A Probabilistic Circuit Framework for Interpretable Graph PU Learning

Sagad Hamid[1] **Dooho Lee**[2] **Myeong Kong**[2] **Tanya Braun**[1] **Jaemin Yoo**[3]

[1]Computer Science Department, University of Münster, Münster, Germany
[2]School of Electrical Engineering, KAIST, Daejeon, Republic of Korea
[3]Department of Computer Science and Engineering, Seoul National University, Seoul, Republic of Korea

Abstract

How can we make graph positive-unlabeled (PU) learning interpretable? Existing methods jointly process node features and edge information, which obscures their interaction and makes predictions difficult to interpret. In this paper, we propose a novel interpretable graph PU learning framework that explicitly decouples feature and edge processing, enabling multi-level interpretability. Our framework first produces a core prediction from node features using probabilistic circuits (PCs) and then refines it using edge information, providing *graph-level interpretability* by exposing how graph structure affects predictions. For feature-based prediction, we construct two PCs through a careful split of the training nodes, yielding *node-level interpretability* by highlighting which nodes support the separation of positive and negative instances. Finally, by leveraging the tractability of PCs, we obtain *feature-level interpretability* via feature attribute marginalization, which quantifies attribute impact and importance while revealing interactions and dependencies. Experiments on 10 datasets show that our framework achieves strong performance while substantially improving interpretability. The code is available at `https://github.com/hagad1/graphpu-cpu`.

1 INTRODUCTION

Graph-structured data are prevalent across many fields, such as social networks, recommendation systems, and natural sciences [Kipf and Welling, 2017, Ying et al., 2018, Fan et al., 2019, Fout et al., 2017, Sanchez-Gonzalez et al., 2020, Wu et al., 2019]. Their growing importance has led to the adaptation of numerous machine learning tasks to graphs [Hu et al., 2020, Gilmer et al., 2017, Zhang et al., 2023, Yun et al., 2021, Vignac et al., 2020, Shirzad et al., 2023, Yun et al., 2019]. At the same time, the structural diversity of real-world graphs makes this adaptation challenging, as models must account for both node features and edge information, as well as topological properties such as homophily and heterophily [Wu et al., 2020, Zhu et al., 2020, Xia et al., 2021, Zhou et al., 2020, Ma et al., 2022, Zheng et al., 2026, Li et al., 2022, Platonov et al., 2023b]. These challenges become particularly important in weakly supervised settings such as *graph positive-unlabeled (PU) learning*, a node classification task in which only a subset of positive nodes is labeled and all remaining nodes are unlabeled [Yoo et al., 2021, Wu et al., 2024, Kiryo et al., 2017, Garg et al., 2021]. Since no negative labels are available, existing graph PU methods typically rely on estimating the class prior of the positive class and then assigning pseudo-labels to unlabeled nodes, which are subsequently used to train a graph neural network (GNN) as a binary classifier.

A key difficulty in this setting is that prior estimation and pseudo-label generation usually process feature and edge information jointly. While effective, this tightly coupled processing obscures how node attributes and graph structure interact, making the resulting predictions difficult to interpret. This lack of transparency is especially problematic in graph PU learning, where understanding relationships among positive nodes is crucial for separating them from negative ones. Moreover, enhanced interpretability facilitates identifying outliers and misleading connections, thereby strengthening robustness and classification performance.

To address these issues, we propose a novel interpretable framework for graph PU learning that provides *multi-level interpretability* while remaining effective for downstream classification. Our main contributions are as follows.

C1. Decoupling Features And Edges. We explicitly separate feature and edge processing to decouple their contributions. Concretely, our approach first generates predictions solely from node features and subsequently refines these predictions through an edge-based propagation mechanism. The

Proceedings of the 42nd Conference on Uncertainty in Artificial Intelligence (UAI 2026), PMLR 337:1913–1930.

resulting refined predictions can either be used directly as final outputs or as pseudo-labels to train a GNN as a binary classifier. Our decoupling provides *graph-level interpretability*, revealing how edges affect feature-based predictions. In particular, it enables explicit analysis, control, and regulation of feature–edge interactions. By generating pseudo-labels, our approach provides additional insight into which signals are advantageous to the learning process and which may adversely affect model training.

C2. Probabilistic Circuits As Feature Predictors. Effective reasoning over features requires a model that balances expressiveness, tractability, and interpretability. Probabilistic circuits (PCs) are suited for this purpose, as they provide a circuit-based representation of complex probability distributions and support efficient inference for a wide range of probabilistic queries [Peharz et al., 2020b, Dang et al., 2020, Peharz et al., 2020a, Vergari et al., 2021, Liu and Van den Broeck, 2021, Sidheekh and Natarajan, 2024]. Building on this, we propose *Probabilistic **C**ircuits for Graph **PU** Learning (CPU)* as the core feature-based predictor. We model feature attributes as random variables (RVs) and employ two PCs to estimate feature-based densities P_P and P_N for positive and negative nodes, respectively. Since negative labels are missing, we construct P_N as a contrastive counterpart to P_P by carefully splitting the node set for the respective training of each PC. We then derive predictions by evaluating the log-likelihood ratio between the two models. CPU provides *node-level interpretability* as modeling densities trained on two node sets highlights which nodes support separating positive instances from negative ones.

C3. Feature Attribute Marginalization. Beyond node-level explanations, we leverage the tractability of PCs to obtain *feature-level interpretability*. In contrast to existing methods that typically use all feature attributes without assessing their individual influence, we compute marginal log-likelihoods by selectively marginalizing feature attributes. This allows us to quantify feature attribute impact and importance and to analyze dependencies and interactions among attributes, thereby clarifying which features drive the predictions.

Our experiments show that our framework significantly outperforms existing methods on many heterophilic graphs, advancing the state of the art across various datasets. Moreover, it provides an effective means to generate pseudo-labels for training highly competitive GNNs, matching or surpassing iterative methods that optimize more complex objective functions [Yoo et al., 2021, Wu et al., 2024]. Overall, our results show that strong predictive performance and multi-level interpretability, namely on graph-, node-, and feature-levels, can be achieved jointly in graph PU learning.

The remainder of this work is organized as follows. Section 2 introduces notations, preliminaries, and related work on graph PU learning and PCs. Our interpretable graph PU learning framework is presented in Section 3. Experimental results are shown in Section 4, followed by a discussion. We conclude our work in Section 5 with potential future work.

2 PRELIMINARIES & RELATED WORK

Notation We denote RVs by uppercase letters X, their domain by $\mathrm{Dom}(X)$, and their values by lowercase letters $x \in \mathrm{Dom}(X)$. Boldface uppercase letters $\boldsymbol{X}$ and boldface lowercase letters $\boldsymbol{x}$ denote sets of RVs and their values, respectively. For a graph $G = (\mathcal{V}, \mathcal{E})$ with nodes $\mathcal{V}$ and edges $\mathcal{E}$, we denote the adjacency matrix by $\mathcal{A} \in \{0,1\}^{|\mathcal{V}|\times|\mathcal{V}|}$, where $a_{ij} = 1$ iff $(\nu_i, \nu_j) \in \mathcal{E}$, and $a_{ij} = 0$ otherwise. We consider undirected graphs, i.e., $(\nu_i, v_j) = (\nu_j, \nu_i)$. Each node ν_i is characterized by a d-dimensional feature vector, yielding a feature matrix $\mathcal{X} \in \mathbb{R}^{|\mathcal{V}|\times d}$. We denote by $F_{\mathcal{S}} = \{\boldsymbol{x}_i \mid \nu_i \in \mathcal{S}\}$ the features of nodes $\mathcal{S} \subseteq \mathcal{V}$, where $\boldsymbol{x}_i$ is the feature assignment of ν_i.

2.1 GRAPH NEURAL NETWORKS

Graph neural networks (GNNs) are a class of neural networks specifically designed for graph-structured data. They operate by iteratively updating node representations based on information aggregated from neighboring nodes. This is achieved through message-passing mechanisms, which combine local neighborhood information with learnable transformation functions. Formally, a GNN $f(\mathcal{A}, \mathcal{X})$ with $\boldsymbol{H}^0 = \mathcal{X}$ and L layers processes the graph data in layer l as follows [Luo et al., 2024]:

$$\boldsymbol{H}_v^l = \mathrm{UPDATE}^l(\boldsymbol{H}_v^{l-1}, \mathrm{AGG}^l(\{\boldsymbol{H}_u^{l-1} \mid u \in \mathrm{N}(v)\})),$$

where $\mathrm{N}(v)$ represents the neighbors of v, AGG^l represents a message aggregation function, and UPDATE^l an update function. The last layer $\boldsymbol{H}^L = f(\mathcal{A}, \mathcal{X})$ yields the output of the GNN. Despite their strengths, GNNs require careful consideration when applied to graphs that yield different types of structural properties.

Homophilic & Heterophilic Graphs The homophily or heterophily ratio within a graph significantly affects the effectiveness of GNN models [Platonov et al., 2023a, Luan et al., 2022]. Homophilic graphs are characterized by edges that mostly connect nodes of the same class. This structure aligns well with message-passing frameworks, which implicitly assume that information from neighboring nodes reinforces a node's own representation. As a result, classic GNNs perform well in such settings by leveraging this local consistency. In contrast, heterophilic graphs exhibit connections between nodes of different classes. This structural property undermines the assumptions of homophily, often causing message passing to propagate misleading signals.

2.2 GRAPH POSITIVE-UNLABELED LEARNING

Graph positive-unlabeled (PU) learning is a weakly supervised binary node classification task that can be formulated as follows:

Definition 2.1 (Graph PU Learning [Yoo et al., 2021])**.** *Let $G = (\mathcal{V}, \mathcal{E})$ be a graph. Let $\mathcal{P}$ be a set of observed positive-labeled nodes and $\mathcal{U} = \mathcal{V} \setminus \mathcal{P}$ be the remaining, unlabeled nodes. The goal of graph PU learning is to train a binary classifier that accurately predicts the unknown labels of $\mathcal{U}$.*

The key challenge is that only positively labeled nodes are observed. In the absence of negative labels, standard supervised objective functions for GNNs are inapplicable. To address this, recent works have reformulated the learning objective by means of a mixture model:

$$\mathbb{P}(\mathcal{X}, \mathcal{A}) = \pi_P \mathbb{P}_P(\mathcal{X}, \mathcal{A}) + (1 - \pi_P)\mathbb{P}_N(\mathcal{X}, \mathcal{A}),$$

where $\mathbb{P}(\mathcal{X}, \mathcal{A})$ is the unconditional node distribution, and $\mathbb{P}_P(\mathcal{X}, \mathcal{A})$ and $\mathbb{P}_N(\mathcal{X}, \mathcal{A})$ are class-conditional distributions for positive and negative nodes, respectively.

The positive class prior π_P represents the fraction of positive nodes among the unlabeled nodes. However, it is usually not expected that π_P is known since no information on negative labels is given. Thus, recent approaches have focused on estimating the true class prior π_P utilizing the graph structure and positive labeled nodes. GRAB [Yoo et al., 2021] addresses graph PU learning by modeling the graph as a Markov network, i.e., an undirected probabilistic graphical model, and iteratively estimates the unknown class prior through belief propagation, alternating between marginal probability estimation and classifier training. GPL [Wu et al., 2024] further improves graph PU learning on graphs with heterophilic structures. It introduces a label propagation loss to reduce heterophily by strengthening homophilic edges and weakening heterophilic ones, using a bilevel optimization: the inner loop optimizes the graph structure reducing heterophily, while the outer loop trains a classifier on the adjusted graph. This mitigates the violation of irreducibility assumptions in class-prior estimation.

2.3 PROBABILISTIC CIRCUITS

PCs are a class of circuit-based tractable probabilistic models designed to represent and reason about complex probability distributions. In contrast to many deep generative models, PCs offer direct access to and modification of the underlying probability distribution, effectively bridging expressiveness, tractability, and interpretability [Kingma and Welling, 2014, Goodfellow et al., 2014, Rezende et al., 2014, Tan and Peharz, 2019, Shih and Ermon, 2020, Peharz et al., 2020b, Papamakarios et al., 2021, Liu and Van den Broeck, 2021, Liu et al., 2023, Sidheekh and Natarajan, 2024]. Formally, a PC $P(\boldsymbol{X})$ defined over RVs $\boldsymbol{X}$ is a directed acyclic graph where each node n encodes a distribution

$$P_n(\boldsymbol{x}_n) = \begin{cases} p_n(\boldsymbol{x}_n), & \text{if } n \text{ is a leaf node} \\ \sum_{c \in \mathrm{ch}(n)} \theta_{c|n} P_c(\boldsymbol{x}_c), & \text{if } n \text{ is a sum node} \\ \prod_{c \in \mathrm{ch}(n)} P_c(\boldsymbol{x}_c), & \text{if } n \text{ is a prod. node,} \end{cases}$$

where $\mathrm{ch}(n)$ denotes the child nodes of node n, and $\theta_{c|n}$ defines a weight parameter for the edge between sum node n and its child node c. For leaves, $p_n(\boldsymbol{x}_n)$ is a tractable probability distribution (mostly categorical distributions or univariate Gaussians) defined over the RV scope $\phi(n) = \boldsymbol{X}_n \subseteq \boldsymbol{X}$. To evaluate $p(\boldsymbol{x})$ for a given sample $\boldsymbol{x} \in \mathrm{Dom}(\boldsymbol{X})$, the values are propagated from the leaf nodes to the root node n_r, usually a sum node that encodes the final output. PCs allow for tractably computing a wide range of probabilistic queries when structural constraints are imposed.

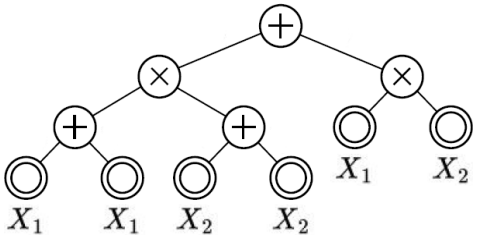

Figure 1: An exemplary decomposable and smooth PC.

Typically, the structure of PCs is learned from data using algorithms designed explicitly to ensure that these constraints are met. Two fundamental structural properties are *decomposability* and *smoothness*, which enable tractable & exact (conditional) marginal queries [Choi et al., 2020].

Definition 2.2 (Decomposability and Smoothness [Liu et al., 2023])**.** *A PC is **smooth** if for a sum node n_s, all its children have the same scope: $\forall c_1, c_2 \in \mathrm{ch}(n_\mathrm{s}), \phi(c_1) = \phi(c_2)$. A PC is **decomposable** if the children of a product node n_p have disjoint scopes: $\forall c_1, c_2 \in \mathrm{ch}(n_\mathrm{p})$ with $c_1 \neq c2$: $\phi(c_1) \cap \phi(c_2) = \emptyset$.*

An exemplary decomposable and smooth PC representing a distribution $P(X_1, X_2)$ over two RVs X_1, X_2 is illustrated in Fig. 1 (weight parameters are omitted). In particular, decomposability ensures that product nodes represent fully factorized distributions by enforcing that their input distributions do not have common RVs, while smoothness ensures that sum nodes represent valid probabilistic mixtures by enforcing their inputs to share the same RVs.

3 PROPOSED METHOD

We introduce a probabilistic framework for interpretable and modular graph PU learning. The key idea is to explicitly decouple feature and edge processing. First, we perform

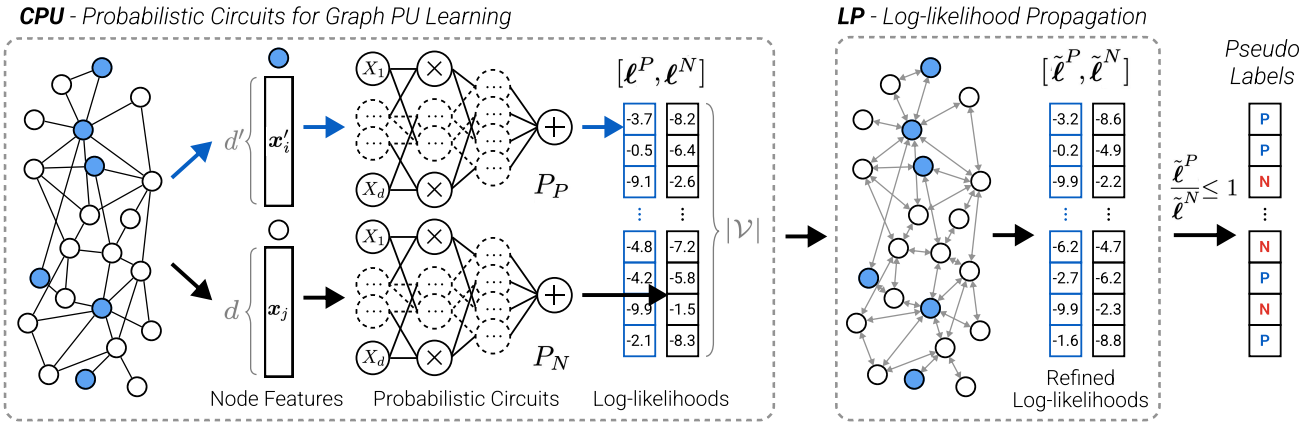

Figure 2: Overview of our interpretable PU learning framework, which decouples feature and edge processing. CPU first trains PCs on a specific node split; each PC takes a carefully-selected subset of nodes. Node features are then evaluated to compute log-likelihoods, which are subsequently refined via LP. The log-likelihood ratio is then used to derive pseudo-labels.

probabilistic predictions based solely on node features; subsequently, we refine these predictions through an edge-based refinement step using log-likelihood propagation (LP) (Section 3.1). At the core of the feature-based model, we model two separate PCs to estimate the densities of positive and negative nodes, respectively. Predictions are obtained by evaluating the corresponding log-likelihood ratio for each node feature. This dual-PC design yields interpretability at both node- and feature-levels (Section 3.2). Specifically, we partition the nodes into two subsets, each used to train one of the PCs. This separation allows for controlling the influence of different node subsets on density estimation. The tractability of PCs further enables efficient marginalization over feature attributes, providing feature-level interpretability. An overview of the framework is presented in Fig. 2.

3.1 INTERPRETABLE FRAMEWORK BASED ON FEATURE-EDGE DECOUPLING

Separating feature and edge processing allows us to analyze their individual effects and also interactions; it facilitates understanding of how well feature-only predictions perform, whether incorporating edge information improves or degrades performance, and how structural information contributes overall to the predictive process.

Given that graph PU learning is a weakly supervised problem characterized by substantial uncertainty, we employ a probabilistic feature-based predictor as the core modeling component. To accurately capture dependencies among feature attributes in $\mathcal{X}$, we construct probabilistic models over a set of discrete RVs $\boldsymbol{X} = \{X_1, \ldots, X_d\}$, where each RV X_i corresponds to the i-th feature attribute and takes values in $\mathrm{Dom}(X_i)$. To further enhance modeling flexibility, we apply simple binning strategies to discretize continuous features at varying levels of granularity. More specifically, for a given threshold ϵ, we discretize the features via $\lfloor \frac{x_i}{\epsilon} \rfloor$, allowing us to explicitly control and regulate the domain size $|\mathrm{Dom}(X_i)|$ of each feature attribute and thus the modeling granularity for the probabilistic models.

Log-Likelihood Ratio of Two Probabilistic Models Instead of relying on a single probabilistic model, we evaluate node features using two separate models, $P_P(\boldsymbol{X})$ and $P_N(\boldsymbol{X})$, which estimate the feature-based densities of positive and negative nodes, respectively. Because explicit negative labels are unavailable and independence assumptions may be violated due to structural dependencies in graph data, we construct P_N as a contrastive counterpart to P_P. More specifically, we first train P_P and compute the log-likelihoods $\boldsymbol{\ell}^P = \left[P_P(\boldsymbol{x}_i)\right]_{\boldsymbol{x}_i \in F_{\mathcal{V}}}$ for all nodes. The log-likelihoods are then used to identify suitable samples for training P_N, yielding corresponding log-likelihoods $\boldsymbol{\ell}^N = \left[P_N(\boldsymbol{x}_i)\right]_{\boldsymbol{x}_i \in F_{\mathcal{V}}}$. This dual-model strategy enables a node-level split that facilitates effective training of P_P and P_N on selected subsets of nodes, improving the estimation of the underlying class-conditional densities. In particular, we provide flexibility w.r.t model choice, training data, and parameterization, particularly enabling controlled and class-specific training of the respective models.

The final pseudo-labels for unlabeled nodes $\mathcal{U}$ based on the log-likelihood ratio are derived as follows:

$$\hat{y}^F(\boldsymbol{x}_u) = \begin{cases} 1 & \text{if } \frac{\ell_u^P}{\ell_u^N} \leq 1, \\ -1 & \text{otherwise}, \end{cases} \tag{1}$$

where $\hat{y}^F(\boldsymbol{x}_u)$ denotes the feature-based predicted label for $\boldsymbol{x}_u \in F_{\mathcal{U}}$. The ratio assigns an unlabeled node a positive label if $\ell_u^P \geq \ell_u^N$ and a negative label otherwise.

Node Splitting To train P_P and P_N for feature-based density estimation, a central challenge is selecting suitable training data which requires splitting the nodes. It is crucial to choose the splitting such that positive and negative

nodes can be separated effectively through the respective log-likelihoods. To address this, we propose the following node splitting strategies:

S1. $\mathcal{P}$ *vs.* $\mathcal{U}$: Using all positively labeled nodes $\mathcal{P}$ for P_P and all unlabeled nodes $\mathcal{U}$ for P_N offers a simple yet effective splitting, particularly when unlabeled features are sufficiently distinct from those of the positive ones. However, this approach may lead to imbalances when the sizes of $\mathcal{P}$ and $\mathcal{U}$ differ significantly.

S2. $\mathcal{P}$ *vs. worst-*$k\%$ $\boldsymbol{\ell}^P$: To provide a more conservative selection of nodes for P_N, we select the $k\%$ of nodes with the lowest log-likelihoods in P_P, providing a data-driven method to select the nodes based on positive features $F_{\mathcal{P}}$. While this approach generally allows for regulating the number of nodes to train P_N for more balanced models, it also accounts for potential outliers in F_P.

S3. $\mathcal{P}$ *vs. worst-*$k_{\mathcal{U}}\%$ $\boldsymbol{\ell}^P$: This strategy is similar to *S2*, but it avoids using the nodes in $\mathcal{P}$ for constructing P_N by selecting among the worst $k\%$ log-likelihoods only unlabeled nodes from $\mathcal{U}$, making an even more conservative selection.

S4. $\mathcal{V}$ *vs.* $\mathcal{U}$: If $F_{\mathcal{P}}$ and $F_{\mathcal{U}}$ have similar features, it can be useful to use all available features $\mathcal{F}_{\mathcal{V}}$ for P_P to provide more training data while explicitly strengthening the positive log-likelihoods since they are excluded in P_N.

Edge Refinement Via Log-Likelihood Propagation Adding structural information in a separate step enables clearer analysis of how features and edges interact and individually contribute to prediction performance. This is particularly valuable when handling graphs with varying degrees of heterophily. Inspired by label propagation [Wu et al., 2024], we adopt a simple yet effective technique to refine feature-based predictions with structural information. Our approach remains transparent in how edge information affects the feature-based predictions, yet flexible enough to regulate its influence among diverse graph structures. Given the feature-based log-likelihoods $L = [\boldsymbol{\ell}^N, \boldsymbol{\ell}^P]$ from the probabilistic models, we propagate them as follows:

$$\tilde{L}^m = (1-\alpha)\tilde{L}^{m-1} + \alpha \mathcal{D}^{-1}\mathcal{A}\tilde{L}^{m-1},$$

where $\mathcal{D}$ is the degree matrix and $\tilde{L}^0 = L$. Isolated nodes, i.e., nodes with degree 0, are excluded from the propagation and keep their feature-based log-likelihoods. The parameter m provides a general means to incorporate and strengthen multi-hop message passing, while $\alpha \in [0, 1]$ controls the influence of neighboring nodes. These parameters particularly allow for adapting LP to various graph datasets, as structural information tends to be more beneficial for homophilic graphs and less so for heterophilic ones.

For the final prediction, we train a GNN classifier and use the log-likelihood ratio w.r.t. the refined log-likelihoods $\tilde{L} = [\tilde{\ell}^N, \tilde{\ell}^P]$ to predict $\hat{\boldsymbol{y}}$ according to Eq. (1) to set pseudo-labels $\hat{\boldsymbol{y}}_{\text{pseudo}} = [\hat{y}_{\text{pseudo}}(\boldsymbol{x}_i)]_{\boldsymbol{x}_i \in F_{\mathcal{V}}}$ as follows:

$$\hat{y}_{\text{pseudo}}(\boldsymbol{x}_i) = \begin{cases} \hat{y}(\boldsymbol{x}_i) & \text{if } \boldsymbol{x}_i \in F_{\mathcal{U}} \\ +1 & \text{if } \boldsymbol{x}_i \in F_{\mathcal{P}}. \end{cases}$$

We use the pseudo-labels $\hat{\boldsymbol{y}}_{\text{pseudo}}$ as the training objective of the GNN to minimize the standard cross-entropy loss. Figure 2 depicts the decoupled procedure for pseudo-label generation, where CPU is our specific instance of the feature-based prediction method discussed next.

3.2 CPU: USING PROBABILISTIC CIRCUITS AS FEATURE PREDICTORS

Various generative probabilistic models can serve as feature-based predictors. However, many of these models do not offer direct access to the underlying probability distribution, making it difficult to perform advanced probabilistic queries, such as marginal queries, in a tractable way [Choi et al., 2020]. We introduce CPU as our core feature-based predictor, leveraging PCs to meet the following conditions:

Condition 1 (Data-Driven Estimation)**.** *Given a set of training nodes $\mathcal{S} \subseteq \mathcal{V}$ with corresponding features $F_{\mathcal{S}}$, any probabilistic model $P(\boldsymbol{X})$ shall be estimated in a data-driven manner by maximizing the log-likelihood $\sum_{\boldsymbol{x}_s \in F_{\mathcal{S}}} \ell_s$.*

Condition 2 (Tractability)**.** *Any model P should be able to compute the marginal log-likelihood ℓ_i over RVs $\boldsymbol{X}' \subseteq \boldsymbol{X}$ exactly for any feature assignment $\boldsymbol{x}'_i \in \text{Dom}(\boldsymbol{X}')$, in time polynomial in the size of the model P.*

Condition 1 particularly ensures that P_P effectively captures the underlying distribution of positively labeled data $F_{\mathcal{P}}$. Specifically, by including $\mathcal{P}$ to train P_P, we increase the model's confidence in feature assignments that resemble those of positively labeled nodes. While many existing models can satisfy Condition 1, our goal is to analogously satisfy Condition 2. Traditional probabilistic models often suffer from exponential complexity with respect to the number of RVs $d = |\boldsymbol{X}|$, typically relying on approximate inference to maintain tractability. To support exact and tractable inference that satisfies both Conditions 1 and 2, we employ structured PCs that balance expressivity, tractability, and interpretability [Sidheekh and Natarajan, 2024]. For the backbone PC structure, we use Hidden-Chow-Liu-Trees (HCLTs) [Liu and Van den Broeck, 2021].

Hidden-Chow-Liu-Tree (HCLT) The HCLT approach constructs a PC starting with a Chow-Liu Tree (CLT), a tree-structured Bayesian network that models dependencies among feature RVs by maximizing pairwise mutual information. A CLT captures key interactions between feature attributes while encoding conditional independencies within its tree structure, offering insights into the (in)dependencies among RVs. HCLT extends this framework by introducing

a latent RV for each feature attribute RV $X_i \in \boldsymbol{X}$. The resulting structure ensures is smooth and decomposable [Liu and Van den Broeck, 2021].

Once the PC structure is built, PC parameters are usually learned using the Expectation-Minimization (EM) algorithm in terms of PC flows. Simply put, PC flows capture which edges are activated during log-likelihood evaluation $\ell_i = P(\boldsymbol{x}_i)$. The learning process adjusts parameters in a data-driven way s.t. edges with higher cumulative flows are strengthened. We refer to [Choi et al., 2021, Liu et al., 2024] for more details on the explicit calclulation of PC flows.

Lemma 3.1. *Employing HCLT-structured PCs P_P, P_N, combined with parameter learning via the EM algorithm satisfies both Conditions 1 and 2.*

Proof. Condition 2 holds due to the structural properties of HCLTs, namely smoothness and decomposability, which ensure exact and tractable inference of marginal queries [Choi et al., 2020]. Condition 1 is satisfied using the EM algorithm, which performs maximum-likelihood estimation to maximize the log-likelihood of the training data. □

Feature Attribute Marginalization The previously described log-likelihood evaluation in CPU utilizes all available node feature attributes $\boldsymbol{X}$ to train P_P. However, to obtain deeper insights into how individual feature attributes influence the predictions, we further leverage the tractability of PCs by explicitly incorporating marginalization over selected RVs which correspond to specific feature attributes. More precisely, we train the positive PC $P_P(\boldsymbol{x})$ on the full set of feature attributes in order to retain relevant dependencies and interactions. During inference, however, we then evaluate log-likelihoods only on a subset of features $\boldsymbol{X}' \subseteq \boldsymbol{X}$ of dimension d', i.e., we query marginals $P_P(\boldsymbol{x}'_i)$ with $\boldsymbol{x}'_i \in \mathrm{Dom}(\boldsymbol{X}')$. This effectively enables reducing the features considered during log-likelihood evaluation, mitigating the influence of noisy or less informative features that may interfere the separation of positive and negative nodes. In addition, this procedure enables assessing the contribution and importance of individual feature attributes.

Generally, any statistics can be used to determine the subset $\boldsymbol{X}'$. In our approach, we select the attributes based on the variance (before discretization) among positive node features F_{P}, the sole source of data. The variance provides a simple measure of how consistently a feature attribute characterizes the positive nodes. While low-variance attributes capture properties shared across positive instances, a high variance may indicate attributes which are less representative of a positive node. Thus marginalizing the latter allows for reducing the influence of potentially noisy attributes. We therefore first determine the variances

$$\mathrm{vars}(F_{\mathcal{P}}) = \{\mathrm{var}(\boldsymbol{x}_{i,j} \mid \boldsymbol{x}_i \in F_{\mathcal{P}})\}_{j=1}^{d},$$

where $\boldsymbol{x}_{i,j}$ is the $j-$th attribute of the feature $\boldsymbol{x}_i$. Then we marginalize out those attributes $\boldsymbol{X}_{\mathrm{mar}}$ that yield the $k\%$ highest variances, i.e.,

$$P_P(\boldsymbol{X}') = \sum_{\boldsymbol{X}_{\mathrm{mar}}} P_P(\boldsymbol{X}), \text{ with } \boldsymbol{X}' = \boldsymbol{X} \setminus \boldsymbol{X}_{\mathrm{mar}},$$

which can be evaluated for $\boldsymbol{x}'_i \in \mathrm{Dom}(\boldsymbol{X}')$ in a tractable manner by Lemma 3.1. We only perform marginalization for P_P as it is explicitly trained with observed positive node features $F_{\mathcal{P}}$. By doing so, we enable evaluating marginal log-likelihoods to reduce the reliance on attributes along which positive nodes differ most strongly, thereby encouraging the model to focus on more stable attributes.

3.3 MULTI-LEVEL INTERPRETABILITY

Our framework enables a comprehensive analysis of how different graph components influence predictions. In particular, it reveals *(i)* which components contribute most to the decision, *(ii)* how component interact and affect outcomes, and *(iii)* which components enhance or degrade performance. To do so, we provide interpretability as follows.

We achieve *graph-level interpretability* by explicitly separating feature and edge processing, enabling explicitly assessing the impact of structural information. Comparing feature-based predictions with edge-refined outputs clarifies whether relational information improves or degrades performance and allows regulation of graph connectivity effects. When refined log-likelihoods are used as pseudo-labels for training a GNN, we can identify which training signals contribute most reliably to strong performances.

We achieve *node-level interpretability* by training separate PCs on different node subsets, providing insight into which nodes best capture positive and negative classes. Controlled node-splitting strategies allow us to study how training composition affects class separation. Moreover, the HCLT backbone of our PCs exposes structural (in)dependencies among nodes, highlighting interrelations among node features.

We achieve *feature-level interpretability* since CPU relies on tractable PCs. In particular, the marginalization of feature attributes enables assessing individual feature importance and detection of noisy attributes. Additionally, the explicit encoding of feature (in)dependencies within the HCLT-structure of our PCs provides insights into intrinsic feature interactions and their influence on predicted log-likelihoods.

Please note that the interpretability considered in this work does not refer to post-hoc explainability, but rather to transparency by design. Specifically, the notion of interpretability considered here is architectural rather than based on auxiliary explanation modules. By doing so, our framework represents a meaningful step towards providing insights into how different components interact and individually contribute to prediction performance.

Table 1: Mean F1 Scores and Standard Deviations Across Various Homophilic (Left) and Heterophilic (Right) Graph Datasets. Baseline Results are Taken From [Wu et al., 2024]. We Report Results for CPU as a Standalone Classifier (CPU), CPU With LP as a Standalone Classifier (CPU-LP), CPU for Pseudo-Labeling and Training a GNN (CPU-GNN), CPU With LP for Pseudo-Labeling and Training a GNN (CPU-LP-GNN). Bold Indicates **Best**, Underlined Second-Best Performance. For our Proposed Models, the Respective Accuracies (%) are Further Reported.

	Cora	Pubmed	Citeseer	Wiki-CS	Cornell	Chameleon	Squirrel	Actor	Wisconsin	Texas
DATASET STATISTICS										
# Edges	10,556	88,648	9,104	216,123	298	36,101	217,073	30,019	515	325
# Nodes	2,708	19,717	3,327	11,701	183	2,277	5,201	7,600	251	183
# Positive	818	7,875	701	2,679	82	521	1,042	1,965	118	101
# Negative	1,890	11,842	2,626	9,022	101	1,756	4,159	5,635	133	82
Feature dim.	1,433	500	3,703	300	1,703	2,325	2,089	932	1703	1,703
Het. ratio	0.19	0.20	0.26	0.35	0.70	0.77	0.78	0.78	0.79	0.89
F1 SCORE ON $\mathcal{U}$ NODES										
GCN	$25.2_{\pm 5.2}$	$19.7_{\pm 3.1}$	$25.7_{\pm 3.1}$	$35.4_{\pm 4.1}$	$1.4_{\pm 2.9}$	$0.7_{\pm 0.6}$	$0.8_{\pm 0.7}$	$0.6_{\pm 0.3}$	$0.9_{\pm 1.8}$	$1.3_{\pm 3.0}$
MLP	$16.6_{\pm 8.1}$	$8.0_{\pm 9.0}$	$4.2_{\pm 8.4}$	$5.4_{\pm 8.4}$	$11.3_{\pm 9.1}$	$18.8_{\pm 6.9}$	$14.8_{\pm 6.4}$	$17.6_{\pm 8.1}$	$15.9_{\pm 8.9}$	$11.3_{\pm 9.2}$
GCN+TED	$80.1_{\pm 0.8}$	$75.4_{\pm 0.4}$	$70.0_{\pm 1.4}$	$80.6_{\pm 0.5}$	$13.9_{\pm 5.5}$	$17.5_{\pm 3.5}$	$22.8_{\pm 1.3}$	$27.6_{\pm 1.0}$	$19.6_{\pm 6.1}$	$11.9_{\pm 5.0}$
MLP+TED	$24.4_{\pm 9.2}$	$15.6_{\pm 19.1}$	$10.6_{\pm 9.5}$	$16.2_{\pm 7.5}$	$29.2_{\pm 9.5}$	$26.5_{\pm 4.1}$	$31.4_{\pm 2.6}$	$\underline{43.3_{\pm 6.7}}$	$40.4_{\pm 9.1}$	$41.0_{\pm 7.9}$
GCN+NNPU	$76.7_{\pm 0.9}$	$76.5_{\pm 2.3}$	$66.2_{\pm 1.1}$	$71.1_{\pm 0.6}$	$11.7_{\pm 6.3}$	$28.2_{\pm 9.2}$	$15.5_{\pm 12.6}$	$25.2_{\pm 21.7}$	$15.5_{\pm 8.6}$	$22.8_{\pm 19.7}$
MLP+NNPU	$25.2_{\pm 5.6}$	$34.2_{\pm 2.3}$	$12.9_{\pm 9.8}$	$24.4_{\pm 5.1}$	$32.9_{\pm 4.9}$	$21.0_{\pm 3.7}$	$32.1_{\pm 5.6}$	$43.2_{\pm 2.6}$	$35.6_{\pm 5.2}$	$40.1_{\pm 6.2}$
LSDAN	$63.5_{\pm 4.1}$	$69.6_{\pm 0.4}$	$47.0_{\pm 19.0}$	$80.4_{\pm 0.9}$	$26.8_{\pm 4.3}$	$30.2_{\pm 8.5}$	$25.5_{\pm 9.3}$	$38.4_{\pm 9.2}$	$25.5_{\pm 8.6}$	$41.8_{\pm 9.7}$
GRAB	$80.4_{\pm 0.2}$	$71.6_{\pm 0.3}$	$69.7_{\pm 0.4}$	$79.4_{\pm 1.0}$	$30.6_{\pm 9.0}$	$15.5_{\pm 2.2}$	$29.0_{\pm 8.4}$	$25.2_{\pm 0.7}$	$39.5_{\pm 9.6}$	$40.2_{\pm 8.2}$
PU-GNN	$79.8_{\pm 0.3}$	$73.0_{\pm 0.4}$	$69.5_{\pm 0.4}$	$80.3_{\pm 1.8}$	$28.3_{\pm 3.9}$	$31.5_{\pm 6.3}$	$29.5_{\pm 4.8}$	$32.6_{\pm 8.4}$	$27.8_{\pm 8.2}$	$42.3_{\pm 9.4}$
GPL	$\underline{81.9_{\pm 0.5}}$	$\mathbf{79.1_{\pm 0.4}}$	$\mathbf{74.1_{\pm 0.9}}$	$\underline{82.3_{\pm 0.8}}$	$37.9_{\pm 1.3}$	$\underline{36.2_{\pm 1.4}}$	$\mathbf{37.7_{\pm 5.8}}$	$\mathbf{48.0_{\pm 4.2}}$	$45.2_{\pm 2.7}$	$46.3_{\pm 3.2}$
CPU (ours)	$37.4_{\pm 0.3}$	$46.8_{\pm 1.8}$	$34.1_{\pm 2.6}$	$66.7_{\pm 1.5}$	$\mathbf{57.0_{\pm 0.5}}$	$28.7_{\pm 1.4}$	$\underline{36.5_{\pm 1.3}}$	$41.2_{\pm 0.7}$	$\mathbf{64.1_{\pm 2.1}}$	$65.8_{\pm 2.3}$
CPU-LP (ours)	$78.0_{\pm 1.8}$	$65.3_{\pm 1.2}$	$62.8_{\pm 1.4}$	$78.5_{\pm 0.9}$	$48.4_{\pm 1.4}$	$26.2_{\pm 1.2}$	$5.2_{\pm 0.8}$	$19.0_{\pm 0.8}$	$59.0_{\pm 0.7}$	$\mathbf{67.9_{\pm 1.2}}$
CPU-GNN (ours)	$65.1_{\pm 0.5}$	$70.2_{\pm 0.7}$	$55.1_{\pm 3.3}$	$81.3_{\pm 0.7}$	$\underline{55.2_{\pm 0.9}}$	$\mathbf{44.0_{\pm 0.6}}$	$21.0_{\pm 3.7}$	$21.5_{\pm 3.1}$	$\underline{60.5_{\pm 1.8}}$	$65.2_{\pm 0.8}$
CPU-LP-GNN (ours)	$\mathbf{82.4_{\pm 1.5}}$	$\underline{78.2_{\pm 0.6}}$	$\underline{71.9_{\pm 0.6}}$	$\mathbf{83.2_{\pm 0.4}}$	$45.9_{\pm 1.0}$	$27.7_{\pm 2.0}$	$0.0_{\pm 0.0}$	$24.1_{\pm 1.7}$	$55.5_{\pm 1.1}$	$\underline{67.4_{\pm 1.2}}$
ACCURACY ON $\mathcal{U}$ NODES										
CPU (ours)	$72.9_{\pm 0.2}$	$78.7_{\pm 0.4}$	$88.3_{\pm 0.3}$	$91.5_{\pm 0.4}$	$64.1_{\pm 0.9}$	$71.4_{\pm 0.7}$	$82.8_{\pm 0.4}$	$81.3_{\pm 1.0}$	$72.1_{\pm 1.9}$	$67.8_{\pm 3.2}$
CPU-LP (ours)	$92.0_{\pm 0.6}$	$83.3_{\pm 0.4}$	$91.1_{\pm 0.4}$	$94.6_{\pm 0.3}$	$40.0_{\pm 2.2}$	$44.8_{\pm 9.8}$	$88.8_{\pm 0.1}$	$68.7_{\pm 2.1}$	$63.9_{\pm 1.5}$	$67.1_{\pm 1.7}$
CPU-GNN (ours)	$86.9_{\pm 0.3}$	$87.0_{\pm 0.2}$	$92.3_{\pm 0.4}$	$95.2_{\pm 0.2}$	$59.2_{\pm 1.8}$	$82.7_{\pm 1.6}$	$88.8_{\pm 0.2}$	$83.2_{\pm 0.7}$	$65.9_{\pm 1.6}$	$63.5_{\pm 1.9}$
CPU-LP-GNN (ours)	$93.6_{\pm 0.5}$	$89.3_{\pm 0.2}$	$93.4_{\pm 0.2}$	$95.7_{\pm 0.1}$	$32.1_{\pm 2.7}$	$41.8_{\pm 11.9}$	$88.9_{\pm 0.0}$	$81.2_{\pm 2.3}$	$58.3_{\pm 1.0}$	$65.3_{\pm 2.2}$

4 EXPERIMENTS

We conduct experiments to assess the effectiveness of the complementary components of our graph PU learning framework. In particular, we measure F1 scores and accuracies for classification, compare node splitting methods for CPU, and evaluate the impact of feature attribute marginalization on the performance. Additional experiments on hyperparameter sensitivity and the effect of the size of $\mathcal{P}$ can be found in the appendix.

4.1 SETTING

Datasets We adopt homophilic and heterophilic graph datasets from previous works, namely `Cora`, `Pubmed`, `Citeseer`, `Wiki-CS` as homophilic ones and `Cornell`, `Chameleon`, `Squirrel`, `Actor`, `Wisconsin`, `Texas` as heterophilic ones [Wu et al., 2024]. We use the datasets provided in PyTorch Geometric (PyG) [Fey and Lenssen, 2019]. We converted directed graphs to undirected ones and used normalized row features for datasets with binary features. The dataset statistics are given in Table 1. Following previous works, we modify the class labels for the graph PU learning setting such that the largest class is designated as the positive class, while all remaining classes are collectively treated as the negative class [Yoo et al., 2021, Wu et al., 2024].

Experimental Setup To align with previous works on graph PU learning, we randomly select 50% of the positive nodes and use them as observed positive nodes $\mathcal{P}$. The other 50% of positive nodes together with the negatively labeled nodes are treated as unlabeled nodes $\mathcal{U}$. Our GNN is a one-hidden layer graph convolutional network (GCN) with a hidden dimension of 16 using the Adam optimizer for training with a learning rate of 0.01. For the PCs in CPU, we use the HCLT implementation provided by Pyjuice [Liu et al., 2024]. We use discrete RV for feature attributes. We use specific hyperparameters, including the node splitting and feature attribute marginalization, which have been chosen by a simple hyperparameter search. We refer to the appendix for details on the hyperparameters used to produce the results in Table 1. We compare our results to several baseline methods where we adapt the performances reported in [Wu et al., 2024]. For more details on the baseline approaches, we refer to that work. We averaged our results for 5 runs using different random seeds, providing different subsets of $\mathcal{P}$ nodes. We conducted our experiments on a machine with an NVIDIA GeForce RTX 3070 Ti Laptop GPU.

Table 2: Mean F1 Scores, Accuracies, and Standard Deviations Across Homophilic Graph Datasets with Varying Propagation Steps m for the LP Models. The Best Performing Results for CPU and CPU-GNN are Highlighted in Bold.

m	Method	Cornell F1	Cornell Acc	Chameleon F1	Chameleon Acc	Squirrel F1	Squirrel Acc	Actor F1	Actor Acc	Wisconsin F1	Wisconsin Acc	Texas F1	Texas Acc
$m=1$	CPU-LP	$40.6_{\pm 1.9}$	$35.6_{\pm 1.5}$	$26.2_{\pm 1.2}$	$44.8_{\pm 9.8}$	$5.2_{\pm 0.8}$	$88.8_{\pm 0.1}$	$19.0_{\pm 0.8}$	$68.7_{\pm 2.1}$	$21.6_{\pm 3.6}$	$38.9_{\pm 2.9}$	$45.3_{\pm 2.1}$	$43.2_{\pm 1.4}$
	CPU-LP-GNN	$46.2_{\pm 1.2}$	$37.7_{\pm 0.3}$	$27.7_{\pm 2.0}$	$41.8_{\pm 11.9}$	$0.0_{\pm 0.0}$	$88.9_{\pm 0.0}$	$24.1_{\pm 1.7}$	$81.2_{\pm 2.3}$	$35.4_{\pm 3.6}$	$46.0_{\pm 4.2}$	$51.2_{\pm 1.9}$	$44.1_{\pm 2.9}$
$m=2$	CPU-LP	$48.4_{\pm 1.4}$	$40.0_{\pm 2.2}$	$24.6_{\pm 1.4}$	$25.7_{\pm 6.1}$	$3.2_{\pm 1.4}$	$89.0_{\pm 0.1}$	$21.3_{\pm 2.1}$	$81.1_{\pm 1.0}$	$59.0_{\pm 0.7}$	$63.9_{\pm 1.5}$	$67.9_{\pm 1.2}$	$67.1_{\pm 1.7}$
	CPU-LP-GNN	$45.9_{\pm 1.0}$	$32.1_{\pm 2.7}$	$24.0_{\pm 1.0}$	$21.3_{\pm 5.4}$	$0.0_{\pm 0.0}$	$88.9_{\pm 0.0}$	$7.2_{\pm 4.3}$	$84.8_{\pm 0.2}$	$55.5_{\pm 1.1}$	$58.3_{\pm 1.0}$	$67.4_{\pm 1.2}$	$65.3_{\pm 2.2}$
$m=3$	CPU-LP	$43.6_{\pm 0.7}$	$31.8_{\pm 0.8}$	$25.7_{\pm 2.9}$	$32.5_{\pm 15.9}$	$0.4_{\pm 0.4}$	$88.9_{\pm 0.0}$	$11.6_{\pm 2.6}$	$78.7_{\pm 1.9}$	$26.6_{\pm 4.7}$	$40.1_{\pm 1.2}$	$50.6_{\pm 1.7}$	$45.7_{\pm 0.9}$
	CPU-LP-GNN	$45.2_{\pm 0.7}$	$30.7_{\pm 1.2}$	$25.7_{\pm 3.0}$	$30.5_{\pm 16.7}$	$0.0_{\pm 0.0}$	$88.9_{\pm 0.0}$	$9.2_{\pm 5.6}$	$84.1_{\pm 1.1}$	$35.6_{\pm 3.2}$	$44.1_{\pm 2.5}$	$54.9_{\pm 1.5}$	$45.6_{\pm 2.8}$
$m=4$	CPU-LP	$45.6_{\pm 0.6}$	$31.5_{\pm 1.4}$	$23.9_{\pm 1.2}$	$20.7_{\pm 5.7}$	$0.0_{\pm 0.0}$	$88.9_{\pm 0.0}$	$9.1_{\pm 1.7}$	$84.1_{\pm 0.3}$	$54.9_{\pm 2.2}$	$58.6_{\pm 4.2}$	$64.4_{\pm 1.9}$	$60.9_{\pm 2.6}$
	CPU-LP-GNN	$44.9_{\pm 0.1}$	$29.0_{\pm 0.3}$	$23.5_{\pm 0.9}$	$16.5_{\pm 5.5}$	$0.0_{\pm 0.0}$	$88.9_{\pm 0.0}$	$0.7_{\pm 0.5}$	$85.1_{\pm 0.0}$	$53.1_{\pm 1.6}$	$55.1_{\pm 2.8}$	$63.3_{\pm 1.2}$	$58.0_{\pm 1.8}$
$m=5$	CPU-LP	$44.8_{\pm 0.2}$	$29.3_{\pm 0.3}$	$24.9_{\pm 2.4}$	$27.2_{\pm 14.9}$	$0.0_{\pm 0.0}$	$88.9_{\pm 0.0}$	$4.0_{\pm 0.9}$	$83.5_{\pm 0.3}$	$32.9_{\pm 8.5}$	$40.7_{\pm 2.2}$	$51.9_{\pm 1.8}$	$44.2_{\pm 2.1}$
	CPU-LP-GNN	$44.8_{\pm 0.0}$	$28.9_{\pm 0.0}$	$24.9_{\pm 2.4}$	$25.4_{\pm 15.4}$	$0.0_{\pm 0.0}$	$88.9_{\pm 0.0}$	$0.5_{\pm 0.3}$	$85.0_{\pm 0.1}$	$39.3_{\pm 6.1}$	$42.2_{\pm 1.7}$	$55.0_{\pm 1.5}$	$44.1_{\pm 2.9}$
	CPU	$\mathbf{57.0_{\pm 0.5}}$	$\mathbf{64.1_{\pm 0.9}}$	$28.7_{\pm 1.4}$	$71.4_{\pm 0.7}$	$\mathbf{36.5_{\pm 1.3}}$	$82.8_{\pm 0.4}$	$\mathbf{41.2_{\pm 0.7}}$	$81.3_{\pm 1.0}$	$\mathbf{64.1_{\pm 2.1}}$	$\mathbf{72.1_{\pm 1.9}}$	$65.8_{\pm 2.3}$	$\mathbf{67.8_{\pm 3.2}}$
	CPU-GNN	$55.2_{\pm 0.9}$	$59.2_{\pm 1.8}$	$\mathbf{44.0_{\pm 0.6}}$	$\mathbf{82.7_{\pm 1.6}}$	$21.0_{\pm 3.7}$	$88.8_{\pm 0.2}$	$21.5_{\pm 3.1}$	$83.2_{\pm 0.7}$	$60.5_{\pm 1.8}$	$65.9_{\pm 1.6}$	$65.2_{\pm 0.8}$	$63.5_{\pm 1.9}$

Table 3: Mean F1 Scores and Standard Deviations Across two Homophilic and Two Heterophilic Graph Datasets for the Best Model in Table 1 with Different Node Splitting Methods.

Dataset	Classifier	$\mathcal{P}-\mathcal{U}$	$\mathcal{P}-$ w-25%	$\mathcal{P}-$ w-50%	$\mathcal{P}-$ w-75%	$\mathcal{P}-$ w-25$_{\mathcal{U}}$%	$\mathcal{P}-$ w-50$_{\mathcal{U}}$%	$\mathcal{P}-$ w-75$_{\mathcal{U}}$%	$\mathcal{V}-\mathcal{U}$
F1 Score on $\mathcal{U}$ nodes									
Cora	CPU-LP-GNN	$81.4_{\pm 2.3}$	$40.5_{\pm 3.0}$	$81.2_{\pm 1.4}$	$80.5_{\pm 1.7}$	$40.5_{\pm 3.0}$	$81.0_{\pm 1.4}$	$80.8_{\pm 2.1}$	$30.2_{\pm 0.0}$
Pubmed	CPU-LP-GNN	$77.6_{\pm 1.2}$	$57.5_{\pm 2.2}$	$74.1_{\pm 1.3}$	$76.4_{\pm 1.1}$	$57.6_{\pm 2.3}$	$75.2_{\pm 0.7}$	$77.7_{\pm 0.5}$	$40.3_{\pm 0.5}$
Cornell	CPU	$55.4_{\pm 1.3}$	$53.1_{\pm 0.7}$	$57.2_{\pm 1.3}$	$55.5_{\pm 1.4}$	$53.1_{\pm 0.7}$	$56.4_{\pm 1.1}$	$56.0_{\pm 1.7}$	$44.9_{\pm 0.1}$
Chameleon	CPU-GNN	$0.0_{\pm 0.0}$	$23.1_{\pm 0.4}$	$13.4_{\pm 1.0}$	$1.0_{\pm 0.7}$	$23.2_{\pm 0.3}$	$12.2_{\pm 2.8}$	$0.5_{\pm 0.5}$	$44.0_{\pm 0.6}$

4.2 RESULTS

We now present our results guided by the subsequent questions addressing different aspects of our framework.

Q1. What Does Decoupling Features and Edges Reveal? As shown in Table 1, decoupling feature and edge information exposes their distinct and dataset-dependent effects on graph PU learning. When CPU is used as a standalone classifier, subsequent LP significantly improves performance on homophilic datasets but mostly degrades it on heterophilic ones, highlighting the sensitivity to edge information in such settings. The results in Table 2 further support this finding, showing that increasing the number of iterations m for LP generally does not outperform the performance of the CPU and CPU-GNN models. In particular, sole feature-based predictions already achieve strong results on heterophilic datasets, where CPU alone is able to significantly outperform the baselines. Moreover, pseudo-labeling with our decoupled approach emphasizes the importance of controlling feature–edge interactions. Using CPU-LP to generate pseudo-labels for a GNN classifier mostly performs worse on heterophilic datasets, whereas CPU-only pseudo-labels alone can provide stronger supervision and even outperform baselines that combine feature and edge processing. However, CPU-LP pseudo-labels show superior performance for the homophilic datasets, highlighting the contribution of LP. Overall, no configuration consistently dominates across datasets, underscoring the importance of regulating feature–edge interactions in graph PU learning.

Q2. Does the Decoupling Harm the Performance? Table 1 demonstrates that our framework outperforms the baselines on four out of six heterophilic datasets, with particularly impressive improvements on `Cornell`, `Wisconsin`, and `Texas`. Moreover, we observe that our approach achieves highly competitive performance on `Squirrel`, while being more stable compared to the baselines. The lack of improvements on `Actor`, the only heterophilic dataset where our method falls short, suggests that the available positive features $F_{\mathcal{P}}$ are not very informative for classification. On homophilic datasets, CPU-derived pseudo-labels prove useful, being particularly effective as a training objective to train a highly competitive GNN for graph PU learning. Our approach outperforms baselines on `Cora` and `Wiki-CS`, while being competitive for the remaining datasets with the second-best performance. Overall, we further note the general robustness of our results for heterophilic datasets, showing particularly small standard deviations compared to the baselines. As F1 scores alone may not fully reflect classification performance, we additionally report accuracies, confirming the strong performance across both graph types.

Q3. How Does Node Splitting Affect CPU Performance? The results in Table 3 show that our dual-model approach offers considerable flexibility in controlling feature-based predictions through the choice of a specific node split. In particular, we observe that the partitioning between P_P and P_N influences CPU performance in a dataset-dependent manner. For datasets such as `Cora` and `Cornell`, most splitting strategies lead to minor performance variations, while still

Table 4: Mean F1 Scores and Standard Deviations for Unlabeled Nodes Evaluated for the Best Model in Table 1 With and Without Marginalization as well as Increased Number of Marginalized Feature Attributes.

Method	Cora	Pubmed	Cornell	Chameleon
F1 SCORE ON $\mathcal{U}$ NODES				
w/o Marginalization	$76.9_{\pm 2.6}$	$74.7_{\pm 2.7}$	$47.0_{\pm 3.6}$	$15.2_{\pm 12.0}$
Marginalization (Table 1)	$\mathbf{82.4}_{\pm \mathbf{1.5}}$	$\mathbf{78.2}_{\pm \mathbf{0.6}}$	$\mathbf{57.0}_{\pm \mathbf{0.5}}$	$\mathbf{44.0}_{\pm \mathbf{0.6}}$
Increased Marginalization	$77.1_{\pm 1.4}$	$72.9_{\pm 6.8}$	$54.2_{\pm 1.2}$	$37.1_{\pm 0.9}$

enabling notable improvements with a proper choice. In contrast, datasets like Chameleon show substantial performance differences across splits, indicating that achieving strong results requires a carefully selected node split. This highlights both the adaptability and the sensitivity of CPU to the splitting, underscoring that graph PU learning benefits from dataset-specific evaluation of node contributions.

Q4. How Does Marginalization Affect CPU Performance? Table 4 demonstrates that marginalizing feature attributes has a clear impact on the performance w.r.t. the F1 score. In particular, the performance consistently drops across all listed datasets when no marginalization is applied, indicating for our learned models that removing selected feature attributes helps to obtain log-likelihoods that better distinguish positive and negative nodes. Moreover, our results indicate that marginalization must be carried out carefully, using appropriate statistics or heuristics as feature attributes can exhibit significant dependencies and interactions. In particular, we observe that even a slight increase in the number of marginalized attributes can result in noticeable performance changes. This highlights that feature attributes should not be marginalized arbitrarily and further emphasizes the significant impact of marginalization in graph PU learning.

4.3 DISCUSSION

Our framework provides novel methods to interpret and regulate several components in graph PU learning. Furthermore, it enables strong results on various datasets, overall establishing feature-based predictions as strong baselines. Notably, CPU performs well as a standalone classifier on heterophilic graphs, while a simple LP step further boosts performance on homophilic ones, offering a transparent alternative to black-box models. Moreover, across both graph types, feature-based pseudo-labels consistently provide a simple yet effective basis for training GNNs. While CPU offers a strong foundation for more reliable graph PU learning, it currently lacks a generalized algorithm for diverse datasets. Moreover, although our framework is generally performing well, its peak performance currently relies on carefully chosen hyperparameters. Some of these hyperparameters enable a more detailed investigation of the contributions of node features and graph edges. However, the results across our four complementary settings indicate that no single model configuration is universally optimal. This highlights both the need for greater interpretability and the importance of dataset-specific regulation of feature-edge interactions. On datasets like Actor, we encounter challenges in separating node features solely based on $F_{\mathcal{P}}$, the sole source of information that guides CPU. Although this limitation is partly due to the inherently sparse nature of PU learning, more sophisticated approaches are needed to enable effective feature-based separation.

5 CONCLUSION & FUTURE WORK

We propose a probabilistic framework for graph PU learning that combines strong predictive performance with enhanced interpretability. Our approach decouples feature and edge processing as follows: CPU estimates feature-based positive and negative densities using two PCs, followed by a simple LP step to incorporate graph connectivity. Our approach serves both as strong standalone classifier and as a reliable pseudo-label generator for training powerful GNNs. Most importantly, our framework enables interpretability at graph-, node-, and feature-levels, enabling deeper insight and better control over graph PU learning. Future work includes exploring different constructions of PCs within the CPU module. While a log-likelihood ratio already performs well, more advanced evaluation strategies may improve separation. Additionally, reducing sole reliance on positive nodes and developing improved heuristics for node partitioning across diverse graph types remains a promising direction. Furthermore, alternative (deep) probabilistic models could be explored for feature-based predictions, allowing the trade-offs among tractability, expressivity, and interpretability to be controlled more flexibly.

Acknowledgements

This work was partly supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2024-00341425 and RS-2024-00406985) and the New Faculty Startup Fund from Seoul National University. Jaemin Yoo is the corresponding author.

References

YooJung Choi, Antonio Vergari, and Guy Van den Broeck. Probabilistic circuits: A unifying framework for tractable probabilistic models. *UCLA. URL: http://starai. cs. ucla. edu/papers/ProbCirc20. pdf*, page 6, 2020.

YooJung Choi, Meihua Dang, and Guy Van den Broeck. Group fairness by probabilistic modeling with latent fair decisions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

Meihua Dang, Antonio Vergari, and Guy Broeck. Strudel: Learning structured-decomposable probabilistic circuits. In *International Conference on Probabilistic Graphical Models*, pages 137–148. PMLR, 2020.

Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *The world wide web conference*, pages 417–426, 2019.

Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. Protein interface prediction using graph convolutional networks. *Advances in neural information processing systems*, 30, 2017.

Saurabh Garg, Yifan Wu, Alexander J Smola, Sivaraman Balakrishnan, and Zachary Lipton. Mixture proportion estimation and pu learning: A modern approach. *Advances in Neural Information Processing Systems*, 34: 8532–8544, 2021.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

Ryuichi Kiryo, Gang Niu, Marthinus C Du Plessis, and Masashi Sugiyama. Positive-unlabeled learning with non-negative risk estimator. *Advances in neural information processing systems*, 30, 2017.

Xiang Li, Renyu Zhu, Yao Cheng, Caihua Shan, Siqiang Luo, Dongsheng Li, and Weining Qian. Finding global homophily in graph neural networks when meeting heterophily. In *International Conference on Machine Learning*, pages 13242–13256. PMLR, 2022.

Anji Liu and Guy Van den Broeck. Tractable regularization of probabilistic circuits. *Advances in Neural Information Processing Systems*, 34:3558–3570, 2021.

Anji Liu, Honghua Zhang, and Guy Van den Broeck. Scaling up probabilistic circuits by latent variable distillation. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*, 2023.

Anji Liu, Kareem Ahmed, and Guy Van den Broeck. Scaling tractable probabilistic circuits: A systems perspective. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, 2024.

Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. Revisiting heterophily for graph neural networks. *Advances in neural information processing systems*, 35: 1362–1375, 2022.

Yuankai Luo, Lei Shi, and Xiao-Ming Wu. Classic gnns are strong baselines: Reassessing gnns for node classification. In *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

Yao Ma, Xiaorui Liu, Neil Shah, and Jiliang Tang. Is homophily a necessity for graph neural networks? In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, 2022.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22 (57):1–64, 2021.

Robert Peharz, Steven Lang, Antonio Vergari, Karl Stelzner, Alejandro Molina, Martin Trapp, Guy Van den Broeck, Kristian Kersting, and Zoubin Ghahramani. Einsum networks: Fast and scalable learning of tractable probabilistic circuits. In *International Conference on Machine Learning*, pages 7563–7574. PMLR, 2020a.

Robert Peharz, Antonio Vergari, Karl Stelzner, Alejandro Molina, Xiaoting Shao, Martin Trapp, Kristian Kersting, and Zoubin Ghahramani. Random sum-product networks: A simple and effective approach to probabilistic deep learning. In *Uncertainty in Artificial Intelligence*, pages 334–344. PMLR, 2020b.

Oleg Platonov, Denis Kuznedelev, Artem Babenko, and Liudmila Prokhorenkova. Characterizing graph datasets for node classification: Homophily-heterophily dichotomy and beyond. *Advances in Neural Information Processing Systems*, 36:523–548, 2023a.

Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A critical look at the evaluation of gnns under heterophily: Are we really making progress? In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*, 2023b.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR, 2014.

Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International conference on machine learning*, pages 8459–8468. PMLR, 2020.

Andy Shih and Stefano Ermon. Probabilistic circuits for variational inference in discrete graphical models. *Advances in neural information processing systems*, 33: 4635–4646, 2020.

Hamed Shirzad, Ameya Velingker, Balaji Venkatachalam, Danica J Sutherland, and Ali Kemal Sinop. Exphormer: Sparse transformers for graphs. In *International Conference on Machine Learning*, pages 31613–31632. PMLR, 2023.

Sahil Sidheekh and Sriraam Natarajan. Building expressive and tractable probabilistic generative models: A review. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI 2024, Jeju, South Korea, August 3-9, 2024*, 2024.

Ping Liang Tan and Robert Peharz. Hierarchical decompositional mixtures of variational autoencoders. In *International Conference on Machine Learning*, pages 6115–6124. PMLR, 2019.

Antonio Vergari, YooJung Choi, Anji Liu, Stefano Teso, and Guy Van den Broeck. A compositional atlas of tractable circuit operations for probabilistic inference. *Advances in Neural Information Processing Systems*, 34:13189–13201, 2021.

Clement Vignac, Andreas Loukas, and Pascal Frossard. Building powerful and equivariant graph neural networks with structural message-passing. *Advances in neural information processing systems*, 33:14143–14155, 2020.

Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. Session-based recommendation with graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, 2019.

Yuhao Wu, Jiangchao Yao, Bo Han, Lina Yao, and Tongliang Liu. Unraveling the impact of heterophilic structures on graph positive-unlabeled learning. In *Forty-first International Conference on Machine Learning, ICML 2024*, 2024.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

Feng Xia, Ke Sun, Shuo Yu, Abdul Aziz, Liangtian Wan, Shirui Pan, and Huan Liu. Graph learning: A survey. *IEEE Transactions on Artificial Intelligence*, 2(2):109–127, 2021.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 974–983, 2018.

Jaemin Yoo, Junghun Kim, Hoyoung Yoon, Geonsoo Kim, Changwon Jang, and U Kang. Accurate graph-based pu learning without class prior. In *2021 IEEE International Conference on Data Mining (ICDM)*, pages 827–836. IEEE, 2021.

Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. Graph transformer networks. *Advances in neural information processing systems*, 32, 2019.

Seongjun Yun, Seoyoon Kim, Junhyun Lee, Jaewoo Kang, and Hyunwoo J Kim. Neo-gnns: Neighborhood overlap-aware graph neural networks for link prediction. *Advances in Neural Information Processing Systems*, 34: 13683–13694, 2021.

Bohang Zhang, Shengjie Luo, Liwei Wang, and Di He. Rethinking the expressive power of gnns via graph biconnectivity. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*, 2023.

Xin Zheng, Yi Wang, Yixin Liu, Ming Li, Miao Zhang, Di Jin, Philip S. Yu, and Shirui Pan. Graph neural networks for graphs with heterophily: A survey. *IEEE Trans. Knowl. Data Eng.*, 2026.

Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI open*, 1:57–81, 2020.

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

A Probabilistic Circuit Framework for Interpretable Graph PU Learning (Supplementary Material)

Sagad Hamid[1] **Dooho Lee**[2] **Myeong Kong**[2] **Tanya Braun**[1] **Jaemin Yoo**[3]

[1]Computer Science Department, University of Münster, Münster, Germany
[2]School of Electrical Engineering, KAIST, Daejeon, Republic of Korea
[3]Department of Computer Science and Engineering, Seoul National University, Seoul, Republic of Korea

A HCLT PARAMETER LEARNING

For an HCLT-based PC $P(\boldsymbol{X}, \boldsymbol{Z})$ defined over RVs $\boldsymbol{X}$ and latent RVs $\boldsymbol{Z}$, let $\mathrm{flow}_{c|n_\mathrm{s}}(\boldsymbol{x}, \boldsymbol{z})$ represent the flow from child node c to its parent sum node n_s. For training nodes $\mathcal{S} \subseteq \mathcal{V}$ with features $F_\mathcal{S}$, the parameters of an HCLT-structured PC can be learned using the Expectation-Maximization (EM) algorithm, calculating expected flows in the E-step and updating parameters in the M-step [Liu and Van den Broeck, 2021]:

$$\text{(E)}\ \ \mathbb{E}^{\mathrm{flow}}_{c|n_\mathrm{s}}(F_\mathcal{S}, \boldsymbol{\theta}) := \mathbb{E}_{\boldsymbol{x}\sim F_\mathcal{D}, \boldsymbol{z}\sim P(\boldsymbol{Z}|\boldsymbol{x},\boldsymbol{\theta})}[\mathrm{flow}_{c|n_\mathrm{s}}(\boldsymbol{x}, \boldsymbol{z})]$$

$$\text{(M)}\ \ \theta_{c|n_\mathrm{s}} = \frac{\mathbb{E}^{\mathrm{flow}}_{c|n_\mathrm{s}}(F_\mathcal{S}, \boldsymbol{\theta})}{\sum_{c\in\mathrm{in}(n_\mathrm{s})} \mathbb{E}^{\mathrm{flow}}_{c|n_\mathrm{s}}(F_\mathcal{S}, \boldsymbol{\theta})}.$$

Pyjuice uses methods to calculate flows in a more GPU-efficient way [Liu et al., 2024].

B EXPERIMENTS DETAILS

B.1 HYPERPARAMETER SETTINGS FOR RESULTS IN TABLE 1

In Table 4, we present the hyperparameters for each dataset including the chosen node splitting and feature marginalization used to generate the main results in the main paper in Table 1. The hyperparameters have been obtained by means of a simple hyperparameter search.

Table 5: Detailed Hyperparameter Settings For Our Main Results in Table 1 in the Main Paper. $\mathcal{P}_P$ and $\mathcal{P}_N$ Data denotes the Data That Has Been Used to Train the Respective PCs. The Marginalization Column Denotes the Top $k\%$ Feature Attributes that Have Been Removed Based on the Variances as Described in the Main Paper. The Bin Threshold denotes the Binning Size for Discretizing Node Features. PC LR and Latent Denote the Learning Rate and the Latent Size Used for the PCs. The Epoch Columns Denote the Number of Epochs for the PCs and the GNN, Respectively. The Last Column denotes the Number of LP Propagations, i.e., the Parameter m in the Main Paper.

Dataset	P_P Data	P_N Data	Marginalization	Bin Threshold	PC LR	PC Latent	PC epochs	GNN epochs	LP m
Cora	$\mathcal{P}$	w-60$_{\mathcal{U}}$%	1%	-	0.01	4	1500	1000	5
Pubmed	$\mathcal{P}$	w-80$_{\mathcal{U}}$%	10%	0.001	0.05	16	1000	1000	1
Citeseer	$\mathcal{P}$	$\mathcal{U}$	2%	-	0.15	16	1000	500	3
Wiki-CS	$\mathcal{P}$	$\mathcal{U}$	1%	0.01	0.01	4	500	1500	1
Cornell	$\mathcal{P}$	w-40$_{\mathcal{U}}$%	15%	-	0.1	4	500	1000	2
Chameleon	$\mathcal{V}$	$\mathcal{U}$	1%	0.01	0.25	4	1000	1000	1
Squirrel	$\mathcal{P}$	$\mathcal{U}$	-	-	0.15	4	500	1000	1
Actor	$\mathcal{P}$	$\mathcal{U}$	3%	0.01	0.008	32	1000	1000	1
Wisconsin	$\mathcal{P}$	w-65$_{\mathcal{U}}$%	10%	-	0.1	4	500	1000	2
Texas	$\mathcal{P}$	w-75$_{\mathcal{U}}$%	10%	-	0.1	4	500	1000	2

B.2 MARGINALIZED ATTRIBUTES FOR RESULTS IN TABLE 3

For our results in Table 3, we have reported 3 marginalization settings while keeping the same hyperparameters as for the results in Table 1. The first setting reports F1 scores without marginalizing out any feature attributes, i.e., $\boldsymbol{X}' = \boldsymbol{X}$. The second setting reports F1 scores with feature attributes being marginalized out as for the results in Table 1. The third setting reports F1 scores where for each dataset the number of marginalized feature attributes (based on the variances) has been increased:

- `Cora`: Increased from 1% to 2%
- `Pubmed`: Increased from 10% to 15%
- `Cornell`: Increased from 15% to 20%.
- `Chameleon`: Increased from 1% to 2%

B.3 DETAILED DATASET STATISTICS

Table 6: Detailed Dataset Statistics Before and After Converting Directed Graphs to Undirected Ones.

	Cora	Pubmed	Citeseer	Wiki-CS	Cornell	Chameleon	Squirrel	Actor	Wisconsin	Texas
DATASET STATISTICS (AS STATED IN PYG)										
# Edges	10,556	88,648	9,104	216,123	298	36,101	217,073	30,019	515	325
# Nodes	2,708	19,717	3,327	11,701	183	2,277	5,201	7,600	251	183
# Positive	818	7,875	701	2,679	82	521	1,042	1,965	118	101
# Negative	1,890	11,842	2,626	9,022	101	1,756	4,159	5,635	133	82
Feature dim.	1,433	500	3,703	300	1,703	2,325	2,089	932	1703	1,703
Het. ratio	0.19	0.20	0.26	0.35	0.70	0.77	0.78	0.78	0.79	0.89
DATASET STATISTICS (AFTER CONVERTING DIRECTED GRAPHS TO UNDIRECTED ONES)										
# Edges	**10,556**	**88,648**	**9,104**	**431,726**	**557**	**62,792**	**396,846**	**53,411**	**916**	**574**

C HYPERPARAMETER SENSITIVITY

Table 7: Mean F1 Scores, Accuracies, and Standard Deviations for Unlabeled Nodes Evaluated With Different Ratios of Observed Positive Nodes $\mathcal{P}$.

	Cora	Pubmed	Citeseer	Wiki-CS	Cornell	Chameleon	Squirrel	Actor	Wisconsin	Texas
F1 Score on $\mathcal{U}$ nodes (Ratio: 30%)										
CPU	$40.9_{\pm 1.2}$	$31.6_{\pm 1.5}$	$26.1_{\pm 0.9}$	$54.4_{\pm 1.4}$	$63.9_{\pm 1.1}$	$32.3_{\pm 2.1}$	$41.3_{\pm 0.9}$	$36.6_{\pm 2.2}$	$70.3_{\pm 1.8}$	$72.8_{\pm 1.8}$
CPU-LP	$73.2_{\pm 2.7}$	$48.3_{\pm 2.9}$	$54.6_{\pm 2.1}$	$67.8_{\pm 1.3}$	$56.8_{\pm 1.8}$	$30.0_{\pm 4.2}$	$4.1_{\pm 0.3}$	$12.8_{\pm 2.7}$	$69.1_{\pm 1.3}$	$72.8_{\pm 1.4}$
CPU-GNN	$60.1_{\pm 1.6}$	$40.0_{\pm 1.8}$	$8.0_{\pm 7.9}$	$66.1_{\pm 1.4}$	$61.2_{\pm 1.3}$	$39.8_{\pm 5.5}$	$13.3_{\pm 2.4}$	$2.5_{\pm 1.9}$	$66.8_{\pm 2.4}$	$72.6_{\pm 1.0}$
CPU-LP-GNN	$78.3_{\pm 2.3}$	$58.8_{\pm 2.5}$	$62.1_{\pm 2.5}$	$76.1_{\pm 1.3}$	$54.4_{\pm 1.8}$	$32.4_{\pm 1.2}$	$0.0_{\pm 0.0}$	$4.4_{\pm 4.3}$	$65.8_{\pm 1.3}$	$73.3_{\pm 1.2}$
Accuracy on $\mathcal{U}$ nodes (Ratio: 30%)										
CPU	$69.4_{\pm 0.9}$	$72.3_{\pm 0.3}$	$84.6_{\pm 0.2}$	$88.5_{\pm 0.2}$	$65.5_{\pm 1.0}$	$66.5_{\pm 7.2}$	$82.1_{\pm 0.5}$	$82.3_{\pm 0.3}$	$73.1_{\pm 1.5}$	$70.6_{\pm 2.3}$
CPU-LP	$88.8_{\pm 0.9}$	$76.4_{\pm 0.7}$	$88.4_{\pm 0.4}$	$91.2_{\pm 0.3}$	$45.8_{\pm 3.7}$	$51.2_{\pm 11.5}$	$85.2_{\pm 0.1}$	$75.5_{\pm 1.6}$	$71.4_{\pm 1.4}$	$69.2_{\pm 2.2}$
CPU-GNN	$82.5_{\pm 0.8}$	$75.4_{\pm 0.5}$	$84.9_{\pm 0.7}$	$91.0_{\pm 0.3}$	$60.6_{\pm 1.3}$	$74.4_{\pm 14.5}$	$85.5_{\pm 0.1}$	$80.4_{\pm 0.1}$	$68.0_{\pm 2.2}$	$68.0_{\pm 2.1}$
CPU-LP-GNN	$90.9_{\pm 0.8}$	$80.5_{\pm 0.8}$	$90.7_{\pm 0.4}$	$93.1_{\pm 0.3}$	$39.2_{\pm 4.0}$	$50.3_{\pm 13.3}$	$85.1_{\pm 0.0}$	$79.7_{\pm 0.8}$	$66.6_{\pm 1.1}$	$68.6_{\pm 2.4}$
F1 Score on $\mathcal{U}$ nodes (Ratio: 40%)										
CPU	$39.4_{\pm 1.0}$	$38.4_{\pm 2.5}$	$32.5_{\pm 0.8}$	$65.1_{\pm 0.6}$	$60.3_{\pm 1.2}$	$32.0_{\pm 1.1}$	$39.1_{\pm 1.5}$	$41.7_{\pm 1.0}$	$68.2_{\pm 2.8}$	$69.9_{\pm 1.1}$
CPU-LP	$76.4_{\pm 3.0}$	$58.9_{\pm 2.0}$	$60.8_{\pm 1.7}$	$76.9_{\pm 1.0}$	$53.1_{\pm 1.9}$	$31.0_{\pm 3.0}$	$4.5_{\pm 0.8}$	$16.7_{\pm 1.5}$	$64.8_{\pm 1.4}$	$70.1_{\pm 2.6}$
CPU-GNN	$62.1_{\pm 1.6}$	$58.5_{\pm 2.4}$	$39.9_{\pm 3.4}$	$79.6_{\pm 0.8}$	$58.5_{\pm 1.0}$	$44.0_{\pm 1.8}$	$14.7_{\pm 4.0}$	$11.2_{\pm 1.8}$	$64.3_{\pm 3.6}$	$68.8_{\pm 1.2}$
CPU-LP-GNN	$80.7_{\pm 2.8}$	$72.3_{\pm 1.2}$	$69.4_{\pm 1.4}$	$82.4_{\pm 0.8}$	$50.5_{\pm 1.7}$	$32.5_{\pm 3.2}$	$0.0_{\pm 0.0}$	$8.8_{\pm 6.4}$	$60.7_{\pm 1.4}$	$69.6_{\pm 2.1}$
Accuracy on $\mathcal{U}$ nodes (Ratio: 40%)										
CPU	$71.6_{\pm 0.6}$	$75.7_{\pm 0.3}$	$86.5_{\pm 0.2}$	$90.8_{\pm 0.1}$	$64.2_{\pm 1.9}$	$70.0_{\pm 1.1}$	$82.3_{\pm 0.4}$	$82.6_{\pm 0.7}$	$73.2_{\pm 2.4}$	$69.0_{\pm 1.4}$
CPU-LP	$90.9_{\pm 0.9}$	$80.8_{\pm 0.5}$	$90.0_{\pm 0.4}$	$93.7_{\pm 0.2}$	$43.6_{\pm 4.2}$	$52.6_{\pm 7.6}$	$87.0_{\pm 0.2}$	$74.4_{\pm 1.7}$	$68.2_{\pm 2.0}$	$67.7_{\pm 3.8}$
CPU-GNN	$84.6_{\pm 0.5}$	$82.2_{\pm 0.6}$	$89.4_{\pm 0.4}$	$94.4_{\pm 0.2}$	$59.9_{\pm 1.8}$	$79.6_{\pm 4.5}$	$87.2_{\pm 0.2}$	$82.4_{\pm 0.3}$	$67.5_{\pm 3.5}$	$64.5_{\pm 1.7}$
CPU-LP-GNN	$92.5_{\pm 0.9}$	$86.4_{\pm 0.4}$	$92.4_{\pm 0.3}$	$95.1_{\pm 0.2}$	$36.8_{\pm 4.5}$	$51.0_{\pm 9.3}$	$86.9_{\pm 0.0}$	$82.0_{\pm 1.1}$	$62.4_{\pm 1.9}$	$65.7_{\pm 3.6}$
F1 Score on $\mathcal{U}$ nodes (Ratio: 50%)										
CPU	$37.4_{\pm 0.3}$	$46.8_{\pm 1.8}$	$34.1_{\pm 2.6}$	$66.7_{\pm 1.5}$	$57.0_{\pm 0.5}$	$28.7_{\pm 1.4}$	$36.5_{\pm 1.3}$	$41.2_{\pm 0.7}$	$64.1_{\pm 2.1}$	$65.8_{\pm 2.3}$
CPU-LP	$78.0_{\pm 1.8}$	$65.3_{\pm 1.2}$	$62.8_{\pm 1.4}$	$78.5_{\pm 0.9}$	$48.4_{\pm 1.4}$	$26.2_{\pm 1.2}$	$5.2_{\pm 0.8}$	$19.0_{\pm 0.8}$	$59.0_{\pm 0.7}$	$67.9_{\pm 1.2}$
CPU-GNN	$65.1_{\pm 0.5}$	$70.2_{\pm 0.7}$	$55.1_{\pm 3.3}$	$81.3_{\pm 0.7}$	$55.2_{\pm 0.9}$	$44.0_{\pm 0.6}$	$21.0_{\pm 3.7}$	$21.5_{\pm 3.1}$	$60.5_{\pm 1.8}$	$65.2_{\pm 0.8}$
CPU-LP-GNN	$82.4_{\pm 1.5}$	$78.2_{\pm 0.6}$	$71.9_{\pm 0.6}$	$83.2_{\pm 0.4}$	$45.9_{\pm 1.0}$	$27.7_{\pm 2.0}$	$0.0_{\pm 0.0}$	$24.1_{\pm 1.7}$	$55.5_{\pm 1.1}$	$67.4_{\pm 1.2}$
Accuracy on $\mathcal{U}$ nodes (Ratio: 50%)										
CPU	$72.9_{\pm 0.2}$	$78.7_{\pm 0.4}$	$88.3_{\pm 0.3}$	$91.5_{\pm 0.4}$	$64.1_{\pm 0.9}$	$71.4_{\pm 0.7}$	$82.8_{\pm 0.4}$	$81.3_{\pm 1.0}$	$72.1_{\pm 1.9}$	$67.8_{\pm 3.2}$
CPU-LP	$92.0_{\pm 0.6}$	$83.3_{\pm 0.4}$	$91.1_{\pm 0.4}$	$94.6_{\pm 0.3}$	$40.0_{\pm 2.2}$	$44.8_{\pm 9.8}$	$88.8_{\pm 0.1}$	$68.7_{\pm 2.1}$	$63.9_{\pm 1.5}$	$67.1_{\pm 1.7}$
CPU-GNN	$86.9_{\pm 0.3}$	$87.0_{\pm 0.2}$	$92.3_{\pm 0.4}$	$95.2_{\pm 0.2}$	$59.2_{\pm 1.8}$	$82.7_{\pm 1.6}$	$88.8_{\pm 0.2}$	$83.2_{\pm 0.7}$	$65.9_{\pm 1.6}$	$63.5_{\pm 1.9}$
CPU-LP-GNN	$93.6_{\pm 0.5}$	$89.3_{\pm 0.2}$	$93.4_{\pm 0.2}$	$95.7_{\pm 0.1}$	$32.1_{\pm 2.7}$	$41.8_{\pm 11.9}$	$88.9_{\pm 0.0}$	$81.2_{\pm 2.3}$	$58.3_{\pm 1.0}$	$65.3_{\pm 2.2}$

Table 8: Mean F1 Scores, Accuracies, and Standard Deviations with Different PC Epochs.

	Cora	Pubmed	Citeseer	Wiki-CS	Cornell	Chameleon	Squirrel	Actor	Wisconsin	Texas
F1 Score on $\mathcal{U}$ nodes (PC Epochs: 500)										
CPU	$38.8_{\pm 0.8}$	$44.3_{\pm 2.4}$	$42.1_{\pm 3.1}$	$66.7_{\pm 1.5}$	$57.0_{\pm 0.5}$	$31.2_{\pm 1.7}$	$36.5_{\pm 1.3}$	$39.0_{\pm 0.4}$	$64.1_{\pm 2.1}$	$65.8_{\pm 2.3}$
CPU-LP	$77.2_{\pm 1.7}$	$63.7_{\pm 2.3}$	$63.5_{\pm 0.9}$	$78.5_{\pm 0.9}$	$48.4_{\pm 1.4}$	$25.7_{\pm 3.7}$	$5.2_{\pm 0.8}$	$22.5_{\pm 2.7}$	$59.0_{\pm 0.7}$	$67.9_{\pm 1.2}$
CPU-GNN	$64.4_{\pm 0.3}$	$68.7_{\pm 1.2}$	$64.7_{\pm 2.2}$	$81.3_{\pm 0.7}$	$55.2_{\pm 0.9}$	$42.5_{\pm 1.4}$	$21.0_{\pm 3.7}$	$24.3_{\pm 2.9}$	$60.5_{\pm 1.8}$	$65.2_{\pm 0.8}$
CPU-LP-GNN	$80.2_{\pm 1.8}$	$77.5_{\pm 1.0}$	$71.9_{\pm 0.9}$	$83.2_{\pm 0.4}$	$45.9_{\pm 1.0}$	$28.6_{\pm 2.9}$	$0.0_{\pm 0.0}$	$8.0_{\pm 5.3}$	$55.5_{\pm 1.1}$	$67.4_{\pm 1.2}$
Accuracy on $\mathcal{U}$ nodes (PC Epochs: 500)										
CPU	$70.7_{\pm 0.5}$	$78.4_{\pm 0.3}$	$87.5_{\pm 0.4}$	$91.5_{\pm 0.4}$	$64.1_{\pm 0.9}$	$71.5_{\pm 1.2}$	$82.8_{\pm 0.4}$	$77.4_{\pm 1.7}$	$72.1_{\pm 1.9}$	$67.8_{\pm 3.2}$
CPU-LP	$90.8_{\pm 1.0}$	$83.1_{\pm 0.6}$	$90.7_{\pm 0.3}$	$94.6_{\pm 0.3}$	$40.0_{\pm 2.2}$	$46.9_{\pm 10.0}$	$88.8_{\pm 0.1}$	$81.2_{\pm 1.5}$	$63.9_{\pm 1.5}$	$67.1_{\pm 1.7}$
CPU-GNN	$84.9_{\pm 0.6}$	$86.7_{\pm 0.3}$	$93.1_{\pm 0.3}$	$95.2_{\pm 0.2}$	$59.2_{\pm 1.8}$	$79.0_{\pm 1.7}$	$88.8_{\pm 0.2}$	$79.2_{\pm 2.3}$	$65.9_{\pm 1.6}$	$63.5_{\pm 1.9}$
CPU-LP-GNN	$92.2_{\pm 0.9}$	$89.2_{\pm 0.3}$	$93.1_{\pm 0.3}$	$95.7_{\pm 0.1}$	$32.1_{\pm 2.7}$	$45.0_{\pm 11.4}$	$88.9_{\pm 0.0}$	$84.6_{\pm 0.4}$	$58.3_{\pm 1.0}$	$65.3_{\pm 2.2}$
F1 Score on $\mathcal{U}$ nodes (PC Epochs: 1000)										
CPU	$37.3_{\pm 0.4}$	$46.8_{\pm 1.8}$	$34.1_{\pm 2.6}$	$65.0_{\pm 0.8}$	$56.4_{\pm 1.1}$	$28.7_{\pm 1.4}$	$35.3_{\pm 1.6}$	$41.2_{\pm 0.7}$	$64.3_{\pm 1.9}$	$65.8_{\pm 2.6}$
CPU-LP	$78.1_{\pm 1.5}$	$65.3_{\pm 1.2}$	$62.8_{\pm 1.4}$	$78.0_{\pm 0.9}$	$48.7_{\pm 1.6}$	$26.2_{\pm 1.2}$	$4.6_{\pm 0.9}$	$19.0_{\pm 0.8}$	$59.6_{\pm 1.1}$	$68.2_{\pm 1.9}$
CPU-GNN	$64.4_{\pm 0.5}$	$70.2_{\pm 0.7}$	$55.1_{\pm 3.3}$	$80.7_{\pm 0.7}$	$55.2_{\pm 1.1}$	$44.0_{\pm 0.6}$	$12.8_{\pm 2.8}$	$21.5_{\pm 3.1}$	$60.9_{\pm 1.7}$	$65.9_{\pm 1.6}$
CPU-LP-GNN	$82.4_{\pm 1.5}$	$78.2_{\pm 0.6}$	$71.9_{\pm 0.6}$	$83.2_{\pm 0.4}$	$46.0_{\pm 1.2}$	$27.7_{\pm 2.0}$	$0.0_{\pm 0.0}$	$24.1_{\pm 1.7}$	$55.1_{\pm 1.0}$	$67.3_{\pm 1.0}$
Accuracy on $\mathcal{U}$ nodes (PC Epochs: 1000)										
CPU	$72.8_{\pm 0.5}$	$78.7_{\pm 0.4}$	$88.3_{\pm 0.3}$	$91.5_{\pm 0.2}$	$63.8_{\pm 1.3}$	$71.4_{\pm 0.7}$	$84.3_{\pm 0.6}$	$81.3_{\pm 1.0}$	$72.4_{\pm 1.7}$	$68.3_{\pm 3.4}$
CPU-LP	$92.1_{\pm 0.6}$	$83.3_{\pm 0.4}$	$91.1_{\pm 0.4}$	$94.6_{\pm 0.2}$	$40.8_{\pm 2.6}$	$44.8_{\pm 9.8}$	$88.9_{\pm 0.1}$	$68.7_{\pm 2.1}$	$64.8_{\pm 2.0}$	$67.8_{\pm 2.3}$
CPU-GNN	$86.5_{\pm 0.4}$	$87.0_{\pm 0.2}$	$92.3_{\pm 0.4}$	$95.3_{\pm 0.2}$	$59.4_{\pm 2.3}$	$82.7_{\pm 1.6}$	$89.1_{\pm 0.2}$	$83.2_{\pm 0.7}$	$66.5_{\pm 1.3}$	$64.7_{\pm 2.7}$
CPU-LP-GNN	$93.6_{\pm 0.6}$	$89.3_{\pm 0.2}$	$93.4_{\pm 0.2}$	$95.8_{\pm 0.1}$	$32.8_{\pm 3.0}$	$41.8_{\pm 11.9}$	$88.9_{\pm 0.0}$	$81.2_{\pm 2.3}$	$58.1_{\pm 1.3}$	$65.6_{\pm 1.6}$
F1 Score on $\mathcal{U}$ nodes (PC Epochs: 1500)										
CPU	$37.4_{\pm 0.3}$	$46.8_{\pm 3.3}$	$29.6_{\pm 2.3}$	$64.7_{\pm 0.9}$	$56.5_{\pm 1.2}$	$26.7_{\pm 1.9}$	$34.4_{\pm 1.4}$	$40.0_{\pm 0.6}$	$64.5_{\pm 1.6}$	$66.0_{\pm 2.6}$
CPU-LP	$78.0_{\pm 1.8}$	$64.6_{\pm 1.8}$	$62.9_{\pm 2.0}$	$78.1_{\pm 0.9}$	$48.8_{\pm 1.6}$	$24.1_{\pm 3.1}$	$4.2_{\pm 1.0}$	$19.1_{\pm 2.1}$	$59.9_{\pm 1.2}$	$68.2_{\pm 1.9}$
CPU-GNN	$65.1_{\pm 0.5}$	$70.8_{\pm 2.6}$	$49.5_{\pm 2.8}$	$80.8_{\pm 0.8}$	$55.3_{\pm 1.0}$	$41.4_{\pm 4.6}$	$4.7_{\pm 3.1}$	$18.4_{\pm 3.0}$	$60.7_{\pm 1.7}$	$66.0_{\pm 1.6}$
CPU-LP-GNN	$82.4_{\pm 1.5}$	$78.1_{\pm 0.9}$	$71.7_{\pm 0.9}$	$83.3_{\pm 0.4}$	$46.1_{\pm 1.3}$	$27.4_{\pm 1.3}$	$0.0_{\pm 0.0}$	$5.1_{\pm 3.0}$	$55.7_{\pm 1.5}$	$67.3_{\pm 1.0}$
Accuracy on $\mathcal{U}$ nodes (PC Epochs: 1500)										
CPU	$72.9_{\pm 0.2}$	$78.1_{\pm 0.9}$	$88.4_{\pm 0.1}$	$91.6_{\pm 0.2}$	$63.9_{\pm 1.4}$	$67.7_{\pm 6.1}$	$85.1_{\pm 0.8}$	$81.6_{\pm 0.8}$	$72.6_{\pm 1.5}$	$68.6_{\pm 3.4}$
CPU-LP	$92.0_{\pm 0.6}$	$82.8_{\pm 0.7}$	$91.4_{\pm 0.4}$	$94.7_{\pm 0.2}$	$41.1_{\pm 2.5}$	$42.2_{\pm 7.7}$	$88.9_{\pm 0.1}$	$81.5_{\pm 0.9}$	$65.3_{\pm 2.0}$	$67.8_{\pm 2.3}$
CPU-GNN	$86.9_{\pm 0.3}$	$87.1_{\pm 0.6}$	$91.8_{\pm 0.3}$	$95.3_{\pm 0.2}$	$59.4_{\pm 2.3}$	$75.8_{\pm 11.3}$	$89.0_{\pm 0.1}$	$83.6_{\pm 0.7}$	$66.1_{\pm 1.3}$	$65.0_{\pm 2.8}$
CPU-LP-GNN	$93.6_{\pm 0.5}$	$89.2_{\pm 0.3}$	$93.6_{\pm 0.2}$	$95.8_{\pm 0.1}$	$33.1_{\pm 3.1}$	$40.4_{\pm 8.6}$	$88.9_{\pm 0.0}$	$84.9_{\pm 0.1}$	$59.1_{\pm 1.6}$	$65.6_{\pm 1.6}$

Table 9: Mean F1 Scores, Accuracies, and Standard Deviations with Different GNN Epochs.

	Cora	Pubmed	Citeseer	Wiki-CS	Cornell	Chameleon	Squirrel	Actor	Wisconsin	Texas
F1 Score on $\mathcal{U}$ nodes (GNN Epochs: 500)										
CPU-GNN	$66.0_{\pm 0.8}$	$69.4_{\pm 0.6}$	$55.1_{\pm 3.3}$	$81.4_{\pm 0.8}$	$54.7_{\pm 0.8}$	$39.1_{\pm 5.2}$	$11.6_{\pm 4.7}$	$12.6_{\pm 10.4}$	$60.1_{\pm 1.8}$	$65.3_{\pm 1.5}$
CPU-LP-GNN	$82.4_{\pm 1.3}$	$77.9_{\pm 0.5}$	$71.9_{\pm 0.6}$	$83.2_{\pm 0.4}$	$45.9_{\pm 1.0}$	$27.6_{\pm 2.1}$	$0.0_{\pm 0.0}$	$4.0_{\pm 5.0}$	$55.2_{\pm 1.2}$	$66.8_{\pm 0.9}$
Accuracy on $\mathcal{U}$ nodes (GNN Epochs: 500)										
CPU-GNN	$87.1_{\pm 0.3}$	$86.9_{\pm 0.1}$	$92.3_{\pm 0.4}$	$95.3_{\pm 0.2}$	$58.5_{\pm 2.0}$	$80.4_{\pm 3.1}$	$89.0_{\pm 0.2}$	$83.6_{\pm 1.3}$	$65.6_{\pm 1.7}$	$63.0_{\pm 2.4}$
CPU-LP-GNN	$93.6_{\pm 0.4}$	$89.3_{\pm 0.2}$	$93.4_{\pm 0.2}$	$95.7_{\pm 0.1}$	$31.8_{\pm 2.6}$	$41.3_{\pm 12.1}$	$88.9_{\pm 0.0}$	$85.0_{\pm 0.2}$	$57.1_{\pm 0.5}$	$64.4_{\pm 1.8}$
F1 Score on $\mathcal{U}$ nodes (GNN Epochs: 1000)										
CPU-GNN	$65.1_{\pm 0.5}$	$70.2_{\pm 0.7}$	$56.1_{\pm 3.5}$	$81.3_{\pm 0.9}$	$55.2_{\pm 0.9}$	$44.0_{\pm 0.6}$	$21.0_{\pm 3.7}$	$21.5_{\pm 3.1}$	$60.5_{\pm 1.8}$	$65.2_{\pm 0.8}$
CPU-LP-GNN	$82.4_{\pm 1.5}$	$78.2_{\pm 0.6}$	$70.9_{\pm 1.1}$	$83.1_{\pm 0.5}$	$45.9_{\pm 1.0}$	$27.7_{\pm 2.0}$	$0.0_{\pm 0.0}$	$24.1_{\pm 1.7}$	$55.5_{\pm 1.1}$	$67.4_{\pm 1.2}$
Accuracy on $\mathcal{U}$ nodes (GNN Epochs: 1000)										
CPU-GNN	$86.9_{\pm 0.3}$	$87.0_{\pm 0.2}$	$92.3_{\pm 0.4}$	$95.3_{\pm 0.2}$	$59.2_{\pm 1.8}$	$82.7_{\pm 1.6}$	$88.8_{\pm 0.2}$	$83.2_{\pm 0.7}$	$65.9_{\pm 1.6}$	$63.5_{\pm 1.9}$
CPU-LP-GNN	$93.6_{\pm 0.5}$	$89.3_{\pm 0.2}$	$93.1_{\pm 0.3}$	$95.7_{\pm 0.2}$	$32.1_{\pm 2.7}$	$41.8_{\pm 11.9}$	$88.9_{\pm 0.0}$	$81.2_{\pm 2.3}$	$58.3_{\pm 1.0}$	$65.3_{\pm 2.2}$
F1 Score on $\mathcal{U}$ nodes (GNN Epochs: 1500)										
CPU-GNN	$65.2_{\pm 0.5}$	$70.1_{\pm 0.8}$	$56.9_{\pm 3.1}$	$81.3_{\pm 0.7}$	$55.3_{\pm 1.1}$	$43.8_{\pm 0.7}$	$20.6_{\pm 3.1}$	$21.9_{\pm 2.2}$	$60.9_{\pm 2.4}$	$64.9_{\pm 1.3}$
CPU-LP-GNN	$82.3_{\pm 1.5}$	$78.2_{\pm 0.5}$	$71.0_{\pm 1.1}$	$83.2_{\pm 0.4}$	$46.1_{\pm 0.7}$	$27.7_{\pm 2.0}$	$0.0_{\pm 0.0}$	$8.8_{\pm 2.0}$	$55.5_{\pm 1.1}$	$67.4_{\pm 1.2}$
Accuracy on $\mathcal{U}$ nodes (GNN Epochs: 1500)										
CPU-GNN	$86.7_{\pm 0.3}$	$87.0_{\pm 0.1}$	$92.4_{\pm 0.4}$	$95.2_{\pm 0.2}$	$59.3_{\pm 1.4}$	$82.6_{\pm 2.3}$	$88.8_{\pm 0.2}$	$83.2_{\pm 0.8}$	$66.5_{\pm 2.4}$	$63.3_{\pm 2.6}$
CPU-LP-GNN	$93.6_{\pm 0.5}$	$89.3_{\pm 0.3}$	$93.2_{\pm 0.3}$	$95.7_{\pm 0.1}$	$33.2_{\pm 1.6}$	$42.0_{\pm 11.9}$	$88.9_{\pm 0.0}$	$84.8_{\pm 0.2}$	$57.9_{\pm 0.9}$	$65.3_{\pm 2.2}$

Table 10: Mean F1 Scores, Accuracies, and Standard Deviations with Different GNN Hidden Layer Sizes.

	Cora	Pubmed	Citeseer	Wiki-CS	Cornell	Chameleon	Squirrel	Actor	Wisconsin	Texas
Dataset Statistics										
F1 Score on $\mathcal{U}$ nodes (GNN Hidden Size: 16)										
CPU-GNN	$65.1_{\pm 0.5}$	$70.2_{\pm 0.7}$	$55.1_{\pm 3.3}$	$81.3_{\pm 0.7}$	$55.2_{\pm 0.9}$	$44.0_{\pm 0.6}$	$21.0_{\pm 3.7}$	$21.5_{\pm 3.1}$	$60.5_{\pm 1.8}$	$65.2_{\pm 0.8}$
CPU-LP-GNN	$82.4_{\pm 1.5}$	$78.2_{\pm 0.6}$	$71.9_{\pm 0.6}$	$83.2_{\pm 0.4}$	$45.9_{\pm 1.0}$	$27.7_{\pm 2.0}$	$0.0_{\pm 0.0}$	$24.1_{\pm 1.7}$	$55.5_{\pm 1.1}$	$67.4_{\pm 1.2}$
Accuracy on $\mathcal{U}$ nodes (GNN Hidden Size: 16)										
CPU-GNN	$86.9_{\pm 0.3}$	$87.0_{\pm 0.2}$	$92.3_{\pm 0.4}$	$95.2_{\pm 0.2}$	$59.2_{\pm 1.8}$	$82.7_{\pm 1.6}$	$88.8_{\pm 0.2}$	$83.2_{\pm 0.7}$	$65.9_{\pm 1.6}$	$63.5_{\pm 1.9}$
CPU-LP-GNN	$93.6_{\pm 0.5}$	$89.3_{\pm 0.2}$	$93.4_{\pm 0.2}$	$95.7_{\pm 0.1}$	$32.1_{\pm 2.7}$	$41.8_{\pm 11.9}$	$88.9_{\pm 0.0}$	$81.2_{\pm 2.3}$	$58.3_{\pm 1.0}$	$65.3_{\pm 2.2}$
F1 Score on $\mathcal{U}$ nodes (GNN Hidden Size: 32)										
CPU-GNN	$64.7_{\pm 1.5}$	$68.5_{\pm 2.5}$	$57.2_{\pm 4.7}$	$81.3_{\pm 0.7}$	$55.6_{\pm 1.2}$	$40.4_{\pm 5.6}$	$20.7_{\pm 3.0}$	$21.5_{\pm 2.1}$	$60.5_{\pm 2.3}$	$65.0_{\pm 0.8}$
CPU-LP-GNN	$82.1_{\pm 1.4}$	$77.3_{\pm 1.3}$	$71.0_{\pm 1.4}$	$83.3_{\pm 0.3}$	$45.9_{\pm 0.7}$	$26.5_{\pm 3.3}$	$0.1_{\pm 0.2}$	$6.4_{\pm 2.5}$	$55.1_{\pm 1.7}$	$67.1_{\pm 1.3}$
Accuracy on $\mathcal{U}$ nodes (GNN Hidden Size: 32)										
CPU-GNN	$86.6_{\pm 0.5}$	$86.6_{\pm 0.4}$	$92.5_{\pm 0.6}$	$95.3_{\pm 0.1}$	$59.7_{\pm 1.1}$	$75.4_{\pm 12.8}$	$88.8_{\pm 0.3}$	$83.3_{\pm 0.6}$	$65.9_{\pm 1.9}$	$63.5_{\pm 1.9}$
CPU-LP-GNN	$93.4_{\pm 0.5}$	$89.1_{\pm 0.4}$	$93.2_{\pm 0.3}$	$95.7_{\pm 0.1}$	$32.8_{\pm 1.3}$	$35.6_{\pm 11.7}$	$88.9_{\pm 0.0}$	$84.9_{\pm 0.1}$	$58.8_{\pm 1.1}$	$64.7_{\pm 1.8}$

Table 11: Mean F1 Scores and Standard Deviations for Unlabeled Nodes Evaluated for the Best Model in Table 1 With Different Learning Rates for the PCs for Each Dataset. Cora: [0.001, 0.01, 0.02], Pubmed: [0.01, 0.05 0.1], Cornell: [0.05, 0.1, 0.15], Chameleon: [0.2, 0.25, 0.3]

	Cora	Pubmed	Cornell	Chameleon
F1 Score on $\mathcal{U}$ nodes				
Decreased	$30.2_{\pm 0.0}$	$77.3_{\pm 0.5}$	$56.1_{\pm 0.4}$	$43.1_{\pm 1.4}$
As In Table 1	$82.4_{\pm 1.5}$	$78.2_{\pm 0.6}$	$57.0_{\pm 0.5}$	$44.0_{\pm 0.6}$
Increased	$81.9_{\pm 1.7}$	$77.8_{\pm 0.9}$	$56.3_{\pm 1.0}$	$44.0_{\pm 4.8}$