Framework for Link Prediction. In *Conference on Neural Information Processing Systems (NeurIPS)*, 29476–29490.