

Attribute-Enhanced Similarity Ranking for Sparse Link Prediction

João Mattos*
jrm28@rice.edu
Rice University
Houston, Texas, USA

Zexi Huang
zexi_huang@cs.ucsb.edu
UC Santa Barbara
Santa Barbara, California, USA

Mert Kosan†
mertkosan@ucsb.edu
UC Santa Barbara
Santa Barbara, California, USA

Ambuj Singh
ambuj@cs.ucsb.edu
UC Santa Barbara
Santa Barbara, California, USA

Arlei Silva‡
arlei@rice.edu
Rice University
Houston, Texas, USA

Abstract

Link prediction is a fundamental problem in graph data. In its most realistic setting, the problem consists of predicting missing or future links between random pairs of nodes from the set of disconnected pairs. Graph Neural Networks (GNNs) have become the predominant framework for link prediction. GNN-based methods treat link prediction as a binary classification problem and handle the extreme class imbalance—real graphs are very sparse—by sampling (uniformly at random) a balanced number of disconnected pairs not only for training but also for evaluation. However, we show that the reported performance of GNNs for link prediction in the balanced setting does not translate to the more realistic imbalanced setting and that simpler topology-based approaches are often better at handling sparsity. These findings motivate Gelato, a similarity-based link-prediction method that applies (1) graph learning based on node attributes to enhance a topological heuristic, (2) a ranking loss for addressing class imbalance, and (3) a negative sampling scheme that efficiently selects hard training pairs via graph partitioning. Experiments show that Gelato outperforms existing GNN-based alternatives.

CCS Concepts

- **Computing methodologies** → **Machine learning algorithms**;
- **Information systems** → *Social networks*.

Keywords

Link Prediction; Graph Neural Networks; Graph Learning; Topological Heuristics

ACM Reference Format:

João Mattos, Zexi Huang, Mert Kosan, Ambuj Singh, and Arlei Silva. 2025. Attribute-Enhanced Similarity Ranking for Sparse Link Prediction. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1 (KDD '25)*, August 3–7, 2025, Toronto, ON, Canada. ACM, New York, NY, USA, 12 pages. <https://doi.org/10.1145/3690624.3709314>

1 Introduction

Machine learning on graphs supports various structured-data applications including social network analysis [42, 64, 74], recommender systems [33, 52, 77], natural language processing [66, 72, 86], and physics modeling [17, 32, 68]. Among the graph-related tasks, one could argue that link prediction, which consists of predicting missing or future links [47, 50], is the most fundamental one. This is because link prediction not only has many concrete applications [45, 62] but can also be considered an (implicit or explicit) step of the graph-based machine learning pipeline [2, 49, 80]—as the observed graph is usually noisy and/or incomplete.

Graph Neural Networks (GNNs) [26, 40, 76] have emerged as the predominant paradigm for machine learning on graphs. Similar to their great success in node classification [41, 81, 96] and graph classification [28, 53, 87, 91], GNNs have been shown to achieve state-of-the-art link prediction performance [10, 46, 59, 79, 89, 90]. Compared to classical approaches that rely on expert-designed heuristics to extract topological information (e.g., Common Neighbors [55], Adamic-Adar [1], Preferential Attachment [4]), GNNs can naturally incorporate attributes and are believed to be able to learn new effective heuristics directly from data via supervised learning.

However, we argue that *the evaluation of GNN-based link prediction methods paints an overly optimistic view of their model performance*. Most real graphs are sparse and have a modular structure [3, 54]. In CORA and CITESEER (citation networks), less than 0.2% of the node pairs are links/positive (see Table 1) and modules arise around research topics. Yet GNN-based link prediction methods are evaluated on an artificially balanced test set that includes every positive pair but only a small sample of the negative ones chosen uniformly at random [27]. Due to modularity, the majority of negative pairs sampled are expected to be relatively far from each other (i.e. across different modules) compared to positive pairs. As a consequence, performance metrics reported for this balanced setting, which we call *biased testing*, differ widely from the ones observed for the more challenging *unbiased testing*, where the test set includes every disconnected pair of nodes. In particular, we have

*Corresponding Author.

†Work done prior to joining Visa Inc.

‡Also affiliated to the Ken Kennedy Institute at Rice University

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

KDD '25, Toronto, ON, Canada

© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-1245-6/25/08
<https://doi.org/10.1145/3690624.3709314>

found that unsupervised topological heuristics are more competitive in the *unbiased setting*, often outperforming recent GNN-based link prediction methods. This finding has motivated us to rethink the design of link prediction methods for sparse graphs.

A key hypothesis of our work is that effective unbiased link prediction in sparse graphs requires a similarity metric that can distinguish positive pairs from hard negative ones. More specifically, link prediction should be seen as a “needle in the haystack” type of problem, where extreme class imbalance makes even the most similar pairs still more likely to be negative. Existing GNN-based approaches fail in this sparse regime due to (1) their use of a binary classification loss that is highly sensitive to class imbalance; (2) their *biased training* that mimics *biased testing*; (3) their inability to learn effective topological heuristics directly from data.

The goal of this paper is to address the key limitations of GNNs for link prediction mentioned above. We present *Gelato*, a novel similarity-based framework for link prediction that combines a topological heuristic and graph learning to leverage both topological and attribute information. *Gelato* applies a ranking-based N-pair loss and partitioning-based negative sampling to select hard training node pairs. Extensive experiments demonstrate that our model significantly outperforms state-of-the-art GNNs in both accuracy and scalability. Figure 1 provides an overview of our approach.

To summarize, our contributions are: (1) We scrutinize the evaluation of supervised link prediction methods and identify their limitations in handling class imbalance; (2) we propose a simple, effective, and efficient framework to combine topological and attribute information for link prediction in an innovative fashion; (3) we introduce an N-pair link prediction loss that we show to be more effective at addressing class imbalance; and (4) we propose an efficient partitioning-based negative sampling scheme that improves link prediction generalization in the sparse setting.

2 Limitations in Supervised Link Prediction Evaluation

Supervised link prediction is often formulated as binary classification, where the positive (or negative) class includes node pairs connected (or not connected) by a link. A key difference between link prediction and other classification problems is that the two classes in link prediction are *extremely* imbalanced as most graphs of interest are sparse—e.g. the datasets from Table 1 are significantly more imbalanced than those in [75]. However, the class imbalance is not properly addressed in the evaluation of existing approaches.

Existing link prediction methods [8, 11, 14, 39, 59, 83, 90, 92, 99] are evaluated on a test set containing all positive test pairs and only an equal number of random negative pairs. Similarly, the Open Graph Benchmark (OGB) ranks predicted links against a very small sample of random negative pairs. We term these approaches *biased testing* as they highly overestimate the ratio of positive pairs in the graph. This issue is exacerbated in most real graphs, where community structure [56] causes random negative pairs to be particularly easy to identify [43]—they likely involve members of different communities. Evaluation metrics based on biased testing provide an overly optimistic assessment of the performance in *unbiased testing*, where every negative pair is included in the test set. In fact, in real

applications where positive test edges are not known a priori, it is impossible to construct those biased test sets to begin with.

Regarding evaluation metrics, Area Under the Receiver Operating Characteristic Curve (AUC) and Average Precision (AP) are the two most popular evaluation metrics for supervised link prediction [8, 11, 14, 39, 59, 83, 90, 92, 99]. We first argue that, as in other imbalanced classification problems [18, 67], AUC is not an effective evaluation metric for link prediction as it is biased towards the majority class (non-edges). On the other hand, AP and other rank-based metrics such as Hits@ k —used in OGB [27]—are effective for imbalanced classification *but only if evaluated on an unbiased test*.

Example: Consider an instance of Stochastic Block Model (SBM) [35] with 10 blocks of size 1k, intra-block density 0.9, and inter-block density 0.1. The number of inter-block negative pairs is $10 \times 1k \times (10 - 1) \times 1k \times (1 - 0.1)/2 = 40.5M$, while the number of intra-block negative pairs, which have high topological similarities like the ground-truth positive pairs and are much harder to contrast against, is $10 \times 1k \times 1k \times (1 - 0.9)/2 = 0.5M$. Biased testing would select less than $0.5M/(0.5M + 40.5M) < 2\%$ of the test negative pairs among the (hard) intra-block ones. In this scenario, even a random classifier is expected to obtain 50% precision. However, the expected precision drops to less than 22% (9M positive pairs vs. 41M negative pairs) under *unbiased testing*.

We will formalize the argument used in the example above by performing link prediction on a generic instance of the SBM with intra-block density p , inter-block density q , where $p > q$, and k blocks of size n . In particular, we will consider an instance of SBM corresponding to the expected node pattern given the parameters, where a node is connected to $(n - 1)p$ other nodes within its block and $(nk - n)q$ nodes outside its block. In this setting, the optimal link prediction algorithm can only distinguish potential links within or across blocks—as pairs within each set are connected with probability p and q , respectively.

LEMMA 1. *The ratio α between inter-cluster and intra-cluster negative node pairs in the SBM is such that:*

$$\alpha \geq (k - 1) \frac{1 - q}{1 - p}$$

The above lemma follows directly from the definition of the SBM and shows that the set of negative pairs is dominated by (easy) inter-cluster pairs as p increases compared to q .

THEOREM 2.1. *In the unbiased setting, the optimal accuracy link prediction method based on binary classification for the SBM predicts no links if $p < 0.5$.*

The proof is given in the Appendix B. Intuitively, even if the classifier has access to the SBM block structure, most within-block pairs are disconnected and thus the accuracy is maximized if no links are predicted. On the other hand, if $p > q$, an effective link prediction method should be able to leverage the SBM block structure to predict within block links. This motivates our formulation of link prediction as a “needle in the haystack” type of problem, where even the top candidate links (i.e., within-block pairs) are still more likely to be negative due to the sparsity of the graph.

LEMMA 2. *In the biased setting, there exist non-trivial link prediction methods with optimal accuracy based on binary classification for the SBM with $p < 0.5$.*

The proof is given in the Appendix C. The idea is that in the biased setting, a link prediction method that predicts within-block pairs as links can outperform the trivial classifier described in Theorem 2.1. This illustrates how biased testing, which is applied by recent work on supervised link prediction, can be misleading for sparse graphs. More specifically, a model trained under the biased setting might perform poorly if evaluated in the, more realistic, unbiased setting due to possibly unforeseen distribution shifts across the settings. This is a key motivation for our work.

The above discussion motivates a more representative evaluation setting for supervised link prediction. We argue for the use of rank-based evaluation metrics—AP, Precision@ k [47], and Hits@ k [5]—with *unbiased testing*, where positive edges are ranked against hard negative node pairs. These metrics have been widely applied in related problems, such as unsupervised link prediction [30, 47, 57, 93], knowledge graph completion [5, 73, 84], and information retrieval [69], where class imbalance is also significant. In our experiments, we will illustrate how these evaluation metrics combined with *unbiased testing* provide a drastically different and more informative performance evaluation compared to existing approaches.

3 Method

The limitations of supervised link prediction methods, including GNNs, to handle *unbiased testing* in sparse graphs motivate the design of a novel link prediction approach. First, preliminary results (see Table 2) have shown that topological heuristics are not impacted by class imbalance. That is because these heuristics are sensitive to small differences in structural similarity between positive and hard negative pairs while not relying on any learning—and thus not being affected by biased training. However, local structure proximity heuristics, such as Common Neighbors, are known to be less efficient in highly sparse scenarios observed in many real-world applications [48]—Table 1 shows the sparsity of our datasets. Further, unlike GNNs, topological heuristics are unable to leverage attribute information. Our approach addresses these limitations by integrating supervision into a powerful topological heuristic to leverage attribute data via graph learning.

Notation and problem. Consider an attributed graph $G = (V, E, X)$, where V is the set of n nodes, E is the set of m edges (links), and $X = (x_1, \dots, x_n)^T \in \mathbb{R}^{n \times r}$ collects r -dimensional node attributes. The topological (structural) information of the graph is represented by its adjacency matrix $A \in \mathbb{R}^{n \times n}$, with $A_{uv} > 0$ if an edge of weight A_{uv} connects nodes u and v and $A_{uv} = 0$, otherwise. The (weighted) degree of node u is given as $d_u = \sum_v A_{uv}$ and the corresponding degree vector (matrix) is denoted as $d \in \mathbb{R}^n$ ($D \in \mathbb{R}^{n \times n}$). The volume of the graph is $\text{vol}(G) = \sum_u d_u$. Our goal is to infer missing links in G based on its topological and attribute information, A and X .

Model overview. Figure 1 provides an overview of our model. It starts by selecting training node pairs using a novel partitioning-based negative sampling scheme. Next, a topology-centric graph learning phase incorporates node attribute information directly into the graph structure via a Multi-layer Perceptron (MLP). We then apply a topological heuristic, Autocovariance (AC), to the attribute-enhanced graph to obtain a pairwise score matrix. Node pairs with the highest scores are predicted as links. The scores for training pairs are collected to compute an N-pair loss. Finally, the loss is

used to train the MLP parameters in an end-to-end manner. We name our model Gelato (Graph enhancement for link prediction with autocovariance). Gelato represents a different paradigm in supervised link prediction combining a graph encoding of attributes with a topological heuristic instead of relying on node embeddings. While the building blocks of Gelato have been proposed by previous work, our paper is the first to apply these building blocks to address challenges in supervised link prediction for sparse graphs.

3.1 Graph learning

The goal of graph learning is to generate an enhanced graph that incorporates node attribute information into the topology. This can be considered as the “dual” operation of message-passing in GNNs, which incorporates topological information into attributes (embeddings). We propose graph learning as a more suitable scheme to combine attributes and topology for link prediction since it does not rely on the GNN to learn a topological heuristic, which we have verified empirically to be a challenge.

Specifically, our first step of graph learning is to augment the original edges with a set of node pairs based on their (untrained) attribute similarity (i.e., adding an ϵ -neighborhood graph):

$$\tilde{E} = E + \{(u, v) \mid s(x_u, x_v) > \epsilon_\eta\} \quad (1)$$

where $s(\cdot)$ can be any similarity function (we use cosine in our experiments) and ϵ_η is a threshold that determines the number of added pairs as a ratio η of the original number of edges m .

A simple MLP then maps the pairwise node attributes into a trained edge weight for every edge in \tilde{E} :

$$w_{uv} = \text{MLP}([x_u; x_v]; \theta) \quad (2)$$

where $[x_u; x_v]$ denotes the concatenation of x_u and x_v and θ contains the trainable parameters. For undirected graphs, we instead use the following permutation invariant operator [13]:

$$w_{uv} = \text{MLP}([x_u + x_v; |x_u - x_v|]; \theta) \quad (3)$$

The final weights of the enhanced graph are a combination of the topological, untrained, and trained weights:

$$\tilde{A}_{uv} = \alpha A_{uv} + (1 - \alpha)(\beta w_{uv} + (1 - \beta)s(x_u, x_v)) \quad (4)$$

where α and β are hyperparameters. The enhanced adjacency matrix \tilde{A} is then fed into a topological heuristic for link prediction introduced in the next section. The MLP is not trained directly to predict the links but instead trained end-to-end to enhance the input graph given to the topological heuristic. Further, the MLP can be easily replaced by a more powerful model such as a GNN, but the goal of this paper is to demonstrate the general effectiveness of our framework and we will show that even a simple MLP leads to significant improvement over the base heuristic.

3.2 Topological heuristic

Assuming that the learned adjacency matrix \tilde{A} incorporates structural and attribute information, Gelato applies a topological heuristic to \tilde{A} . Specifically, we generalize Autocovariance, which has been shown to be effective for non-attributed graphs [30], to the attributed setting. Autocovariance is a random-walk-based similarity metric. Intuitively, it measures the difference between the co-visiting probabilities for a pair of nodes in a truncated walk

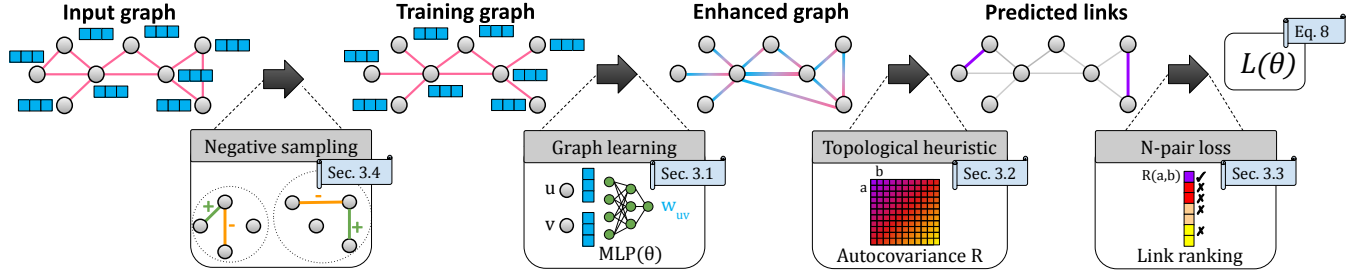


Figure 1: Gelato applies graph learning to incorporate attribute information into the topology. The learned graph is given to a topological heuristic that predicts edges between node pairs with high Autocovariance similarity. The parameters of the MLP are optimized end-to-end using the N-pair loss over node pairs selected via a partitioning-based negative sampling scheme. Experiments show that Gelato outperforms state-of-the-art GNN-based link prediction methods.

and in an infinitely long walk. Given the enhanced graph \tilde{G} , the Autocovariance similarity matrix $R \in \mathbb{R}^{n \times n}$ is given as

$$R = \frac{\tilde{D}}{\text{vol}(\tilde{G})} (\tilde{D}^{-1} \tilde{A})^t - \frac{\tilde{d} \tilde{d}^T}{\text{vol}^2(\tilde{G})} \quad (5)$$

where $t \in \mathbb{N}_0$ is the scaling parameter of the truncated walk. Each entry R_{uv} represents a similarity score for node pair (u, v) , and top similarity pairs are predicted as links. Note that R_{uv} only depends on the t -hop enclosing subgraph of (u, v) and can be easily differentiated with respect to the edge weights in the subgraph. Gelato could be applied with any differentiable topological heuristics or even a combination of them. In our experiments (Section 4.3), we will show that Autocovariance alone enables state-of-the-art link prediction without requiring any learning.

Autocovariance versus other heuristics. Following [48], we show that local structural heuristics commonly employed by GNNs, such as Common Neighbors, exhibit reduced efficacy in sparse networks with less informative neighborhood structures. This observation motivates our selection of Autocovariance as our topological heuristic, given its ability to capture global structural patterns through random walks. Further, the parameter t in Autocovariance offers adaptability to varying network sparsity levels[48], ranging from denser (lower t values) to sparser (higher t values) networks. **Autocovariance distinguishes negative pairs.** Autocovariance can be seen as a general case of the Modularity metric Q [19]:

$$Q = \frac{1}{4m} \sum_{ij} (A_{ij} - \frac{d_i d_j}{2m}) s_i s_j, \quad (6)$$

in which $m = \text{vol}(G)/2$, d_i and d_j are the degrees of nodes i and j , and $s_i s_j$ is a product that indicates whether both nodes are in the same partition. More specifically, for $t = 1$, Autocovariance expresses the graph partitioning resulting in the optimal Modularity value, which captures the relationship between the expected number of edges between two partitions compared to the probability of any random edge in the graph. This key property directly applies to our scenario, enabling Gelato to distinguish between hard (same partitions) and easy (different partitions) negative pairs and motivating us to adopt Autocovariance as our graph heuristic. Further, as t increases, Autocovariance expresses growing coarser partitions

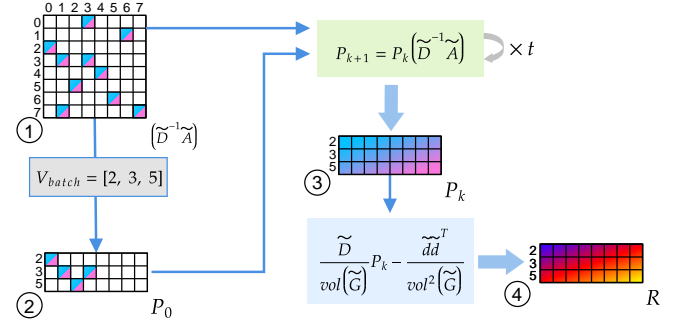


Figure 2: Scaling up Gelato using batching and sparse tensors. We represent sparse tensors (1 and 2) as matrices with blank entries and dense tensors (3 and 4) as color-filled matrices. We extract from the enhanced transition matrix (1) a slice P_0 (2) given a batch of node indices V_{batch} . Instead of a matrix exponentiation, we compute $P_0 (\tilde{D}^{-1} \tilde{A})$ repeatedly for t times to obtain P_k (3), a dense tensor. Finally, we use P_k to obtain the autocovariance R (4) for nodes in the batch. This is implemented efficiently using dense-sparse tensor multiplication.

until approximating spectral clustering (for $t \rightarrow \infty$), being flexible regarding partition sizes according to different domains.

Scaling up Gelato with batching and sparse operations. Naively implementing Gelato using dense tensors is infeasible, due to the quadratic VRAM requirement ($R \in \mathbb{R}^{|V| \times |V|}$). To address this limitation, we propose storing \tilde{A} as a sparse matrix. Then, instead of directly computing $(\tilde{D}^{-1} \tilde{A})^t$ from Equation 5 (resulting on a dense $|V| \times |V|$ matrix), we compute

$$P_{l+1} = P_l (\tilde{D}^{-1} \tilde{A}), \quad l \in \{1, 2, \dots, t\} \quad (7)$$

$$R = \frac{\tilde{D}}{\text{vol}(\tilde{G})} P_t - \frac{\tilde{d} \tilde{d}^T}{\text{vol}^2(\tilde{G})} \quad (8)$$

where $P_0 = (\tilde{D}^{-1} \tilde{A})_{ij}$, for all $i \in V_{batch}$, where V_{batch} consists of the nodes in the current batch. This operation substitution allows us to compute a sequence of t multiplications between a dense

$P_k \in \mathbb{R}^{|batch| \times |V|}$ matrix and a sparse matrix $(\tilde{D}^{-1}\tilde{A}) \in \mathbb{R}^{|V| \times |V|}$ instead of a dense matrix power operation, $(\tilde{D}^{-1}\tilde{A})^t$. The overall VRAM usage is reduced from $O(|V|^2)$ to $O(|batch| \cdot |V|)$.

3.3 N-pair loss

Supervised link prediction methods rely on the cross entropy loss (CE) to optimize model parameters. However, CE is known to be sensitive to class imbalance [7]. Instead, Gelato leverages the N-pair loss [71] that is inspired by the metric learning and learning-to-rank literature [9, 51, 65, 78] to train the parameters of our graph learning model from highly imbalanced *unbiased training* data.

The N-pair loss (NP) contrasts each positive training edge (u, v) against a set of negative pairs $N(u, v)$. It is computed as follows:

$$L(\theta) = - \sum_{(u,v) \in E} \log \frac{\exp(R_{uv})}{\exp(R_{uv}) + \sum_{(p,q) \in N(u,v)} \exp(R_{pq})} \quad (9)$$

Intuitively, $L(\theta)$ is minimized when each positive edge (u, v) has a much higher similarity than its contrasted negative pairs: $R_{uv} \gg R_{pq}, \forall (p, q) \in N(u, v)$. Compared to CE, NP is more sensitive to negative pairs that have comparable similarities to those of positive pairs—they are more likely to be false positives. While NP achieves good performance in our experiments, alternative losses from the learning-to-rank literature [6, 23, 82] could also be applied.

3.4 Negative sampling

Supervised methods for link prediction sample a small number of negative pairs uniformly at random but most of these pairs are expected to be easy (see Section 2). To minimize distribution shifts between training and test, negative samples $N(u, v)$ should ideally be generated using *unbiased training* (see additional example in Appendix A). This means that $N(u, v)$ is a random subset of all disconnected pairs in the training graph, and $|N(u, v)|$ is proportional to the ratio of negative pairs. In this way, we enforce $N(u, v)$ to include hard negative pairs. However, due to graph sparsity (see Table 1), this approach does not scale to large graphs as the total number of negative pairs would be $O(|V|^2 - |E|)$.

LEMMA 3. *Let a Stochastic Block Model with intra-block density p , inter-block density q , and $p > q$. Then the expected Autocovariance of intra-block pairs (R_{intra}) is greater than the expected Autocovariance of inter-block pairs R_{inter} , i.e. $\mathbb{E}[R_{intra}] > \mathbb{E}[R_{inter}]$.*

LEMMA 4. *Let a Stochastic Block Model with intra-block density p , inter-block density q , and $p > q$. Then, $\mathbb{E}[R_{intra}]$ monotonically increases as the number of partitions increases.*

Considering Lemma 3 (see proof in the Appendix D), we argue that it is unlikely for an inter-block pair to be ranked within the top Autocovariance pairs, implying that removing these pairs from training would not affect the results. To efficiently generate a small number of hard negative pairs, we propose a novel negative sampling scheme for link prediction based on graph partitioning [16, 21]. The idea is to select negative samples inside partitions (or communities) as they are expected to have similarity values comparable to positive pairs. We adopt METIS [36] as our graph partitioning method due to its scalability and its flexibility to generate partitions of a size given as a parameter. METIS' partitions are expected to be densely connected inside and sparsely connected

across (partitions). We apply METIS to obtain k partitions in which $\forall i \in \{1, 2, \dots, k\} : G_i = (V_i, E_i, X_i), V_i \subset V, E_i \subset E, X_i \subset X$, such that $V = \bigcup_{i=1}^k V_i$ and $|V_i| \approx |V|/k$. Then, we apply *unbiased training* only *within each partition*, reducing the number of sampled negative pairs to $|E^-| = \sum_i^k |V_i|^2 - |E_i|$. Following Lemma 4 (see proof on Appendix E), the choice of the value of k should consider a trade-off between training speed and link prediction performance (see Appendix 4.3). Further, the algorithm proposed by [56] could be adopted to find the optimal value of k that maximizes the Modularity gain while obtaining the minimal training time. In the remainder of the paper, we refer to this approach as *partitioned training*. We claim that this procedure filters (easy) pairs consisting of nodes that would be too far away in the network topology from training while maintaining the more informative (hard) pairs that are closer and topologically similar, according to METIS. We include in the Appendix 4.3 (See Figure 5) a performance comparison between Gelato trained using *unbiased training* against *partitioned training*.

4 Experiments

In this section, we provide empirical evidence supporting our claims about supervised link prediction, demonstrate the accuracy and efficiency of Gelato, and present ablation studies. The implementation is available on Anonymous GitHub¹.

4.1 Experiment settings

Datasets. Our method is evaluated on five attributed graphs commonly used for link prediction [11, 14, 27, 59, 83, 92, 99]. Table 1 shows dataset statistics.

Table 1: A summary of dataset statistics.

	#Nodes	#Edges	#Attrs	Avg. degree	Density
CORA	2,708	5,278	1,433	3.90	0.14%
CITESEER	3,327	4,552	3,703	2.74	0.08%
PUBMED	19,717	44,324	500	4.50	0.02%
OGBL-DDI	4,267	1,334,889	0	500.5	7.33%
OGBL-COLLAB	235,868	1,285,465	128	8.2	0.0046%

Baselines. For GNN-based link prediction, we include four state-of-the-art methods published in the past two years: Neo-GNN [89], BUDDY [10], and NCN / NCNC [79], as well as the pioneering work—SEAL [90]. For topological link prediction heuristics, we consider Common Neighbors (CN) [55], Adamic Adar (AA) [1], Personalized PageRank (PPR) [58], and Autocovariance (AC) [30]—the base heuristic in our model. For comparison with older baselines, please refer to an earlier preprint version of our paper [29].

Hyperparameters. For Gelato, we tune the proportion of added edges η from $\{0.0, 0.25, 0.5, 0.75, 1.0\}$, the topological weight α from $\{0.0, 0.25, 0.5, 0.75\}$, and the trained weight β from $\{0.25, 0.5, 0.75, 1.0\}$. All other settings are fixed across datasets: MLP with one hidden layer of 128 neurons, AC scaling parameter $t = 3$, Adam optimizer [38] with a learning rate of 0.001, a dropout rate of 0.5, and *unbiased training* without downsampling. To maintain fairness in our results, we also tuned the baselines and exposed our procedures in detail

¹<https://anonymous.4open.science/r/Gelato/>

in Appendix F. For all models, including Gelato, the tuning process is done in all datasets, except for OGBL-COLLAB.

Data splits for unbiased training and unbiased testing. Following [11, 14, 39, 59, 90, 92], we adopt 85%/5%/10% ratios for training, validation, and testing. Specifically, for *unbiased training* and *unbiased testing*, we first randomly divide the (positive) edges E of the original graph into E_{train}^+ , E_{valid}^+ , and E_{test}^+ for training, validation, and testing based on the selected ratios. Then, we set the negative pairs in these three sets as (1) $E_{train}^- = E^- + E_{valid}^+ + E_{test}^+$, (2) $E_{valid}^- = E^- + E_{test}^+$, and (3) $E_{test}^- = E^-$, where E^- is the set of all negative pairs (excluding self-loops) in the original graph. Notice that the validation and testing *positive* edges are included in the *negative* training set, and the testing *positive* edges are included in the *negative* validation set. This setting simulates the real-world scenario where the test edges (and the validation edges) are unobserved during validation (training). For *negative sampling*, we repeat the dividing procedure above for each generated partition G_i . The final sets are unions of individual sets for each partition: $E_{train}^{+/-} = \bigcup_{i=1}^k E_{train,i}^{+/-}$, $E_{valid}^{+/-} = \bigcup_{i=1}^k E_{valid,i}^{+/-}$, and $E_{test}^{+/-} = \bigcup_{i=1}^k E_{test,i}^{+/-}$. We notice that these splits do not leak training data to the test, as both positive and negative test pairs are disconnected during training. **Evaluation metrics.** We adopt $hits@k$ —the ratio of positive edges individually ranked above k th place against all negative pairs—as our evaluation metric since it represents a good notion of class distinction under heavily imbalanced scenarios in information retrieval, compatible with the intuition of link prediction as a similarity-based ranking task.

4.2 Partitioned sampling and link prediction as a similarity task

This section provides empirical evidence for some of the claims made regarding limitations in the evaluation of supervised link prediction methods (see Section 2). It also demonstrates the effectiveness of Gelato to distinguish true links from hard negative node pairs in sparse graphs.

Negative sampling for harder pairs. Based on the hardness of negative pairs, the easiest scenario is the *biased testing*, followed by *unbiased testing* and *partitioned testing*—i.e. only negative pairs from inside partitions are sampled. This can be verified by Figure 3, which compares the predicted scores of NCN against the similarities computed by Gelato on the test set of CITESEER. *Biased testing*, the easiest and most unrealistic scenario, shows a good separation between positive and negative pairs both in NCN and Gelato. For *unbiased testing*, which is more realistic, Gelato is better at distinguishing positive and negative pairs. Finally, *partitioned testing* presents a particular challenge but Gelato still ranks most positive pairs above negative ones. Other GNN-based link prediction approaches have shown similar behaviors to NCN.

Similarity-based link prediction. Figure 3 shows densities normalized by the size of positive and negative sets, respectively. However, in real-world sparse graphs, the number of negative pairs is much larger than that of positive ones. The results show that for *unbiased* and *partitioned testing*, ranking positive pairs over hard negative pairs is especially challenging due to their overwhelming number, i.e. positive pairs are “needles in a haystack”. This provides

evidence that classifiers, such as GNNs for link prediction, are not suitable for finding decision boundaries in these extremely imbalanced settings, which motivates the design of Gelato as a similarity ranking model trained using an N-pair loss.

4.3 Link prediction performance

Table 2 summarizes the link prediction performance in terms of the mean and standard deviation of $hits@1000$ for all methods. We show the same results for varying values of k in Figure 4.

First, we want to highlight the drastically different performance of GNN-based methods compared to those found in the original papers [10, 79, 89, 90]. Some of them underperform even the simplest topological heuristics such as Common Neighbors under *unbiased testing*. Moreover, Autocovariance, which is the base topological heuristic applied by Gelato and does not account for node attributes, outperforms all the GNN-based baselines for the majority of the datasets. These results support our arguments from Section 2 that evaluation metrics based on *biased testing* can produce misleading results compared to *unbiased testing*.

The overall best-performing GNN model is NCNC, which generalizes a pairwise topological heuristic (Common Neighbors) using message-passing. NCNC only outperforms Gelato on OGBL-DDI, which is consistent with previous results [48] showing that local structural heuristics are effective for very dense networks (see Table 1). Moreover, OGBL-DDI is the only dataset considered that does not contain natural node features, which explains why our approach achieves the same performance as AC. Gelato also remains superior for different values of $hits@K$, especially for CORA, CITESEER and OGBL-COLLAB, and being remains competitive for OGBL-DDI being competitive as shown in Figure 4. This characteristic is especially relevant in real-world scenarios where robustness is desired, mainly in more conservative regimes with lower values of k . Overall, Gelato outperforms the best GNN-based method by **138%**, **125%**, **156%**, and **11%** for CORA, CITESEER, PUBMED, and OGBL-COLLAB, respectively. Further, Gelato outperforms its base topological heuristic (Autocovariance) by **48%**, **39%**, **10%**, and **139%** for CORA, CITESEER, PUBMED, and OGBL-COLLAB, respectively.

Unbiased Training vs. Partitioned Training. Figure 5 compares *partitioned* and *unbiased* training on CiteSeer using $hits@K$ for varying K . Results show minimal performance differences, even in extreme partitioning scenarios, while *partitioned* sampling achieves a significant speedup—training up to 6x faster and scaling with the number of partitions. Unbiased training requires $O(V^2 - E)$ for sparse graphs due to extensive negative sampling, whereas *partitioned* sampling reduces this to $O(\sum_i^p |V_i|^2 - |E_i|)$, where (V_i, E_i) are nodes and edges within partition i . Experiments confirm that performance is largely insensitive to different values of p .

4.4 Ablation study

Here, we collect the results with the same hyperparameter setting as Gelato and present an ablation study in Table 3. *Gelato-MLP (AC)* represents Gelato without the MLP (Autocovariance) component, i.e., only using Autocovariance (MLP) for link prediction. *Gelato-NP (UT)* replaces the proposed N-pair loss (*unbiased training*) with the cross entropy loss (*biased training*) applied by the baselines. Finally, *Gelato-NP+UT* replaces both the loss and the training setting.

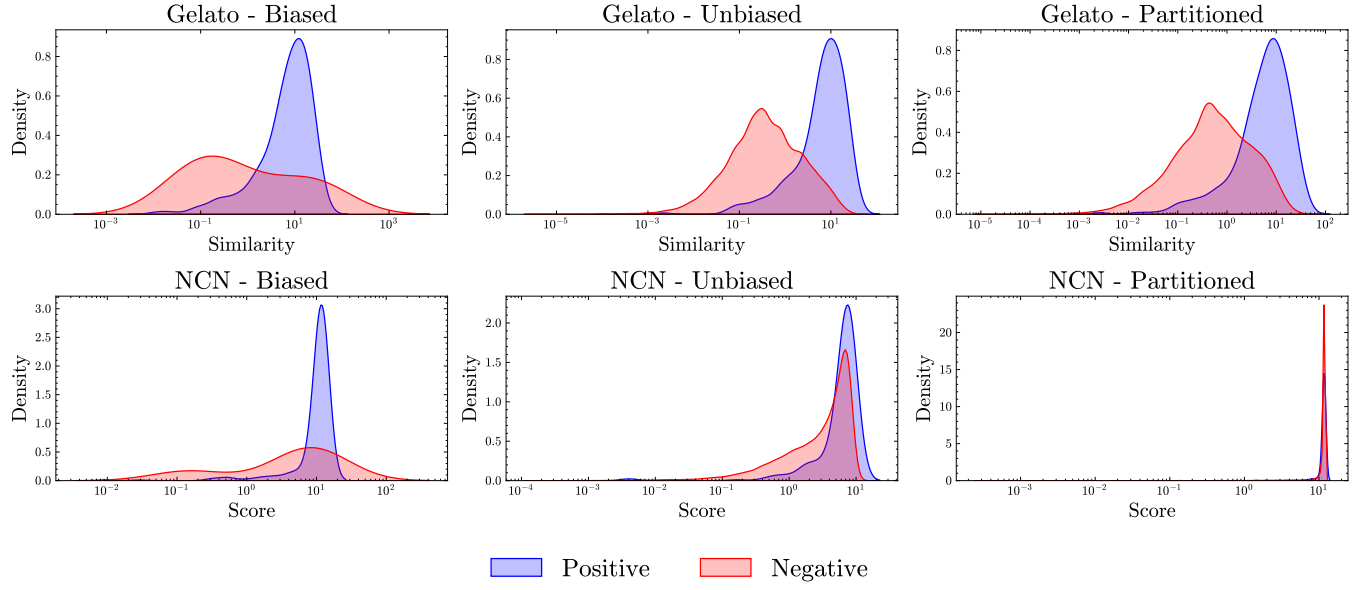


Figure 3: We analyze classification-based and similarity-based link prediction approaches through a comparison between the probability density functions of predicted similarities/scores by Gelato and NCN (state-of-the-art GNN), on the test set in three different regimes (biased, unbiased, and partitioned). Negative pairs are represented in **red**, and positive pairs are represented in **blue**. By treating link prediction as a similarity-based problem, Gelato presents better separation (smaller overlap) between the similarity curves in the harder scenarios, distinguishing between positive and negative pairs across all testing regimes. NCN presents a drastic increase in overlap as negative pairs become harder, struggling to separate positive and negative pairs.

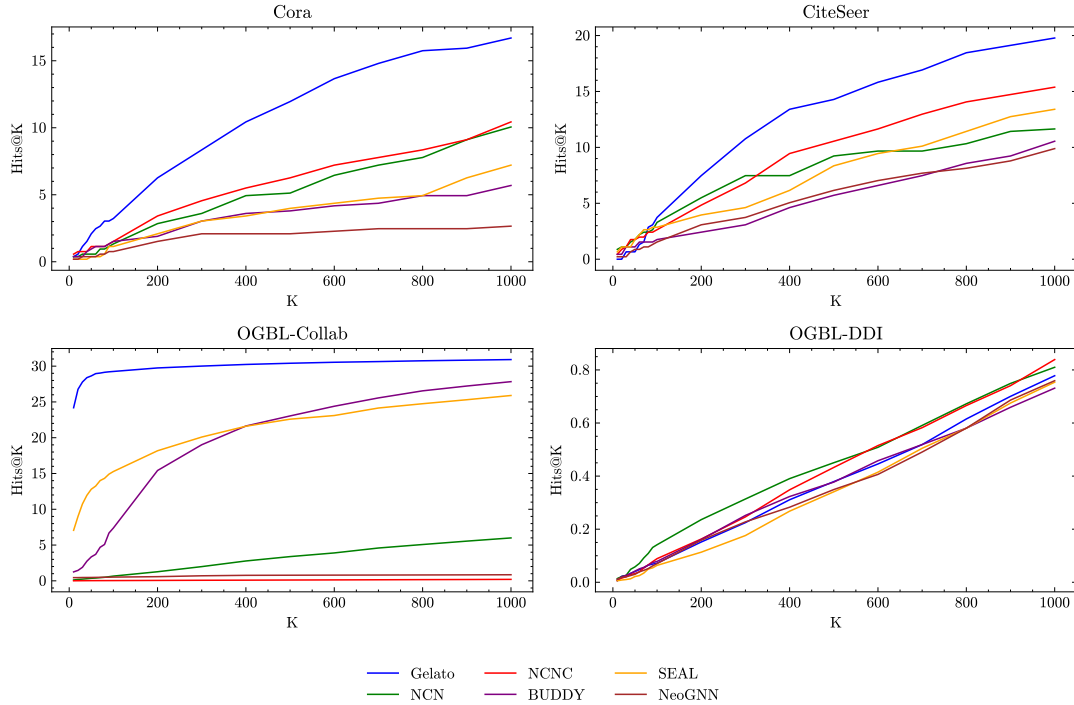


Figure 4: Link prediction comparison in terms of *hits@k* varying *k* using Cora, CiteSeer, OGBL-DDI and OGBL-Collab. All datasets were split using *unbiased* sampling, except OGBL-Collab, which was split using *partitioned* sampling. Gelato outperforms the baselines across different values of *k* and remains competitive on OGBL-DDI, a dataset in which all methods struggle.

Table 2: Link prediction performance comparison (mean \pm std hits@1000) for all datasets considered. Gelato consistently outperforms GNN-based methods, topological heuristics, and two-stage approaches combining attributes/topology. For CORA, CITESEER, OGBL-DDI and PUBMED results we used *unbiased* training, while for OGBL-COLLAB *partitioned* sampling is used, for scalability reasons. The top three models are colored by **First, **Second** and **Third**.**

		CORA	CITESEER	PUBMED	OGBL-DDI	OGBL-COLLAB
GNN	SEAL	0.0*	7.25*	***	0.75*	25.9*
	Neo-GNN	6.96 \pm 4.24	5.42 \pm 0.13	1.63 \pm 0.32	0.76*	0.85*
	BUDDY	4.81 \pm 0.72	5.86 \pm 0.34	OOM	0.74 \pm 0.01	27.66 \pm 0.24
	NCN	4.11 \pm 1.22	7.84 \pm 1.13	0.06 \pm 0.1	0.82 \pm 0.02	7.16 \pm 1.42
	NCNC	6.58 \pm 0.58	8.72 \pm 2.08	1.04 \pm 0.09	0.89 \pm 0.09	0.44 \pm 0.37
Topological Heuristics	CN	4.17 \pm 0.00	4.4 \pm 0.00	0.36 \pm 0.00	0.8 \pm 0.00	2.4 \pm 0.00
	AA	6.64 \pm 0.00	4.4 \pm 0.00	1.13 \pm 0.00	0.79 \pm 0.00	4.88 \pm 0.00
	PPR	9.30 \pm 0.00	6.59 \pm 0.00	0.32 \pm 0.00	0.08 \pm 0.00	1.24 \pm 0.00
	AC	11.20 \pm 0.00	14.29 \pm 0.00	3.81 \pm 0.00	0.78 \pm 0.00	12.89 \pm 0.00
Gelato		16.62 \pm 0.31	19.78 \pm 0.23	4.18 \pm 0.19	0.78 \pm 0.00	30.92*

* Run only once as each run takes >24 hrs; *** Each run takes >1000 hrs; OOM: Out Of Memory.

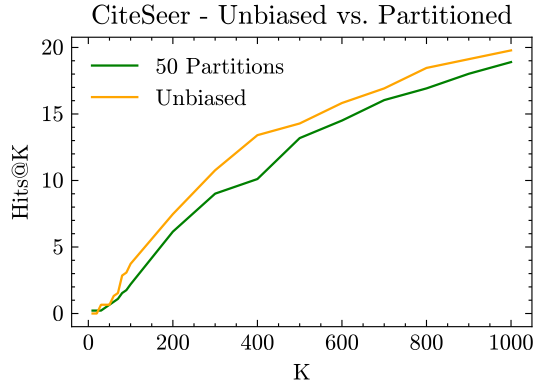


Figure 5: Comparison of *unbiased* vs. *partitioned* sampling on CiteSeer for different K . Even with extreme partitioning ($k = 50$, ≈ 66 nodes/partition), performance differences are minor, while *partitioned* sampling trains significantly faster.

We observe that removing either MLP or Autocovariance leads to inferior performance, as the corresponding attribute or topology information would be missing. Further, to address the class imbalance problem of link prediction, both the N-pair loss and *unbiased training* are crucial for the effective training of Gelato.

We also present results for Gelato using different ranking-based loss functions. We choose between Precision@k, pairwise hinge, pairwise exponential, and pairwise logistic losses as candidates for replacing the N-pair loss based on [12]. The results are shown in Table 4, demonstrating that there is no clear winner considering the hits@1000 metric in the two datasets used (CORA and CITESEER).

We analyze hyperparameter sensitivity and its impact on performance. Figure 6 shows that optimal α and β values depend on dataset characteristics, though performance varies smoothly for Cora and CiteSeer, easing tuning. Figure 7 highlights diminishing returns from increasing η , as performance plateaus.

Table 3: Results of the ablation study based on hits@1000 scores. Each component of Gelato plays an important role in enabling state-of-the-art link prediction performance.

	CORA	CITESEER	PUBMED
<i>Gelato-MLP</i>	16.13 \pm 0.00	19.78 \pm 0.00	3.81 \pm 0.0
<i>Gelato-AC</i>	2.66 \pm 2.57	12.6 \pm 0.71	0.0 \pm 0.0
<i>Gelato-NP+UT</i>	16.32 \pm 0.19	19.41 \pm 0.34	4.05 \pm 0.12
<i>Gelato-NP</i>	16.51 \pm 0.19	17.88 \pm 0.46	1.74 \pm 0.14
<i>Gelato</i>	16.62 \pm 0.31	19.89 \pm 0.23	4.18 \pm 0.19

Table 4: Comparison between N-pair loss (Gelato) against the Precision@K (PK), pairwise hinge (PH), pairwise exponential (PE), and pairwise logistic (PL) losses considering the hits@1000 metric.

	CORA	CITESEER
<i>Gelato-PK</i>	16.32 \pm 0.19	19.19 \pm 0.99
<i>Gelato-PH</i>	18.09 \pm 0.48	16.56 \pm 0.13
<i>Gelato-PE</i>	16.82 \pm 0.48	15.9 \pm 0.34
<i>Gelato-PL</i>	18.03 \pm 0.38	17.14 \pm 0.66
<i>Gelato</i>	16.62 \pm 0.31	19.89 \pm 0.24

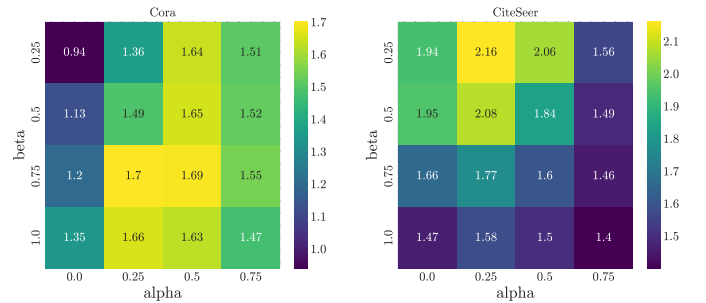


Figure 6: Sensitivity analysis of α and β considering AP.

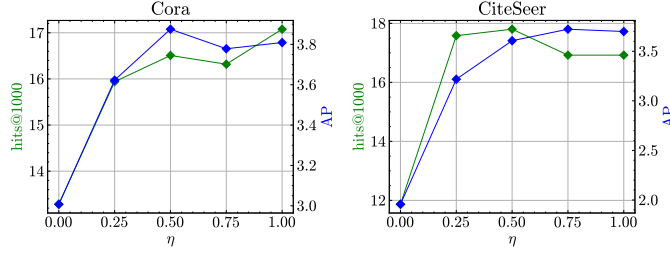


Figure 7: Performance of Gelato with different values of η , using $T = 3$. We represent hits@1000 in green and AP in blue.

5 Related Work

An earlier preprint version of our paper is available at arXiv [29]. **Topological heuristics for link prediction.** The early link prediction literature focuses on topology-based heuristics. This includes approaches based on local (e.g., Common Neighbors [55], Adamic Adar [1], and Resource Allocation [98]) and higher-order (e.g., Katz [37], PageRank [58], and SimRank [34]) information. More recently, random-walk based graph embedding methods, which learn vector representations for nodes [24, 30, 61], have achieved promising results in graph machine learning tasks. Popular embedding approaches, such as DeepWalk [61] and node2vec [24], have been shown to implicitly approximate the Pointwise Mutual Information similarity [63], which can also be used as a link prediction heuristic. This has motivated the investigation of other similarity metrics such as Autocovariance [19, 30, 31]. However, these heuristics are unsupervised and cannot take advantage of data beyond the topology. **Graph Neural Networks for link prediction.** GNN-based link prediction addresses the limitations of topological heuristics by training a neural network to combine topological and attribute information and potentially learn new heuristics. These works often assume that links are correlated with homophily in node attributes [20, 97], as also is the case for this paper. GAE [39] combines a graph convolution network [40] and an inner product decoder based on node embeddings. SEAL [90] models link prediction as a binary subgraph classification problem (edge/non-edge), and follow-up work (e.g., SHHF [46], WalkPool [59]) investigates different pooling strategies. Other recent approaches for GNN-based link prediction include learning representations in hyperbolic space (e.g., HGCN [11], LGCN [92]), generalizing topological heuristics (e.g., NeoGNN [89], NBFNet [99]), and incorporating additional topological features (e.g., TLC-GNN [83], BScNets [14]). ELPH and BUDDY [10] apply hashing to efficiently approximate subgraph-based link prediction models, such as SEAL, using a message-passing neural network (MPNN) with distance-based structural features. NCNC [79] combines the Common Neighbors heuristic with an MPNN achieving state-of-the-art results. Motivated by the growing popularity of GNNs for link prediction, this work investigates key questions regarding their training, evaluation, and ability to learn effective topological heuristics directly from data. We propose Gelato, which is simpler, more accurate, and faster than most state-of-the-art GNN-based link prediction methods.

Graph learning. Gelato learns a graph that combines topological and attribute information. Our goal differs from generative

models [25, 44, 88], which learn to sample from a distribution over graphs. Graph learning also enables the application of GNNs when the graph is unavailable, noisy, or incomplete [94]. LDS [22] and GAUG [95] jointly learn a probability distribution over edges and GNN parameters. IDGL [15] and EGLN [85] alternate between optimizing the graph and embeddings for node/graph classification and collaborative filtering. [70] proposes two-stage link prediction by augmenting the graph as a preprocessing step. In comparison, Gelato effectively learns a graph in an end-to-end manner by minimizing the loss of a topological heuristic.

6 Conclusion

This work exposes key limitations in evaluating supervised link prediction methods due to the widespread use of *biased testing*. These limitations led to a consensus in the graph machine learning community that (1) GNNs are superior for link prediction, casting topological heuristics obsolete; and (2) link prediction is now an easy task due to deep learning advances. We challenge both assumptions, demonstrating that link prediction in sparse graphs remains a hard problem when evaluated properly. GNNs struggle with link prediction in sparse graphs due to extreme class imbalance, motivating Gelato, our novel link prediction framework.

Gelato is a similarity-based method that combines graph learning and autocovariance to leverage attribute and topological information. Gelato employs an N-pair loss instead of cross-entropy to address the class imbalance and introduces a partitioning-based negative sampling scheme for efficient hard negative pair sampling. Through extensive experiments, we demonstrate superior accuracy and scalability of Gelato when compared to state-of-the-art GNN-based solutions across various datasets.

Acknowledgments

We gratefully acknowledge the support by the National Science Foundation (Grant No. IIS 2229876), the US Department of Transportation (USDOT) Tier-1 University Transportation Center (UTC) Transportation Cybersecurity Center for Advanced Research and Education (CYBER-CARE) (Grant No. 69A3552348332), and the Rice University Ken Kennedy Institute.

References

- [1] Lada A Adamic and Eytan Adar. 2003. Friends and neighbors on the web. *Social networks* 25, 3 (2003), 211–230.
- [2] Ashwin Bahulkar, Boleslaw K Szymanski, N Orkun Baycik, and Thomas C Sharkey. 2018. Community detection with edge augmentation in criminal networks. In *ASONAM*.
- [3] Albert-László Barabási. 2016. *Network Science*. Cambridge University Press.
- [4] Albert-László Barabási, Hawoong Jeong, Zoltan Neda, Erzsébet Ravasz, Andras Schubert, and Tamas Vicsek. 2002. Evolution of the social network of scientific collaborations. *Physica A: Statistical mechanics and its applications* 311, 3-4 (2002), 590–614.
- [5] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *NeurIPS*.
- [6] Sebastian Bruch. 2021. An alternative cross entropy loss for learning-to-rank. In *WebConf*.
- [7] Jonathon Byrd and Zachary Lipton. 2019. What is the effect of importance weighting in deep learning?. In *International conference on machine learning*. PMLR, 872–881.
- [8] Lei Cai, Jundong Li, Jie Wang, and Shuiwang Ji. 2021. Line graph neural networks for link prediction. *IEEE TPAMI* (2021).
- [9] Fatih Cakir, Kun He, Xide Xia, Brian Kulis, and Stan Sclaroff. 2019. Deep metric learning to rank. In *CVPR*.

- [10] Benjamin Paul Chamberlain, Sergey Shirobokov, Emanuele Rossi, Fabrizio Frasca, Thomas Markovich, Nils Hammerla, Michael M Bronstein, and Max Hamsire. 2023. Graph Neural Networks for Link Prediction with Subgraph Sketching. In *ICLR*.
- [11] Ines Chami, Zhitao Ying, Christopher Ré, and Jure Leskovec. 2019. Hyperbolic graph convolutional neural networks. In *NeurIPS*.
- [12] Wei Chen, Tie-Yan Liu, Yanyan Lan, Zhi-Ming Ma, and Hang Li. 2009. Ranking measures and loss functions in learning to rank. *Advances in Neural Information Processing Systems* 22 (2009).
- [13] Xu Chen, Xiuyuan Cheng, and Stéphane Mallat. 2014. Unsupervised deep haar scattering on graphs. In *NeurIPS*.
- [14] Yuzhou Chen, Yulia R Gel, and H Vincent Poor. 2022. BScNets: Block Simplicial Complex Neural Networks. In *AAAI*.
- [15] Yu Chen, Lingfei Wu, and Mohammed Zaki. 2020. Iterative deep graph learning for graph neural networks: Better and robust node embeddings. In *NeurIPS*.
- [16] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. 2019. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 257–266.
- [17] Arlei Lopes da Silva, Furkan Kocayusufoglu, Saber Jafarpour, Francesco Bullo, Ananthram Swami, and Ambuj Singh. 2020. Combining Physics and Machine Learning for Network Flow Estimation. In *ICLR*.
- [18] Jesse Davis and Mark Goadrich. 2006. The relationship between Precision-Recall and ROC curves. In *ICML*.
- [19] J-C Delvenne, Sophia N Yaliraki, and Mauricio Barahona. 2010. Stability of graph communities across time scales. *PNAS* 107, 29 (2010), 12755–12760.
- [20] Andrea Giuseppe Di Francesco, Francesco Caso, Maria Sofia Bucarelli, and Fabrizio Silvestri. 2024. Link Prediction under Heterophily: A Physics-Inspired Graph Neural Network Approach. *arXiv preprint arXiv:2402.14802* (2024).
- [21] Santo Fortunato. 2010. Community detection in graphs. *Physics reports* 486, 3-5 (2010), 75–174.
- [22] Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He. 2019. Learning discrete structures for graph neural networks. In *ICML*.
- [23] Yoav Freund, Raj Iyer, Robert E Schapire, and Yoram Singer. 2003. An efficient boosting algorithm for combining preferences. *JMLR* 4, Nov (2003), 933–969.
- [24] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *SIGKDD*.
- [25] Aditya Grover, Aaron Zweig, and Stefano Ermon. 2019. Graphite: Iterative generative modeling of graphs. In *ICML*.
- [26] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*.
- [27] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. In *NeurIPS*.
- [28] Zexi Huang, Mert Kosan, Sourav Medya, Sayan Ranu, and Ambuj Singh. 2023. Global counterfactual explainer for graph neural networks. In *WSDM*.
- [29] Zexi Huang, Mert Kosan, Arlei Silva, and Ambuj Singh. 2023. Link prediction without graph neural networks. *arXiv preprint arXiv:2305.13656* (2023).
- [30] Zexi Huang, Arlei Silva, and Ambuj Singh. 2021. A Broader Picture of Random-walk Based Graph Embedding. In *SIGKDD*.
- [31] Zexi Huang, Arlei Silva, and Ambuj Singh. 2022. POLE: Polarized Embedding for Signed Networks. In *WSDM*.
- [32] Boris Ivanovic and Marco Pavone. 2019. The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs. In *ICCV*.
- [33] Mohsen Jamali and Martin Ester. 2009. Trustwalker: a random walk model for combining trust-based and item-based recommendation. In *SIGKDD*.
- [34] Glen Jeh and Jennifer Widom. 2002. Simrank: a measure of structural-context similarity. In *SIGKDD*.
- [35] Brian Karrer and Mark EJ Newman. 2011. Stochastic blockmodels and community structure in networks. *PRE* 83, 1 (2011), 016107.
- [36] George Karypis and Vipin Kumar. 1998. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing* 20, 1 (1998), 359–392.
- [37] Leo Katz. 1953. A new status index derived from sociometric analysis. *Psychometrika* 18, 1 (1953), 39–43.
- [38] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.
- [39] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308* (2016).
- [40] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.
- [41] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *ICLR*.
- [42] Cheng Li, Jiaqi Ma, Xiaoxiao Guo, and Qiaozhu Mei. 2017. Deepcas: An end-to-end predictor of information cascades. In *WebConf*.
- [43] Juanhui Li, Harry Shomer, Haitao Mao, Shenglai Zeng, Yao Ma, Neil Shah, Jiliang Tang, and Dawei Yin. 2024. Evaluating graph neural networks for link prediction: Current pitfalls and new benchmarking. *Advances in Neural Information Processing Systems* 36 (2024).
- [44] Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. 2018. Learning deep generative models of graphs. In *ICML*.
- [45] David Liben-Nowell and Jon Kleinberg. 2007. The link-prediction problem for social networks. *Journal of the American society for information science and technology* 58, 7 (2007), 1019–1031.
- [46] Zheyi Liu, Darong Lai, Chuanyou Li, and Meng Wang. 2020. Feature Fusion Based Subgraph Classification for Link Prediction. In *CIKM*.
- [47] Linyuan Lü and Tao Zhou. 2011. Link prediction in complex networks: A survey. *Physica A: statistical mechanics and its applications* 390, 6 (2011), 1150–1170.
- [48] Haitao Mao, Juanhui Li, Harry Shomer, Bingheng Li, Wenqi Fan, Yao Ma, Tong Zhao, Neil Shah, and Jiliang Tang. 2023. Revisiting link prediction: A data perspective. *arXiv preprint arXiv:2310.00793* (2023).
- [49] Travis Martin, Brian Ball, and Mark EJ Newman. 2016. Structural inference for uncertain networks. *Physical Review E* 93, 1 (2016), 012306.
- [50] Victor Martinez, Fernando Berzal, and Juan-Carlos Cubero. 2016. A survey of link prediction in complex networks. *ACM computing surveys (CSUR)* 49, 4 (2016), 1–33.
- [51] Brian McFee and Gert Lanckriet. 2010. Metric learning to rank. In *ICML*.
- [52] Federico Monti, Michael Bronstein, and Xavier Bresson. 2017. Geometric matrix completion with recurrent multi-graph neural networks. In *NeurIPS*.
- [53] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. 2019. Weisfeiler and leman go neural: Higher-order graph neural networks. In *AAAI*.
- [54] Mark Newman. 2018. *Networks*. Oxford university press.
- [55] Mark EJ Newman. 2001. Clustering and preferential attachment in growing networks. *Physical review E* 64, 2 (2001), 025102.
- [56] Mark EJ Newman. 2006. Modularity and community structure in networks. *PNAS* 103, 23 (2006), 8577–8582.
- [57] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. 2016. Asymmetric transitivity preserving graph embedding. In *SIGKDD*.
- [58] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank citation ranking: Bringing order to the web*. Technical Report. Stanford InfoLab.
- [59] Liming Pan, Cheng Shi, and Ivan Dokmanić. 2022. Neural Link Prediction with Walk Pooling. In *ICLR*.
- [60] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*.
- [61] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *SIGKDD*.
- [62] Yanjun Qi, Ziv Bar-Joseph, and Judith Klein-Seetharaman. 2006. Evaluation of different biological data and computational classification methods for use in protein interaction prediction. *Proteins: Structure, Function, and Bioinformatics* 63, 3 (2006), 490–500.
- [63] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. 2018. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *WSDM*.
- [64] Jiezhong Qiu, Jian Tang, Hao Ma, Yuxiao Dong, Kuansan Wang, and Jie Tang. 2018. Deepinf: Social influence prediction with deep learning. In *SIGKDD*.
- [65] Jerome Revaud, Jon Almazán, Rafael S Rezende, and Cesar Roberto de Souza. 2019. Learning with average precision: Training image retrieval with a listwise loss. In *ICCV*.
- [66] Sunil Kumar Sahu, Fenia Christopoulou, Makoto Miwa, and Sophia Ananiadou. 2019. Inter-sentence Relation Extraction with Document-level Graph Convolutional Neural Network. In *ACL*.
- [67] Takaya Saito and Marc Rehmsmeier. 2015. The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PLoS one* 10, 3 (2015), e0118432.
- [68] Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin Riedmiller, Raia Hadsell, and Peter Battaglia. 2018. Graph networks as learnable physics engines for inference and control. In *ICML*.
- [69] Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. 2008. *Introduction to information retrieval*. Vol. 39. Cambridge University Press Cambridge.
- [70] Abhay Singh, Qian Huang, Sijia Linda Huang, Omkar Bhalerao, Horace He, Ser-Nam Lim, and Austin R Benson. 2021. Edge proposal sets for link prediction. *arXiv preprint arXiv:2106.15810* (2021).
- [71] Kihyuk Sohn. 2016. Improved deep metric learning with multi-class n-pair loss objective. In *NeurIPS*.
- [72] Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen. 2018. Open Domain Question Answering Using Early Fusion of Knowledge Bases and Text. In *EMNLP*.
- [73] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2018. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In *ICLR*.

- [74] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. Arnetminer: extraction and mining of academic social networks. In *SIGKDD*.
- [75] Yuchun Tang, Yan-Qing Zhang, Nitesh V Chawla, and Sven Krasser. 2008. SVMs modeling for highly imbalanced classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 39, 1 (2008), 281–288.
- [76] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.
- [77] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. Kgat: Knowledge graph attention network for recommendation. In *SIGKDD*.
- [78] Xinshao Wang, Yang Hua, Elyor Kodirov, Guosheng Hu, Romain Garnier, and Neil M Robertson. 2019. Ranked list loss for deep metric learning. In *ICCV*.
- [79] Xiyuan Wang, Haotong Yang, and Muhan Zhang. 2023. Neural Common Neighbor with Completion for Link Prediction. *arXiv preprint arXiv:2302.00890* (2023).
- [80] Bryan Wilder, Eric Ewing, Bistra Dilkina, and Milind Tambe. 2019. End to end learning and optimization on graphs. *NeurIPS* (2019).
- [81] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *ICML*.
- [82] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. 2008. Listwise approach to learning to rank: theory and algorithm. In *ICML*.
- [83] Zuoyu Yan, Tengfei Ma, Liangcai Gao, Zhi Tang, and Chao Chen. 2021. Link prediction with persistent homology: An interactive view. In *ICML*.
- [84] Bishan Yang, Scott Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *ICLR*.
- [85] Yonghui Yang, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. 2021. Enhanced graph learning for collaborative filtering via mutual information maximization. In *SIGIR*.
- [86] Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In *AAAI*.
- [87] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. 2018. Hierarchical graph representation learning with differentiable pooling. In *NeurIPS*.
- [88] Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. 2018. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *ICML*.
- [89] Seongjun Yun, Seoyoon Kim, Junhyun Lee, Jaewoo Kang, and Hyunwoo J Kim. 2021. Neo-GNNs: Neighborhood Overlap-aware Graph Neural Networks for Link Prediction. In *NeurIPS*.
- [90] Muhan Zhang and Yixin Chen. 2018. Link prediction based on graph neural networks. In *NeurIPS*.
- [91] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. 2018. An end-to-end deep learning architecture for graph classification. In *AAAI*.
- [92] Yiding Zhang, Xiao Wang, Chuan Shi, Nian Liu, and Guojie Song. 2021. Lorentzian graph convolutional networks. In *WebConf*.
- [93] Ziwei Zhang, Peng Cui, Xiao Wang, Jian Pei, Xuanrong Yao, and Wenwu Zhu. 2018. Arbitrary-order proximity preserved network embedding. In *SIGKDD*.
- [94] Tong Zhao, Gang Liu, Stephan Günnemann, and Meng Jiang. 2022. Graph data augmentation for graph machine learning: A survey. *arXiv preprint arXiv:2202.08871* (2022).
- [95] Tong Zhao, Yozen Liu, Leonardo Neves, Oliver Woodford, Meng Jiang, and Neil Shah. 2021. Data augmentation for graph neural networks. In *AAAI*.
- [96] Cheng Zheng, Bo Zong, Wei Cheng, Dongjin Song, Jingchao Ni, Wenchao Yu, Haifeng Chen, and Wei Wang. 2020. Robust graph representation learning via neural sparsification. In *ICML*.
- [97] Shijie Zhou, Zhimeng Guo, Charu Aggarwal, Xiang Zhang, and Suhang Wang. 2022. Link prediction on heterophilic graphs via disentangled representation learning. *arXiv preprint arXiv:2208.01820* (2022).
- [98] Tao Zhou, Linyuan Lü, and Yi-Cheng Zhang. 2009. Predicting missing links via local information. *The European Physical Journal B* 71, 4 (2009), 623–630.
- [99] Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian Tang. 2021. Neural bellman-ford networks: A general graph neural network framework for link prediction. In *NeurIPS*.

A Analysis of Link Prediction Evaluation Metrics with Different Test Settings

Example: Consider a graph with 10K nodes, 100K edges, and 99.9M disconnected (or negative) pairs. A (bad) model that ranks 1M false positives higher than the true edges achieves 0.99 AUC and 0.95 AP under *biased testing* with equal negative samples.

Figures 8a and 8b show the receiver operating characteristic (ROC) and precision-recall (PR) curves for the model under *biased testing* with equal number of negative samples. Due to the down-sampling, only 100k (out of 99.9M) negative pairs are included in

the test set, among which only $100k/99.9M \times 1M \approx 1k$ pairs are ranked higher than the positive edges. In the ROC curve, this means that once the false positive rate reaches $1k/100k = 0.01$, the true positive rate would reach 1.0, leading to an AUC score of 0.99. Similarly, in the PR curve, when the recall reaches 1.0, the precision is $100k/(1k + 100k) \approx 0.99$, leading to an overall AP score of ~ 0.95 .

By comparison, as shown in Figure 8c, when the recall reaches 1.0, the precision under *unbiased testing* is only $100k/(1M + 100k) \approx 0.09$, leading to an AP score of ~ 0.05 . This demonstrates that evaluation metrics based on *biased testing* provide an overly optimistic measurement of link prediction model performance compared to the more realistic *unbiased testing* setting.

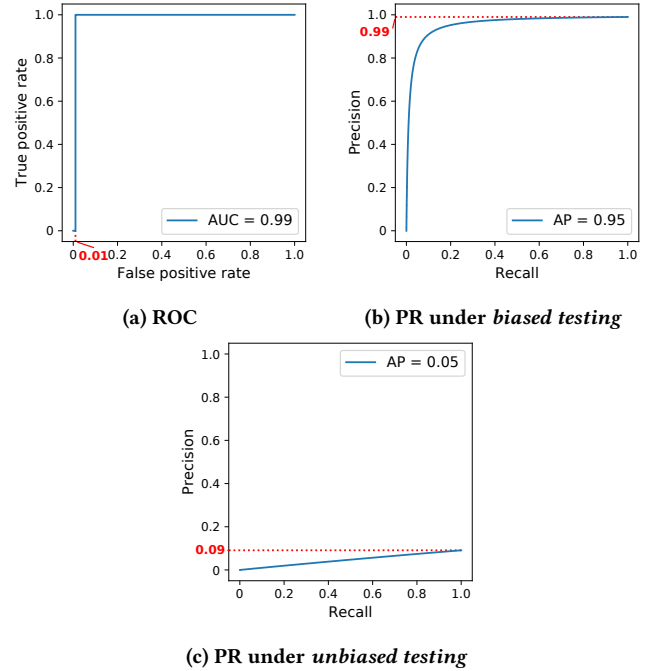


Figure 8: Receiver operating characteristic and precision-recall curves for the bad link prediction model that ranks 1M false positives higher than the 100k true edges. The model achieves 0.99 in AUC and 0.95 AP under *biased testing*, while the more informative performance evaluation metric, Average Precision (AP) under *unbiased testing*, is only 0.05.

B Proof of Theorem 2.1

There are only three classifiers that we need to consider in this setting, assuming that the classifier can recover the block structure:

- (1) It predicts every disconnected pair as a link;
- (2) It predicts every disconnected pair as a non-link;
- (3) It predicts within-block pairs as links and across-block pairs as non-links.

The classifier 1 cannot be optimal for sparse graphs—i.e., density lower than .5—and thus we will focus on classifiers 2 and 3. We will compute the expected number of True Positives (TP), False

Positives (FP), False Negatives (FN), and True Negatives (TN) per node for each of them:

Classifier 2:

$$\begin{aligned} TP &= 0 \\ FN &= 0 \\ FP &= (n-1)p + (nk-n)q \\ TN &= (n-1)(1-p) + (nk-n)(1-q) \end{aligned}$$

Classifier 3:

$$\begin{aligned} TP &= (n-1)p \\ FN &= (nk-n)q \\ FP &= (n-1)p \\ TN &= (nk-n)(1-q) \end{aligned}$$

The accuracy of the classifiers is computed as $(TP + TN)/(TP + TN + FP + FN)$. It follows that the difference between accuracy of the classifier 2 and 3 is as follows:

$$\frac{(n-1)(1-p) + (nk-n)(1-q)}{nk-1} - \frac{(n-1)p + (nk-n)(1-q)}{nk-1}$$

And thus, classifier 2 outperforms classifier 3 for $p < 0.5$.

C Proof of Lemma 2

We will consider the same classifiers 2 and 3 from the proof of Theorem 2.1. Moreover, we will assume that the number of sampled negative pairs is the same as the number of positive pairs (i.e., balanced sampling).

By definition, the accuracy of classifier 2 is 0.5, as all predictions for negative pairs will be correct and all those for positive pairs will be incorrect. Thus, we only have to show that there exists an SBM instance for which classifier 3 achieves better accuracy than 2.

The accuracy of classifier 3 is computed as $a_1 + a_2/2$, where:

$$\begin{aligned} a_1 &= \frac{(n-1)p}{(n-1)p + (nk-n)q} \\ a_2 &= \frac{(nk-n)(1-q)}{(nk-n)(1-q) + (n-1)(1-p)} \end{aligned}$$

It follows that, as $q \rightarrow 0$, classifier 3 can achieve an accuracy higher than 0.5.

D Proof of Lemma 3

Let us initially consider Autocovariance with $t = 1$ computed in the Stochastic Block Model described in Lemma 3. We will adopt the entry-wise notation of the original Autocovariance definition presented in Section 3.2, using lower-case letters to represent individual entries in matrices and vectors, and for the sake of consistency with the Modularity definition, we adopt $\text{vol}(G) = 2m$. We first obtain the shortened form of Autocovariance for $t = 1$:

$$R_{ij} = \frac{1}{2m}(a_{ij} - \frac{d_i d_j}{2m}). \quad (10)$$

We can obtain the expected expression value for the case where (i, j) is an intra-cluster pair ($\mathbb{E}[R_{intra}]$):

$$\mathbb{E}[R_{intra}] = \frac{1}{2m}((1 - \frac{d_i d_j}{2m})p + (0 - \frac{d_i d_j}{2m})(1-p)) \quad (11)$$

$$= \frac{1}{2m}(p - \frac{d_i d_j}{2m}). \quad (12)$$

Likewise, we follow the same procedure for the case where (i, j) is an inter-cluster pair ($\mathbb{E}[R_{inter}]$):

$$\mathbb{E}[R_{inter}] = \frac{1}{2m}((1 - \frac{d_i d_j}{2m})(1-p) + (0 - \frac{d_i d_j}{2m})p) \quad (13)$$

$$= \frac{1}{2m}(1-p - \frac{d_i d_j}{2m}) \quad (14)$$

$$= \frac{1}{2m}(q - \frac{d_i d_j}{2m}). \quad (15)$$

Due to the reversible property of Markov chains, this holds for larger values of t .

Since $p > q \implies \mathbb{E}[R_{intra}] > \mathbb{E}[R_{inter}]$.

E Proof of Lemma 4

From Appendix D, we have $\mathbb{E}[R_{intra}] = \frac{1}{2m}(p - \frac{d_i d_j}{2m})$ is solely dependent on the value of p , since all the other terms are constants. We will denominate V_{ik} and E_{ik}^+ the number of nodes and positive pairs in the i -th partition of our graph partitioned in k partitions.

Considering the estimate $p = |E_{ik}^+|/|V_{ik}|^2$, for simplicity, the number of positive pairs we can lose by increasing k to $k+1$ is *at most* $|E_{ik+1}^+| \geq |E_{ik}^+| - (|V_{ik}|^2 - |V_{ik+1}|^2)$, if we consider the extreme scenario in which every pair lost was positive. With this estimate, we can compare with the actual p estimate:

$$\frac{|E_{ik+1}^+|}{|V_{ik+1}|^2} \geq \frac{|E_{ik}^+| - (|V_{ik}|^2 - |V_{ik+1}|^2)}{|V_{ik+1}|^2} \quad (16)$$

$$|V_{ik}|^2 - |V_{ik+1}|^2 \geq |E_{ik}^+| - |E_{ik+1}^+| \quad (17)$$

It follows that, since the number of pairs drops faster than the number of positive edges for a given partition, $\mathbb{E}[R_{intra}]$ increases when k increases.

F Detailed experiment settings

Positive masking. To ensure generalizability under *unbiased training*, we employ a trick similar to *negative injection* [90]. Training positive edges are divided into batches, and for each batch E_b , only the residual edges $E - E_b$ are used as structural input to the model. This simulates testing, where edge predictions are made without leveraging their own connectivity. We term this *positive masking*.

Implementation details. Self-loops are added to isolated nodes in the enhanced adjacency matrix to ensure valid transition probabilities for computing Autocovariance. Gelato similarity scores are standardized as in [30] before loss computation. We train the model using gradient descent with pytorch [60] and skip parameter updates for batches with invalid gradients (e.g., with cross-entropy loss). Models are selected based on *prec@100%* on the (unbiased) validation set. Maximum epochs are set to 100 for CORA/CITESEER and 250 for OGBL-DDI/OGBL-COLLAB. For *partitioned testing*, we use METIS [36] for its scalability and balanced partitions.