Accurate Link Prediction for Edge-Incomplete Graphs via PU Learning (Supplementary Document)

Abstract

We provide additional information including the problem definition, symbols, lemma, proofs, detailed experimental settings, and further experimental results.

Problem Definition

Problem 1 (Link Prediction in Edge-incomplete Graphs)**.** *We have an edge-incomplete graph $\mathcal{G}_{\mathcal{P}} = (\mathcal{V}, \mathcal{E}_{\mathcal{P}})$, along with a feature matrix $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d}$ where $\mathcal{V}$ and $\mathcal{E}_{\mathcal{P}}$ are the sets of nodes and observed edges, respectively, and d is the number of features for each node. The remaining unconnected node pairs are denoted as a set $\mathcal{E}_{\mathcal{U}}$. The objective of link prediction in edge-incomplete graphs is to train a link predictor f that accurately identifies the connected node pairs within $\mathcal{E}_{\mathcal{U}}$.*

Symbols

Table 3: Symbols.

Symbol	Description
$\mathcal{G}_{\mathcal{P}} = (\mathcal{V}, \mathcal{E}_{\mathcal{P}})$	Edge-incomplete graph with sets $\mathcal{V}$ of nodes and $\mathcal{E}_{\mathcal{P}}$ of observed edges
$\mathcal{E}_{\mathcal{U}}$	Set of unconnected node pairs (unconnected edges)
e_{ij}	Edge between nodes i and j
$L(\mathcal{G})$	Corresponding line graph of $\mathcal{G}$
$\mathbf{A}^{\mathcal{G}}$	Corresponding adjacency matrix of $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathbf{A}^{\mathcal{G}}_{ij} = 1$ if $e_{ij} \in \mathcal{E}$
$\mathbf{X}$	Feature matrix for every node in $\mathcal{G}_{\mathcal{P}}$
$f_\theta(\cdot, \cdot)$	Link predictor parameterized by θ
$\mathcal{L}(\cdot)$	Objective function that PULL aims to minimize
$\bar{\mathcal{G}}$	Expected graph structure
$\bar{\mathcal{G}}'$	Approximated version of $\bar{\mathcal{G}}$

Lemma

Lemma 1. *We are given a graph $\mathcal{G}_{\mathcal{P}}$ and its corresponding line graph $L(\mathcal{G}_{\mathcal{P}}) = (\mathcal{V}_L, \mathcal{E}_L)$ where $\mathcal{V}_L$ and $\mathcal{E}_L$ are sets of nodes and edges in $L(\mathcal{G}_{\mathcal{P}})$, respectively. We are also given node potentials $\phi_{ij}(z_{ij} \mid \theta)$ of nodes ij in graph $L(\mathcal{G}_{\mathcal{P}})$. Then the equation $\sum_{\mathbf{z}} \prod_{ij \in \mathcal{V}_L} \phi_{ij}(z_{ij} \mid \theta) = 1$ holds for $\sum_{z_{ij}} \phi_{ij}(z_{ij} \mid \theta) = 1$.*

Proof. Let $N = |\mathcal{V}_L|$, and $\mathcal{E}$ be the set of all observed edges and unconnected edges in $\mathcal{G}_{\mathcal{P}}$. Then the sum of $\prod_{ij \in \mathcal{V}_L} \phi_{ij}(z_{ij}|\theta)$ for all possible $\mathbf{z}$ is computed as follows:

$$\begin{aligned}
&\sum_{\mathbf{z}} \prod_{ij \in \mathcal{V}_L} \phi_{ij}(z_{ij} \mid \theta) = \sum_{\mathbf{z}} \prod_{e_{ij} \in \mathcal{E}} \phi_{ij}(z_{ij} \mid \theta) \\
&= \sum_{\mathbf{z} \setminus \{z_{11}\}} \prod_{e_{ij} \in \mathcal{E} \setminus \{e_{11}\}} \phi_{ij}(z_{ij} \mid \theta) \sum_{z_{11}} \phi_{11}(z_{11} \mid \theta) \\
&= \sum_{\mathbf{z} \setminus \{z_{11}, z_{12}\}} \prod_{e_{ij} \in \mathcal{E} \setminus \{e_{11}, e_{12}\}} \phi_{ij}(z_{ij} \mid \theta) \sum_{z_{12}} \phi_{12}(z_{12} \mid \theta) \\
&= \cdots = \sum_{z_{NN}} \phi_{NN}(z_{NN} \mid \theta) = 1
\end{aligned}$$

which ends the proof. Similarly, the equation $\sum_{\mathbf{z}|z_{ij}=1} \prod_{e_{kl} \in \mathcal{E}_{\mathcal{U}} \setminus \{e_{ij}\}} \phi_{kl}(z_{kl} \mid \theta) = 1$ holds for $\sum_{z_{ij}} \phi_{ij}(z_{ij} \mid \theta) = 1$. □

Proofs for Theorems

Proof of Theorem 1

Proof. Using Equations (2) and (8), the expected log likelihood $Q(\theta^{\text{new}} \mid \theta)$ in Equation (7) is expressed as follows:

$$\begin{aligned}
&Q(\theta^{\text{new}} \mid \theta) = \sum_{\mathbf{z}} p(\mathbf{z} \mid \mathbf{X}, \mathcal{E}_{\mathcal{P}}, \theta) \log p(\mathcal{E}_{\mathcal{P}}, \mathbf{z} \mid \mathbf{X}, \theta^{\text{new}}) \\
&\approx \sum_{\mathbf{z}} p(\mathbf{z} \mid \mathbf{X}, \mathcal{E}_{\mathcal{P}}, \theta) \Big(\sum_{e_{ij} \in \mathcal{E}_{\mathcal{P}}} \log \hat{y}_{ij} + \sum_{e_{ij} \in \mathcal{E}_{\mathcal{U}}} \log\big(\hat{y}_{ij}(z_{ij})\big) \Big) \\
&= \sum_{e_{ij} \in \mathcal{E}_{\mathcal{P}}} \log \hat{y}_{ij} + \sum_{\mathbf{z}} \prod_{e_{kl} \in \mathcal{E}_{\mathcal{U}}} \phi_{kl}(z_{kl} \mid \theta) \sum_{e_{ij} \in \mathcal{E}_{\mathcal{U}}} \log\big(\hat{y}_{ij}(z_{ij})\big)
\end{aligned} \tag{10}$$

where $\hat{y}_{ij}(z_{ij}) = z_{ij}\hat{y}_{ij} + (1 - z_{ij})(1 - \hat{y}_{ij})$.

The last term in Equation (10) is expressed as follows:

$$\begin{aligned}
&\sum_{\mathbf{z}} \prod_{e_{kl}\in\mathcal{E}_{\mathcal{U}}} \phi_{kl}(z_{kl} \mid \theta) \sum_{e_{ij}\in\mathcal{E}_{\mathcal{U}}} \log\big(\hat{y}_{ij}(z_{ij})\big) \\
&= \sum_{\mathbf{z}} \prod_{e_{kl}\in\mathcal{E}_{\mathcal{U}}\setminus\{e_{ij}\}} \phi_{kl}(z_{kl} \mid \theta) \sum_{e_{ij}\in\mathcal{E}_{\mathcal{U}}} \phi_{ij}(z_{ij} \mid \theta) \log\big(\hat{y}_{ij}(z_{ij})\big) \\
&= \sum_{e_{ij}\in\mathcal{E}_{\mathcal{U}}} \big(\phi_{ij}(z_{ij}=1 \mid \theta)\log\hat{y}_{ij} + \phi_{ij}(z_{ij}=0 \mid \theta)\log(1-\hat{y}_{ij})\big) \\
&= \sum_{e_{ij}\in\mathcal{E}_{\mathcal{U}}} \big(\mathbf{A}^{\bar{\mathcal{G}}}_{ij}\log\hat{y}_{ij} + (1-\mathbf{A}^{\bar{\mathcal{G}}}_{ij})\log(1-\hat{y}_{ij})\big)
\end{aligned} \tag{11}$$

where the second equality uses the fact that $\sum_{\mathbf{z}\setminus\{z_{ij}\}} \prod_{e_{kl}\in\mathcal{E}_{\mathcal{U}}\setminus\{e_{ij}\}} \phi_{kl}(z_{kl} \mid \theta) = 1$ (from Lemma 1), and the third equality uses Equation (4).

Using the result of Equation (11), the expected log likelihood $Q(\theta^{\text{new}} \mid \theta)$ reduces to the negative of the loss function $\mathcal{L}_E$ of PULL:

$$\begin{aligned}
Q(\theta^{\text{new}} \mid \theta) \approx &\sum_{e_{ij}\in\mathcal{E}_{\mathcal{P}}} \log\hat{y}_{ij} \\
&+ \sum_{e_{ij}\in\mathcal{E}_{\mathcal{U}}} \big(\mathbf{A}^{\bar{\mathcal{G}}}_{ij}\log\hat{y}_{ij} + (1-\mathbf{A}^{\bar{\mathcal{G}}}_{ij})\log(1-\hat{y}_{ij})\big)
\end{aligned} \tag{12}$$

which ends the proof. Note that Equation (12) uses $\bar{\mathcal{G}}$ which is approximated to $\bar{\mathcal{G}}'$ in PULL. □

Proof of Theorem 2

Proof. Let n_o be the number of iterations in Algorithm 1, and n_i be the number of gradient-descent updates for obtaining the model parameter θ^{new} in line 7 of the algorithm. Each iteration of PULL consists of two steps: 1) generating the expected graph structure $\bar{\mathcal{G}}$, and 2) training the link predictor f using the approximated one of $\bar{\mathcal{G}}$. The time complexity of generating the expected graph structure is $O(d(M|\mathcal{V}|))$ since we compute the linking probabilities for node pairs where at least one node of each pair belongs to the set of nodes with top-M largest degree. The time complexity for training f in the k-th iteration is $O\left(n_i m d((1+rk)|\mathcal{E}_{\mathcal{P}}| + d|\mathcal{V}|)\right)$ where r is the increase factor of edges for approximating the expected graph structure $\bar{\mathcal{G}}$. Note that the complexity for training f is upper-bounded by $O\left(n_i m d((1+rn_o)|\mathcal{E}_{\mathcal{P}}| + d|\mathcal{V}|)\right)$ since $1+rk \le 1+rn_o$. As a result, the time complexity of PULL is computed as

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

which ends the proof.

□

Detailed Settings of Experiments

Datasets

The statistics of datasets are summarized in Table 4.

Implementation

We provide detailed settings of implementation for PULL and the baselines. All the experiments are conducted under a single GPU machine with GTX 1080 Ti.

Table 4: Summary of datasets.

Datasets	Nodes	Edges	Features	Description
PubMed[1]	19,717	88,648	500	Citation
Cora-full[2]	19,793	126,842	8,710	Citation
Chameleon[3]	2,277	36,101	2,325	Wikipedia
Crocodile[3]	11,631	191,506	500	Wikipedia
Facebook[4]	22,470	342,004	128	Social
Physics[5]	34,493	495,924	8,415	Citation
ogbn-arxiv[6]	169,343	1,166,243	128	Citation

[1] https://github.com/kimiyoung/planetoid
[2] https://www.cs.cit.tum.de/daml/g2g/
[3] https://snap.stanford.edu/data/wikipedia-article-networks.html
[4] https://github.com/benedekrozemberczki/MUSAE
[5] https://github.com/shchur/gnn-benchmark/raw/master/data/npz/
[6] https://ogb.stanford.edu/docs/nodeprop/#ogbn-arxiv

GCN+CE. We use the GCN code implemented with torch-geometric package (Fey and Lenssen 2019). For each epoch, the model randomly samples $|\mathcal{E}_{\mathcal{P}}|$ negative samples (unconnected node pairs), and minimizes the cross entropy (CE) loss.

GAT+CE. We use the GAT code provided by torch-geometric package. For each epoch, GAT+CE randomly samples $|\mathcal{E}_{\mathcal{P}}|$ negative samples, and minimizes the cross entropy loss. We set the multi-head attention number as 8 with the mean aggregation strategy, and the dropout ratio as 0.6 following the original paper (Velickovic et al. 2017).

SAGE+CE. We use the GraphSAGE code implemented with torch-geometric package. For each epoch, the model randomly samples $|\mathcal{E}_{\mathcal{P}}|$ negative samples, and minimizes the cross entropy loss. We use the mean aggregation following the original paper (Hamilton, Ying, and Leskovec 2017).

GAE & VGAE. We use the GAE and VGAE codes implemented with torch-geometric package. We use GCN-based encoder and decoder for both GAE and VGAE following the original paper (Kipf and Welling 2016). The number of layers and units for decoders are set to 2 and 16, respectively.

ARGA & ARGVA. We use the ARGA and ARGVA codes implemented with torch-geometric package. We use the same hyperparameter settings for the adversarial training as presented in the original paper (Pan et al. 2018).

VGNAE. We use the VGNAE code implemented by the authors (Ahn and Kim 2021). The scaling constant s is set to 1.8 following the original paper.

Bagging-PU. We reimplement Bagging-PU since there is no public implementation of authors. We use GCN instead of SDNE (Wang, Cui, and Zhu 2016) for the node embedding model since SDNE is an unsupervised representation-based method, which limits the performance. We use the mean aggregation following the original paper (Gan, Alshahrani, and Liu 2022), and set the bagging size as 3.

NESS. We use the NESS code implemented by the authors (Ucar 2023). We use GCN-based embedding model. For the other hyperparameters, we use the default settings described in their paper.

Table 5: The link prediction accuracy of PULL and the baselines in larger datasets. Bold numbers denote the best performance, and underlined ones denote the second-best performance. Note that PULL gives the highest accuracy among the methods, showing its efficacy in larger graphs.

Missing ratio r_m = 0.1				
Model	**Physics**		**ogbn-arxiv**	
	AUROC	AUPRC	AUROC	AUPRC
GCN+CE	96.90 ± 0.19	96.65 ± 0.23	80.58 ± 0.13	85.11 ± 0.10
GAT+CE	93.58 ± 0.46	92.23 ± 0.52	82.31 ± 0.22	79.46 ± 0.43
SAGE+CE	95.40 ± 0.47	94.95 ± 0.49	83.07 ± 1.60	81.01 ± 1.07
GAE	96.81 ± 0.13	96.56 ± 0.14	80.62 ± 0.14	85.20 ± 0.11
VGAE	95.00 ± 0.82	94.51 ± 0.89	80.29 ± 0.32	83.83 ± 0.27
ARGA	91.72 ± 0.61	90.57 ± 0.51	83.09 ± 1.18	86.13 ± 0.77
ARGVA	92.56 ± 1.38	91.84 ± 1.47	82.77 ± 1.71	85.74 ± 1.77
VGNAE	94.68 ± 0.69	93.87 ± 0.74	77.37 ± 0.10	81.43 ± 0.07
Bagging-PU	95.86 ± 0.20	96.00 ± 0.27	81.25 ± 0.24	85.47 ± 0.10
NESS	96.96 ± 0.08	96.56 ± 0.10	85.91 ± 0.08	87.90 ± 0.12
PULL (ours)	**97.27 ± 0.07**	**97.12 ± 0.10**	**86.46 ± 0.04**	**89.08 ± 0.44**

Missing ratio r_m = 0.2				
Model	**Physics**		**ogbn-arxiv**	
	AUROC	AUPRC	AUROC	AUPRC
GCN+CE	96.60 ± 0.09	96.32 ± 0.11	80.36 ± 0.20	84.99 ± 0.14
GAT+CE	93.55 ± 0.41	92.19 ± 0.49	82.49 ± 0.07	79.64 ± 0.22
SAGE+CE	95.13 ± 0.35	94.67 ± 0.42	83.31 ± 2.03	81.30 ± 1.42
GAE	96.57 ± 0.20	96.31 ± 0.25	80.43 ± 0.44	85.06 ± 0.28
VGAE	94.30 ± 0.59	93.73 ± 0.60	79.38 ± 0.30	83.29 ± 0.26
ARGA	91.75 ± 0.39	90.49 ± 0.52	82.91 ± 0.79	86.04 ± 0.47
ARGVA	92.65 ± 1.19	91.94 ± 1.26	81.57 ± 1.55	84.43 ± 0.90
VGNAE	94.48 ± 0.71	93.69 ± 0.69	76.58 ± 0.15	81.01 ± 0.12
Bagging-PU	95.59 ± 0.12	95.77 ± 0.14	80.84 ± 0.35	85.26 ± 0.22
NESS	96.96 ± 0.08	96.56 ± 0.10	**85.81 ± 0.04**	87.81 ± 0.03
PULL (ours)	**97.01 ± 0.05**	**96.89 ± 0.07**	84.95 ± 0.66	**88.80 ± 0.74**

PULL. We use torch-geometric package to implement the weighted propagation of GCN. The number of inner epochs is set to 200, while that of outer iteration is set to 10. We increase the number K of edges in the approximated version of expected graph $\bar{\mathcal{G}}$ in proportion to that of observed edges through the iterations: $K \leftarrow K + r|\mathcal{E}_\mathcal{P}|$ where r is the increasing ratio. We set $r = 0.05$ in our experiments. For the number M of candidate nodes for generating the candidate edges, we set $M = 100$.

Further Experiments

Link Prediction in Larger Network

We additionally perform link prediction on larger graph datasets compared to those used in Table 1. The ogbn-arxiv dataset is a citation network consisting of 169,343 nodes and 1,166,243 edges, where each node represents an arXiv paper and an edge indicates that one paper cites another one. Each node has 128-dimensional feature vector, which is derived by averaging the embeddings of the words in its title and abstract. Physics is a co-authorship graph based on the Microsoft Academic Graph from the KDD Cup 2016 challenge 3. Physics contains 34,493 nodes and 495,924 edges where each node represents an author, and they are connected if they co-authored a paper. We summarize the statistics of the larger networks in Table 4. For PULL, we set the maximum number of iterations as 20. For the baselines, we set the maximum number of epochs as 4,000. This is because a larger data size requires a greater number of epochs to train the link predictor. PULL incorporates the early stopping with a patience of one, for more stable learning. For other cases, we used the same settings as in the Experiments section.

Table 5 presents the link prediction performance of PULL and the baselines in ogbn-arxiv and Physics. Note that PULL consistently shows superior performance than the baselines in terms of both AUROC and AUPRC. This indicates that PULL is also effective in handling larger graphs.

Table 6: The link prediction accuracy of PULL and its variant PULL-WS. PULL-WS is PULL that approximates $\bar{\mathcal{G}}$ by performing weighted random sampling of edges based on the linking probabilities. Bold numbers denote the best performance. PULL outperforms PULL-WS in every case.

Missing ratio r_m = 0.1				
Dataset	**PULL-WS**		**PULL (proposed)**	
	AUROC	AUPRC	AUROC	AUPRC
PubMed	96.54 ± 0.18	96.80 ± 0.14	**96.59 ± 0.19**	**96.83 ± 0.18**
Cora-full	95.94 ± 0.31	96.12 ± 0.32	**96.06 ± 0.34**	**96.25 ± 0.35**
Chameleon	97.69 ± 0.28	97.68 ± 0.28	**97.87 ± 0.33**	**97.83 ± 0.33**
Crocodile	97.38 ± 0.31	97.66 ± 0.26	**98.31 ± 0.20**	**98.36 ± 0.22**
Facebook	97.05 ± 0.15	97.30 ± 0.14	**97.41 ± 0.11**	**97.67 ± 0.09**

Missing ratio r_m = 0.2				
Dataset	**PULL-WS**		**PULL (proposed)**	
	AUROC	AUPRC	AUROC	AUPRC
PubMed	96.24 ± 0.14	96.43 ± 0.14	**96.28 ± 0.13**	**96.47 ± 0.17**
Cora-full	95.31 ± 0.35	95.62 ± 0.33	**95.39 ± 0.32**	**95.65 ± 0.31**
Chameleon	97.66 ± 0.18	97.65 ± 0.18	**97.89 ± 0.14**	**97.87 ± 0.16**
Crocodile	97.28 ± 0.22	97.57 ± 0.21	**98.19 ± 0.13**	**98.29 ± 0.16**
Facebook	96.95 ± 0.10	97.20 ± 0.09	**97.30 ± 0.07**	**97.59 ± 0.06**

Weighted Random Sampling for Constructing $\bar{\mathcal{G}}'$

PULL keeps the top-K edges with the highest linking probabilities to approximate $\bar{\mathcal{G}}$. In this section, we compare PULL with PULL-WS (PULL with Weighted Sampling) that constructs the approximated version $\bar{\mathcal{G}}'$ by performing weighted random sampling of edges from $\bar{\mathcal{G}}$ based on the linking probabilities. As the weighted random sampling empowers PULL to mitigate the excessive self-reinforcement in the link predictor, we additionally exclude the loss term $\mathcal{L}_C$, which serves the same purpose. We conduct experiments five times with random seeds, while using the same experimental settings as in the Experiments section.

Table 6 shows that PULL-WS presents marginal performance decrease compared to PULL. This indicates that keeping the top-K edges having the highest linking probabilities with an additional loss term $\mathcal{L}_C$ shows better link prediction performance than performing weighted random sampling of edges without $\mathcal{L}_C$. However, PULL-WS is an efficient variant of PULL that uses only a single loss term $\mathcal{L}'_E$ instead of the proposed loss $\mathcal{L} = \mathcal{L}'_E + \mathcal{L}_C$.

Performance with other ratio r_m of missing edges

In the experiment section, we demonstrated the superior performance of PULL with a test missing edge ratio $r_m = 0.1$. To further validate the robustness of PULL across varying

Table 7: 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 = 0.2										
Model	**PubMed**		**Cora-full**		**Chameleon**		**Crocodile**		**Facebook**	
	AUROC	AUPRC	AUROC	AUPRC	AUROC	AUPRC	AUROC	AUPRC	AUROC	AUPRC
GCN+CE	$\underline{96.14 \pm 0.19}$	$\underline{96.25 \pm 0.21}$	94.92 ± 0.64	95.01 ± 0.75	96.85 ± 0.36	96.78 ± 0.45	97.06 ± 0.46	97.37 ± 0.40	$\underline{97.00 \pm 0.23}$	97.26 ± 0.22
GAT+CE	90.67 ± 0.37	89.32 ± 0.45	93.99 ± 0.35	93.47 ± 0.40	91.75 ± 1.65	91.28 ± 1.42	91.09 ± 1.50	91.80 ± 1.09	92.41 ± 0.48	92.19 ± 0.56
SAGE+CE	85.90 ± 0.67	87.22 ± 0.93	93.71 ± 0.60	94.25 ± 0.66	96.11 ± 0.51	95.68 ± 0.63	95.92 ± 0.67	96.48 ± 0.62	94.96 ± 0.46	95.06 ± 0.55
GAE	96.10 ± 0.15	96.22 ± 0.21	95.15 ± 0.39	95.24 ± 0.48	96.76 ± 0.42	96.60 ± 0.57	96.36 ± 0.65	96.74 ± 0.56	96.87 ± 0.38	97.12 ± 0.37
VGAE	94.12 ± 1.13	94.17 ± 1.10	91.71 ± 3.94	91.73 ± 3.72	96.21 ± 0.22	96.01 ± 0.32	95.21 ± 0.45	95.40 ± 0.86	95.89 ± 0.54	96.11 ± 0.52
ARGA	93.00 ± 0.58	92.43 ± 0.54	90.93 ± 0.62	90.40 ± 0.63	94.72 ± 0.34	94.37 ± 0.41	95.98 ± 0.47	95.63 ± 0.39	91.90 ± 0.51	91.98 ± 0.46
ARGVA	93.19 ± 1.30	93.38 ± 1.17	87.56 ± 4.49	87.53 ± 4.21	94.07 ± 0.51	94.09 ± 0.40	94.85 ± 0.14	94.00 ± 0.15	92.68 ± 1.82	93.11 ± 1.69
VGNAE	95.70 ± 0.39	95.62 ± 0.38	$\underline{95.40 \pm 1.04}$	95.13 ± 1.06	97.45 ± 0.30	$\underline{97.13 \pm 0.35}$	96.41 ± 0.77	95.91 ± 1.36	95.22 ± 0.88	95.33 ± 0.87
Bagging-PU	94.02 ± 0.34	94.38 ± 0.41	92.56 ± 0.54	94.48 ± 0.67	$\underline{97.13 \pm 0.47}$	97.08 ± 0.54	$\underline{97.48 \pm 0.41}$	$\underline{97.79 \pm 0.37}$	96.95 ± 0.21	$\underline{97.31 \pm 0.21}$
NESS	95.62 ± 0.04	95.45 ± 0.03	**96.07 ± 0.24**	**96.09 ± 0.27**	96.68 ± 0.29	96.64 ± 0.29	96.07 ± 0.11	96.78 ± 0.03	95.79 ± 0.06	96.20 ± 0.03
PULL (proposed)	**96.28 ± 0.13**	**96.47 ± 0.17**	95.39 ± 0.32	$\underline{95.65 \pm 0.31}$	**97.89 ± 0.14**	**97.87 ± 0.16**	**98.19 ± 0.13**	**98.29 ± 0.16**	**97.30 ± 0.07**	**97.59 ± 0.06**

Table 8: 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 = 0.2										
Model	**PubMed**		**Cora-full**		**Chameleon**		**Crocodile**		**Facebook**	
	AUROC	AUPRC	AUROC	AUPRC	AUROC	AUPRC	AUROC	AUPRC	AUROC	AUPRC
GAE	96.10 ± 0.15	96.22 ± 0.21	95.15 ± 0.39	95.24 ± 0.48	96.76 ± 0.42	96.60 ± 0.57	96.36 ± 0.65	96.74 ± 0.56	96.87 ± 0.38	97.12 ± 0.37
GAE+PULL	**96.23 ± 0.10**	**96.47 ± 0.12**	**95.44 ± 0.41**	**95.69 ± 0.51**	**98.00 ± 0.15**	**98.03 ± 0.15**	**98.18 ± 0.19**	**98.31 ± 0.17**	**97.26 ± 0.12**	**97.53 ± 0.12**
VGAE	94.12 ± 1.13	94.17 ± 1.10	91.71 ± 3.94	91.73 ± 3.72	96.21 ± 0.22	96.01 ± 0.32	95.21 ± 0.45	95.40 ± 0.86	95.89 ± 0.54	96.11 ± 0.52
VGAE+PULL	**95.30 ± 0.65**	**95.35 ± 0.66**	**93.19 ± 3.46**	**93.37 ± 3.30**	**96.97 ± 0.56**	**96.97 ± 0.63**	**97.24 ± 0.67**	**97.44 ± 0.54**	**96.51 ± 0.23**	**96.67 ± 0.23**
VGNAE	95.70 ± 0.39	95.62 ± 0.38	95.40 ± 1.04	95.13 ± 1.06	97.45 ± 0.30	97.13 ± 0.35	96.41 ± 0.77	95.91 ± 1.36	95.22 ± 0.88	95.33 ± 0.87
VGNAE+PULL	**95.84 ± 0.31**	**95.74 ± 0.26**	**95.65 ± 0.70**	**95.42 ± 0.73**	**97.70 ± 0.31**	**97.36 ± 0.36**	**96.67 ± 1.32**	**96.13 ± 2.18**	**95.72 ± 0.44**	**95.78 ± 0.41**

levels of missing lines, we conducted additional experiments with a larger ratio, $r_m = 0.2$.

The results in Table 7 highlight that PULL consistently achieves the best link prediction accuracy compared to other baselines. Furthermore, as shown in Table 8, the integration of PULL into existing baseline methods significantly enhances their performance, underscoring the adaptability and robustness of PULL across different levels of missing data.

References

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

Fey, M.; and Lenssen, J. E. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *International Conference on Learning Representations (ICLR) Workshop on Representation Learning on Graphs and Manifolds*.

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.

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

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.

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, D.; Cui, P.; and Zhu, W. 2016. Structural deep network embedding. In *SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1225–1234.