

CORE: Data Augmentation for Link Prediction via Information Bottleneck

Kaiwen Dong
University of Notre Dame
USA
kdong2@nd.edu

Zhichun Guo
University of Notre Dame
USA
zguo5@nd.edu

Nitesh V. Chawla
University of Notre Dame
USA
nchawla@nd.edu

ABSTRACT

Link prediction (LP) is a fundamental task in graph representation learning, with numerous applications in diverse domains. However, the generalizability of LP models is often compromised due to the presence of noisy or spurious information in graphs and the inherent incompleteness of graph data. To address these challenges, we draw inspiration from the Information Bottleneck principle and propose a novel data augmentation method, CComplete and REduce (CORE) to learn compact and predictive augmentations for LP models. In particular, CORE aims to recover missing edges in graphs while simultaneously removing noise from the graph structures, thereby enhancing the model's robustness and performance. Extensive experiments on multiple benchmark datasets demonstrate the applicability and superiority of CORE over state-of-the-art methods, showcasing its potential as a leading approach for robust LP in graph representation learning.

CCS CONCEPTS

- Information systems → Data mining; • Computing methodologies → Regularization.

KEYWORDS

link prediction, data augmentation, information bottleneck, graph neural networks

ACM Reference Format:

Kaiwen Dong, Zhichun Guo, and Nitesh V. Chawla. 2024. CORE: Data Augmentation for Link Prediction via Information Bottleneck. In . ACM, New York, NY, USA, 17 pages. <https://doi.org/10.1145/nnnnnnnn.nnnnnnnn>

1 INTRODUCTION

Graph-structured data is ubiquitous in various domains, including social networks [31], recommendation systems [27], and protein-protein interactions [47]. Link prediction (LP), the task of predicting missing or future edges in a graph, is a fundamental problem in graphs. Over the years, a plethora of link prediction algorithms have been proposed, ranging from heuristics-based link predictors [1, 23, 31, 71] to more sophisticated graph neural network (GNN) based methods [25, 37, 63].

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 ACM 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.

© 2024 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06...\$15.00
<https://doi.org/10.1145/nnnnnnnn.nnnnnnnn>

One major challenge in LP is the quality, reliability, and veracity of graph data. In many real-world scenarios, data collection can be difficult due to factors such as incomplete information, errors in data labeling, and noise introduced by measurement devices or human mistakes [9, 72]. As a consequence, the graph constructed based on the collected data may contain missing or erroneous edges. This can subsequently impact the performance of LP models. Moreover, the excessive dependence on noisy graphs can impede the model's capability in distinguishing the real and spurious edges, which harms the generalizability of models. Therefore, the question of preserving the robust learning capacity and generalizability of LP models on noisy graphs remains unresolved.

To mitigate the degradation of model performance on noisy data with inferior data quality, data augmentation (DA) has emerged as a powerful technique by artificially expanding the training dataset with transformed versions of the original data instances, primarily in the field of computer vision [29, 41]. However, in the context of LP, few works have been proposed to overcome the limitation of models on noisy graphs [67]. For example, CFLP [68] employs causal inference by complementing counterfactual links into the observed graph. Edge Proposal [42] seeks to inject highly potential edges into the graph as a signal-boosting preprocessing step. Nevertheless, these works fail to consider the inherent noise or that brought by the augmentation process, holding an implicit assumption that the observed graph truly reflects the underlying relationships

In this paper, we investigate how to augment the graph data for link prediction to accomplish two primary goals: ***eliminating noise inherent in the data and recovering missing information in graphs***. To augment with robust, diverse, and noise-free data, we employ the Information Bottleneck (IB) principle [48, 49]. IB offers a framework for constraining the flow of information from input to output, enabling the acquisition of a maximally compressed representation while retaining its predictive relevance to the task at hand [2]. Learning such an effective representation is particularly appealing because it gives us a flexible DA pipeline where we can seamlessly integrate other DA techniques without concern for the introduction of extraneous noise they might bring.

Present work. We introduce CComplete and REduce (CORE), a novel data augmentation framework tailored for the link prediction task. CORE comprises of two distinct stages: the Complete stage and the Reduce stage. The Complete stage addresses the incompleteness of the graph by incorporating highly probable edges, resulting in a more comprehensive graph representation. One can plug in any link predictors that may be advantageous in recovering the graph's structural information, despite the possibility of introducing noisy edges. The Reduce stage, which is the crux of the proposed method, operates on the augmented graph generated by

the Complete stage. It aims to shrink the edge set while preserving those critical to the link prediction task. In doing so, the Reduce stage effectively mitigates any misleading information either inherently or introduced during the Complete stage. By adhering to the IB principle, the Reduce stage yields a minimal yet sufficient graph structure that promotes more generalizable and robust link prediction performance.

However, unlike DA in images where transformations can be applied independently, modifications to a single node or edge in a graph inevitably impact the surrounding neighborhood. This arises from the interdependence of data instances within a graph. Under these conditions, applying a universal DA to different instances in a graph may be suboptimal, as a specific augmentation could benefit one link prediction while negatively affecting another. For example, in Figure 1, the inference of different links may favor adding different edges into their neighborhood. To address this dependency issue within a graph, we recast the link prediction task as a subgraph link prediction [12, 63]. In this context, we can apply different DAs to neighboring links without concerns about potential conflicts between their preferred augmentations. This approach allows for more targeted and effective augmentation, ultimately enhancing the performance of our CORE framework in link prediction tasks.

2 PRELIMINARY

In this section, we introduce the notations and concepts utilized throughout the paper.

Graph and link prediction. Let $G = (V, E, \mathbf{X})$ represent an undirected graph, where V is the set of nodes with size n , indexed as $\{i\}_{i=1}^n$. N_v is the neighborhood of the node v . $E \subseteq V \times V$ denotes the observed set of edges, and $\mathbf{X}_i \in \mathcal{X}$ represents the feature of node i . The unobserved set of edges, denoted by $E_c \subseteq V \times V \setminus E$, comprises either missing edges or those expected to form in the future within the original graph G . Thus, for those links $(i, j) \in E \cup E_c$, we can assign label $Y = 1$ and regard them as positive samples, while the rest $\{(i, j) \subseteq V \times V | (i, j) \notin E \cup E_c\}$ we assign label $Y = 0$ as negatives. Based on the given graph G , the goal of the link prediction task is to compute the nodes similarity scores to identify the unobserved set of edges E_c [32]. Numerous heuristic models are proposed for link prediction task over time, including Common Neighbor (CN) [31], Adamic-Adar index (AA) [1], Resource Allocation (RA) [71], and Katz index [23]. While these traditional approaches effectively utilize the topological structure of the graph, GNN-based models [25, 63] exhibit a superior ability to exploit both the structure and node attributes associated with the graph.

Subgraph link prediction. Even though some link predictors, such as the Katz index and PageRank [6], require the entire graph to calculate similarity scores for a target link, many others only rely on a local neighborhood surrounding the target link for computation. For instance, the Common Neighbor predictor necessitates only a 1-hop neighborhood of the target link, and generally, an l -layer GNN requires the l -hop neighborhood of the target link. Moreover, Zhang and Chen [63] has demonstrated that local information can be sufficient for link prediction tasks. As a result, the link prediction task for a specific target link can be reformulated

as a graph classification problem based on the local neighborhood of the target link, aiming to determine whether the link exists or not [12]. Formally, given a subgraph $G_{(i,j)}^l$ induced by the nodes l -hop reachable from node pair (i, j) , a subgraph link prediction is a task of predicting the label $Y \in \{0, 1\}$ for the subgraph, where $Y = 1$ indicates that the target link exists, and vice versa.

Data augmentation. Data augmentation is the process of expanding the input data by either slightly perturbing existing data instances or creating plausible variations of the original data. This technique has been proven effective in mitigating overfitting issues during training, particularly in the fields of computer vision [10] and natural language processing [14]. In the realm of graph representation learning, several DA methods have been proposed [67] to address challenges such as oversmoothing [39], generalization [9], and over-squashing [50]. However, most graph data augmentation techniques have primarily focused on node and graph classification tasks, with relatively limited exploration in the context of LP [68].

3 PROPOSED FRAMEWORK: CORE

In this section, we present our proposed two-stage data augmentation framework for LP, referred to as CORE. We begin by introducing the Complete stage, which aims to recover missing edges in the original graph. Following this, we discuss the Reduce stage, the most critical component of our proposed method, designed to eliminate noisy and spurious edges in the graph. Finally, we outline a practical implementation that leverages the Graph Information Bottleneck (GIB) [58] for pruning inflated edges. The overview of the framework is shown in Figure 1.

3.1 Complete stage: inflating missing connections

The data collection process is inherently susceptible to errors, which can result in incomplete or even erroneous structural information in the original graph. Furthermore, the nature of the link prediction task involves identifying missing or potentially newly forming edges, implicitly assuming that graph data is incomplete. Therefore, mitigating the incompleteness of graph structures can be advantageous. In Theorem 1, we will also see how inflating missing edges can help identify the most crucial component that determines whether a link should exist.

Implementation. We begin with a simple and straightforward method introduced by Singh et al. [42] to inflate the original graph with additional edges. Although more sophisticated graph completion methods [54] can also be plugged into the Complete stage, we find that employing a straightforward, low-computational-cost algorithm is sufficient for augmenting the graph structures effectively.

Due to the sparsity of most real-world graphs, the number of potentially missing edges is proportional to the quadratic of the number of nodes $O(n^2)$. Scoring all non-connected node pairs can be computationally prohibitive. To reduce the size of the candidate node pairs, we only consider those non-connected pairs that have at least one common neighbor for potential addition to the graph. Subsequently, we can use any link prediction method to score these candidate node pairs. For large-scale datasets, like OGB-Collab [20],

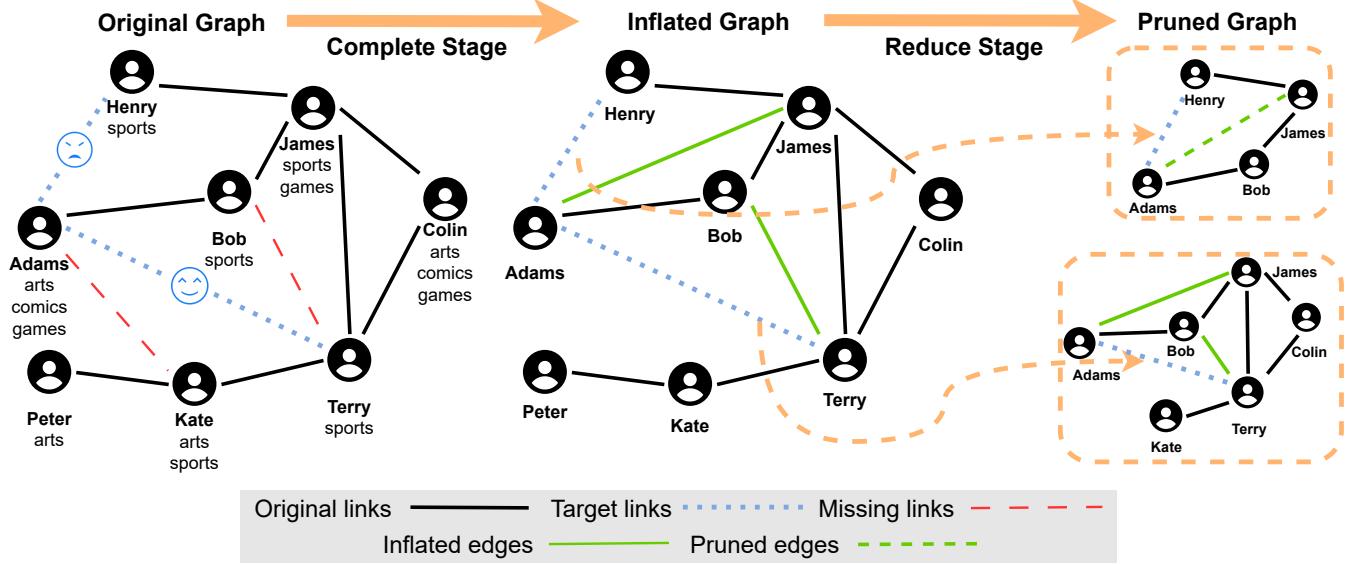


Figure 1: Overview of our CORE framework. It consists of two stages: (1) the Complete stage, which aims to recover missing edges by incorporating highly probable edges into the original graph, and (2) the Reduce stage, which is the core component of our method, designed to prune noisy edges from the graph in order to prevent overfitting on the intrinsic noise and that introduced by the Complete stage. Recognizing that predicting different links may require distinct augmentations, we extract the surrounding subgraph of each link and apply independent augmentations accordingly. In the social network example illustrated, assuming that Adams and Terry will become friends while Adams and Henry will not, tailored augmentations can facilitate more accurate link prediction by the model.

we can rely on the faster computation of non-parametric heuristic methods. For graphs of moderate size, one may choose any heuristic methods or GNN-based methods like GCN [26] and SAGE [17]. Note that only scoring node pairs with common neighbors is a pragmatic choice. Most real-world graphs, governed by popular models such as the assortative SBM [18] or the Watts–Strogatz model [56], exhibit higher connection likelihood for node pairs with shared neighbors. Therefore, this design balances computational efficiency with empirical effectiveness, a trade-off we believe to be minimal but crucial for practical purposes. An empirical investigation can be found in Appendix C.6.

After scoring all the candidate node pairs, we sort them based on their similarity scores and select the top k node pairs E_{ext} to add to the original graph, where k is a hyperparameter. Thus, the graph G becomes $G^+ = (V, EU E_{ext}, X)$. It is important to note that, although we add these predicted links to the graph, we mark them as *inflated edges* to differentiate them from the original graph. These inflated edges will not be used as training signals for later stages but will only serve as a complement to the graph’s topological structure. This distinction is crucial, as we can tolerate the noisy edges in the input space but do not want to introduce any noise to the labels of LP in our DA process.

3.2 Reduce stage: pruning noisy edges

The Reduce stage is the central aspect of our CORE data augmentation framework. Inspired by GSAT [35], we leverage GIB [58] to parameterize a reducer, which constrains the graph structure

to a minimal yet sufficient graph component for link prediction. The resulting graph component is expected to achieve three goals: (1) remove task-irrelevant information from the data (the regularization in Equation 6); (2) prune the graph structures so that only the most predictive graph components remain for inference (the log-likelihood in Equation 6); and (3) provide diversified augmentation to the original data (the edge sampling step). We begin by introducing the necessity of decoupling the DA for each link. Then we discuss the GIB objective and its tractable variational bound. Finally, we present the implementation of our data augmentation in the Reduce stage.

Interdependence of graph data. In the link prediction task, the data instances we are interested in are the links in the graph. However, unlike images, links in a graph are correlated; the existence and properties of each link are dependent on one another. Consequently, when applying DA to a specific link, it will inevitably affect the environment of other links, especially those in close proximity. This can yield suboptimal results, as links will compete with each other to obtain the best augmentation for their own sake. Furthermore, it becomes computationally infeasible to apply the IB principle when the i.i.d assumption does not hold [58].

To address these issues, we reformulate LP as a subgraph link prediction. Subgraph link prediction allows for decoupling the overlapping environments of each link, making it possible to have a different DA for each link. Specifically, for each node pair (i, j) , we extract its l -hop enclosing subgraph $G_{(i,j)}^{+,l}$ from the entire graph

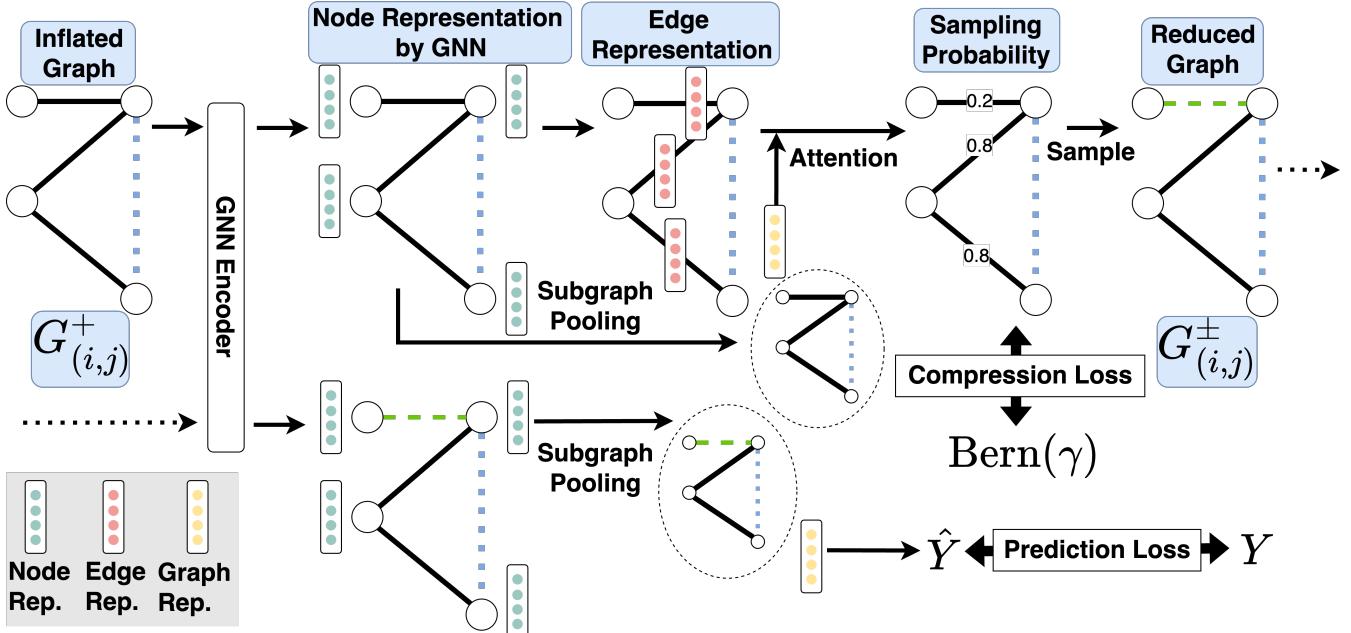


Figure 2: The Reduce stage commences with the inflated subgraph $G^+_{(i,j)}$ surrounding the target link (i, j) . We first apply a GNN to encode node representations, followed by edge representation derived from the node encodings. To compute sampling probability scores for each edge, we utilize an attention mechanism that combines the edge representation with the subgraph pooling. Since the subgraph pooling encapsulates information from the entire subgraph and is employed for target link prediction, the generated probability scores reflect not only the edge’s inherent property but also its relationship to the target link (i, j) . Subsequently, we sample each edge using a Bernoulli distribution based on its probability to obtain the pruned graph. Finally, the pruned graph $G^\pm_{(i,j)}$ is fed back into the model as augmented input for enhanced graph structures.

G^+ . To simplify notation when there is no ambiguity, we may omit the number of hops and represent the subgraph as $G^+_{(i,j)}$. It is worth noting that prior works also adopt a similar strategy to handle the non i.i.d nature of graph data [55, 69] when perturbing the data. In Section 4.3, we empirically examine the optimal DAs tailored to different target links. Notably, the DA derived from a single edge may vary depending on the target link under consideration.

GIB. In general, IB aims to learn a concise representation Z from the input X that is also expressive for the output y , measured by the mutual information between the latent representation and input/output [2, 49]. Thus, the objective is:

$$\max_Z I(Z, Y) \text{ s.t. } I(X, Z) \leq I_c. \quad (1)$$

where $I(\cdot, \cdot)$ denotes the mutual information and I_c is the information constraint. In the context of LP, we can regard the enclosing subgraph $G^+_{(i,j)}$ as input X , including both the node attributes and graph structure. Y is the link’s existence at (i, j) , and Z is the latent representation.

While the original GIB [58] constrains the information flow from both node attributes of a graph and graph structures, we propose to only constrain the structural information in our data augmentation for the link prediction task. Compared to node attributes in a graph, graph structures are overwhelmingly more critical for the

link prediction task [32, 37]. Moreover, many graphs without node attributes still exhibit the need for link prediction. Thus, we define our objective as:

$$\max_{G^\pm_{(i,j)} \in \mathbb{G}_{\text{sub}}(G^+_{(i,j)})} I(G^\pm_{(i,j)}, Y) \text{ s.t. } I(G^\pm_{(i,j)}, G^+_{(i,j)}) \leq I_c. \quad (2)$$

where $G^\pm_{(i,j)} \in \mathbb{G}_{\text{sub}}(G^+_{(i,j)})$ is a subgraph pruned from the inflated graph $G^+_{(i,j)}$. In other words, we aim to find the subgraph of the inflated graph that is simultaneously the most predictive and concise for the link prediction task. We assume that this graph reduction process can prune the noisy edges introduced by the previous Complete stage while retaining the beneficial added information. Our method shares a similar spirit with GSAT [35] and IB-subgraph [62], as we explore finding a subgraph structure that is most essential for the task.

Next, by introducing a Lagrange multiplier β , we obtain the unconstrained version of the objective:

$$\min_{G^\pm_{(i,j)} \in \mathbb{G}_{\text{sub}}(G^+_{(i,j)})} -I(G^\pm_{(i,j)}, Y) + \beta I(G^\pm_{(i,j)}, G^+_{(i,j)}). \quad (3)$$

where β is the hyperparameter to balance the tradeoff between predictive power and compression.

The computation of the mutual information term $I(\cdot, \cdot)$ is, in general, computationally intractable. To address this issue, we follow the works of Alemi et al. [2], Miao et al. [35], Wu et al. [58]

to derive a tractable variational upper bound for Equation 3. The detailed derivation is provided in Appendix D. To approximate the first term $I(G_{(i,j)}^\pm; Y)$, we derive a variational lower bound. The lower bound can be formulated as:

$$I(G_{(i,j)}^\pm; Y) \geq \mathbb{E}[\log q_\theta(Y|G_{(i,j)}^\pm)]. \quad (4)$$

Essentially, q_θ is the predictor of our model, which can be a parameterized GNN.

For the second term $I(G_{(i,j)}^\pm, G_{(i,j)}^+)$, we derive an upper bound by introducing a variational approximation $r(G_{(i,j)}^\pm)$ for the marginal distribution of $G_{(i,j)}^\pm$:

$$I(G_{(i,j)}^\pm, G_{(i,j)}^+) \leq \mathbb{E}[\text{KL}(p_\phi(G_{(i,j)}^\pm|G_{(i,j)}^+))||r(G_{(i,j)}^\pm))] \quad (5)$$

where p_ϕ is the reducer to prune noisy edges. Then we can put everything together and get the empirical loss to minimize:

$$\mathcal{L} \approx \frac{1}{|E|} \sum_{(i,j) \in E} [-\log q_\theta(Y|G_{(i,j)}^\pm)] \quad (6)$$

$$+ \beta \text{KL}(p_\phi(G_{(i,j)}^\pm|G_{(i,j)}^+))||r(G_{(i,j)}^\pm))). \quad (7)$$

Next, we discuss how to parameterize the predictor q_θ and the reducer p_ϕ in the Reduce stage, as well as the choice of marginal distribution $r(G_{(i,j)}^\pm)$.

3.3 Implementation of the Reduce stage.

The overall architecture of the Reduce stage is shown in Figure 2. It is important to note that both the predictor q_θ and the reducer p_ϕ utilize a GNN encoder to encode the graph representation for either prediction or graph pruning purposes. These two components can share a common GNN encoder with the same parameters, as the entire training of the Reduce stage is end-to-end. This shared encoder allows for more efficient learning and reduces the number of parameters required in the model.

Subgraph encoding. The Reduce stage begins with encoding the inflated graph $G_{(i,j)}^+$ using a GNN. We can choose any Message Passing Neural Network (MPNN) [16] as the instantiation of the GNN encoder. The MPNN can be described as follows:

$$\mathbf{m}_v^{(l)} = \text{AGG}(\{\mathbf{h}_u^{(l)}, \mathbf{h}_v^{(l)}, \forall u \in N_v\}), \quad (8)$$

$$\mathbf{h}_v^{(l+1)} = \text{UPDATE}(\{\mathbf{h}_v^{(l)}, \mathbf{m}_v^{(l)}\}). \quad (9)$$

where a neighborhood aggregation function $\text{AGG}(\cdot)$ and an updating function $\text{UPDATE}(\cdot)$ are adopted in the t -th layer of a T -layer GNN. Consequently, $\{\mathbf{h}_v^{(T)} | v \in G_{(i,j)}^+\}$ represents the node embeddings learned by the GNN encoder.

A typical link prediction method, such as GAE [25] or SEAL [63], can make a prediction by pooling the node representations, namely $\mathbf{h}_{G_{(i,j)}^+} = \text{POOL}(\{\mathbf{h}_v^{(T)} | v \in G_{(i,j)}^+\})$, where $\mathbf{h}_{G_{(i,j)}^+}$ is the final representation for the node pair (i, j) . In our data augmentation approach, however, we first need to prune the noisy edges in order to obtain more concise graph structures.

Reduce by edge sampling. After encoding the node representations, we proceed to prune the noisy edges in the inflated graph. We first represent each edge (u, v) in the inflated graph $G_{(i,j)}^+$ by concatenating the node representations of the two end nodes and

a trainable embedding indicating whether this edge comes from the original graph $G_{(i,j)}$ or the Complete Stage. Specifically, we obtain $\mathbf{h}_{(u,v)} = [\mathbf{h}_u; \mathbf{h}_v; \tilde{\mathbf{h}}_{(u,v)}]$, where $[\cdot; \cdot]$ is the concatenation operation. Appending $\tilde{\mathbf{h}}_{(u,v)}$ to the edge representation enables the model to be aware of whether the edges are originally in the graph or introduced by link predictors at the Complete stage.

In this setting, the edge representation $\mathbf{h}_{(u,v)}$ solely contains information about its structural role in the inflated graph. While structurally similar edges might influence distinct target node pairs differently, this representation does not convey information about how the edge (u, v) in the local subgraph $G_{(i,j)}^+$ affects the prediction of the target node pair (i, j) . To make the edge representation directly interact with the downstream link prediction task, we further apply an attention mechanism [51, 52] and attend it to the overall representation of the entire subgraph to define its importance for link prediction. We compute this as follows:

$$a_{(u,v)} = Q_\phi(\mathbf{h}_{G_{(i,j)}^+})^T K_\phi(\mathbf{h}_{(u,v)}) / \sqrt{F''}, \quad (10)$$

where Q_ϕ and K_ϕ are two MLPs and F'' is the output dimension of the MLPs.

Unlike GAT [52], which directly applies the attention scores as edge weights in each layer, we use these scores $a_{(u,v)}$ to sample the edges to diversify the views of the graph. For each edge (u, v) in $G_{(i,j)}^+$, we sample an edge mask from the Bernoulli distribution $\omega_{(u,v)} \sim \text{Bern}(\text{sigmoid}(a_{(u,v)}))$, which masks off unnecessary edges in the graph for LP. To ensure the gradient can flow through the stochastic node here, we utilize the Gumbel-Softmax trick [22, 34]. This procedure gives us a way to generate the reduced subgraph $G_{(i,j)}^\pm$ by the variational distribution $p_\phi(G_{(i,j)}^\pm|G_{(i,j)}^+)$.

To control the marginal distribution in Equation 5, we follow [35, 58] and apply a non-informative prior $r(\tilde{G}_{(i,j)})$. In other words, $\tilde{G}_{(i,j)}$ is obtained by sampling edge connectivity $\tilde{\omega}_{(u,v)} \sim \text{Bern}(\gamma)$ for every node pair (u, v) in $G_{(i,j)}^+$. γ is a hyperparameter. Regardless the graph structure of $G_{(i,j)}^\pm$, we connect (u, v) if $\tilde{\omega}_{(u,v)} = 1$ and disconnect the rest. This is essentially an Erdős-Rényi random graph [13]. The derivation of the KL loss term with respect to the marginal distribution is detailed in Appendix B.1.

Prediction based on pruned subgraph. Once we obtain the edge mask ω , we can encode the subgraph and make a link prediction. The edge mask can be regarded as the edge weight of the inflated graph. In this way, the edge weight plays the role of a message passing restrictor to prune the noisy edges in the inflated graph, which modifies the message passing part of Equation 8 as follows:

$$\mathbf{m}_v^{(l)} = \text{AGG}(\{\omega_{(u,v)} * \mathbf{h}_u^{(l)}, \mathbf{h}_v^{(l)}, \forall u \in N_v\}). \quad (11)$$

Using the representation learned from the reduced subgraph $G_{(i,j)}^\pm$, we can feed them into a pooling layer plus an MLP to estimate Y . This models the distribution $q_\theta(Y|G_{(i,j)}^\pm)$.

3.4 Theoretical analysis

In this section, we provide a theoretical foundation for the integration of the Complete and Reduce stages in our data augmentation approach for link prediction tasks.

THEOREM 1. Assume that: (1) The existence Y of a link (i, j) is solely determined by its local neighborhood $G_{(i,j)}^*$ in a way such that $p(Y) = f(G_{(i,j)}^*)$, where f is a deterministic invertible function; (2) The inflated graph contains sufficient structures for prediction $G_{(i,j)}^* \in \mathbb{G}_{\text{sub}}(G_{(i,j)}^+)$. Then $G_{(i,j)}^\pm = G_{(i,j)}^*$ minimizes the objective in Equation 3.

The first assumption in Theorem 1 is consistent with a widely accepted *local-dependence* assumption [58, 63] for graph-structured data. The second assumption highlights the importance of incorporating enough structural information into the graph in the Complete Stage prior to executing the reduce operation. Even though we assume that the link existence Y is causally determined by $G_{(i,j)}^*$, there still can be some other spurious correlations between Y and $G_{(i,j)}^+$. These correlations can be brought by the environments [3, 28, 57], and the shift of such correlations in the testing phase can cause performance degradation for LP models.

Theorem 1 implies that under mild assumptions, optimizing the objective in Equation 3 can help us uncover the most crucial component of the graph, which determines whether a link should exist. As a result, our approach enables the elimination of noisy and spurious edges, thereby enhancing the generalizability of link prediction models. The proof can be found in Appendix E.

4 EXPERIMENTS

In this section, we present experimental results for our proposed method. We first assess the performance of CORE in comparison to various baseline DA techniques for the link prediction task. Then, we illustrate that heuristic link predictors can also benefit from the augmented graph structure by CORE. Furthermore, we demonstrate its robustness against adversarial edge perturbations. Further details of the experiments can be found in Appendix B.

4.1 Experimental setup

Baseline methods. We select three heuristic link predictors for non-GNN models: CN [36], AA [1], and RA [71]. For GNN models that exploit node-level representation, we employ the two most widely used architectures: GCN [26] and SAGE [17]. For the link prediction utilizing edge-level representation, we choose SEAL [63], ELPH [7] and NCNC [54] as the baseline.

We select *Edge Proposal* [42] and *CFLP* [?], as two representative DA baselines with *GCN* as backbone. For *SEAL*, we evaluate two standard graph perturbation techniques [61], *Node Drop* [38] and *Edge Drop* [39]. Then, we present our results with *Complete Only*, *Reduce Only*, and the combined *CORE*. Details about these baseline models can be found in Appendix C.1.

Benchmark datasets. We select four attributed and four non-attributed graphs as the benchmark. The attributed graphs consist of three collaboration networks, **CS**, **Physics** [40] and **Collab** [58], as well as a co-purchased graph, **Computers** [40]. The non-attributed graphs include **USAir** [4], **Yeast** [53], **C.ele** [56], and **Router** [44]. The comprehensive descriptions and statistics of the benchmark datasets can be found in Appendix C.2.

Evaluation protocols. We follow the evaluation settings from previous work [?] and split the links as 10% for validation, 20% for

testing. For **Collab** [20], we use the official train-test split. The evaluation metric is Hits@50, which is widely accepted for evaluating link prediction tasks [20]. The results are reported for 10 different runs with varying model initializations.

4.2 Experimental results

Link prediction. Table 1 presents the link prediction performance of Hits@50 for all methods. Given the strong backbone model *SEAL*, we observe that our proposed data augmentation can further improve its performance on various datasets. In comparison to *SEAL* without any DA techniques, CORE consistently boosts the performance by 1% to 9% in terms of Hits@50. More specifically, both *Complete Only* and *Reduce Only* can increase the model capability by different margins. Moreover, by combining those two stages together, CORE can almost always achieve the best performance and significantly outperforms baselines. Our results also reveal that CORE yields greater performance improvements when the available data size is limited. This observation suggests that models may be prone to overfitting to noise in low-data regimes. However, CORE effectively mitigates this issue by learning an underlying (Bernoulli) distribution associated with the graph structures, and prevents the model from overfitting to idiosyncratic structural perturbations.

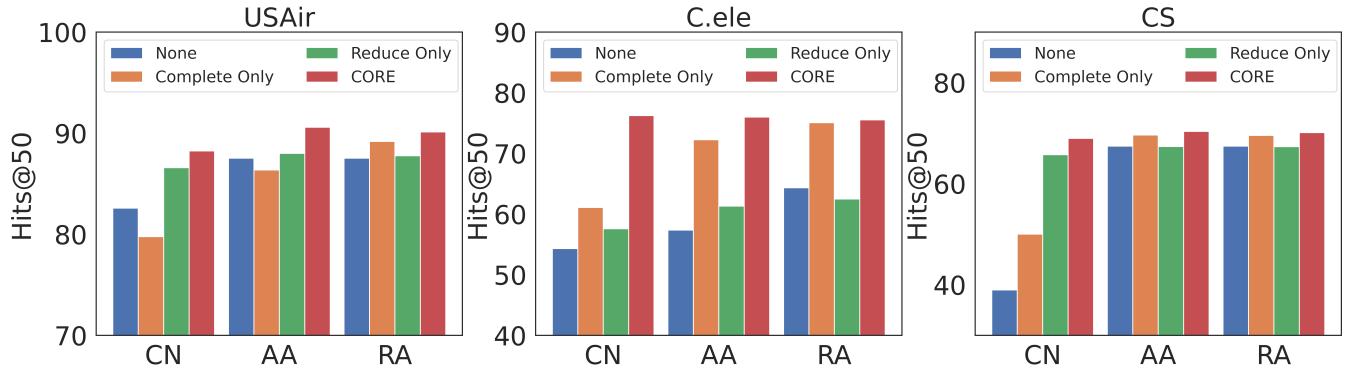
Learnable and transferrable. One potential concern with using the reducer of CORE, which is a neural network possessing the capability of universal approximation [19], is that the performance improvement might be attributed to the overparameterization [5] of the model instead of the quality of our augmented graph. To address this concern and validate the efficacy of CORE as a DA method, we decouple the reducer from the model and investigate its ability to extract a generalizable view of the graph. We feed the augmented graph generated by the reducer to three heuristic link predictors: *CN*, *AA*, and *RA*. The results of this experiment can be found in Figure 3. Our findings demonstrate that the graph refined by CORE consistently improves the performance of heuristic link predictors. This outcome validates CORE’s ability to learn a transferable and generalizable DA.

Robustness. To assess the robustness of our graph data augmentation method, we conduct additional experiments using an unsupervised graph poisoning attack, CLGA [65], to adversarially perturb the graph structures at varying attack rates. The results of this analysis can be found in Table 2 and Table 5. Intriguingly, we observe that the advanced link prediction models, like *SEAL*, *ELPH* and *NCNC*, exhibits a higher vulnerability to adversarial attacks compared to other baseline models. The capability to capture complex structural relationships of these expressive models renders them more sensitive to structural changes. However, our proposed method, CORE, leverages the robustness inherent in IB [2, 58] to enhance the model resilience by pruning spurious or even harmful perturbations. These findings suggest that the performance improvement offered by our method may be attributed to its ability to mitigate model vulnerability to adversarial perturbations.

GIN as backbone. We examine whether CORE remains an effective DA technique when utilizing a different backbone model. In this case, we choose the Graph Isomorphism Network (*GIN*) [60],

Table 1: Link prediction performance evaluated by Hits@50. The best-performing method is highlighted in bold, while the second-best performance is underlined. OOM means out of memory.

Model Type	Models	C.ele	USAir	Yeast	Router	CS	Physics	Computers	Collab
Heuristics	<i>CN</i>	54.31 \pm 0.00	82.59 \pm 0.00	72.71 \pm 0.00	9.11 \pm 0.00	38.99 \pm 0.00	63.44 \pm 0.00	25.48 \pm 0.00	61.37 \pm 0.00
	<i>AA</i>	57.34 \pm 0.00	87.53 \pm 0.00	72.71 \pm 0.00	9.11 \pm 0.00	67.44 \pm 0.00	74.38 \pm 0.00	31.14 \pm 0.00	64.17 \pm 0.00
	<i>RA</i>	64.34 \pm 0.00	87.53 \pm 0.00	72.71 \pm 0.00	9.11 \pm 0.00	67.44 \pm 0.00	74.68 \pm 0.00	34.17 \pm 0.00	63.81 \pm 0.00
Network Embedding	<i>Node2Vec</i>	50.82 \pm 3.24	74.12 \pm 2.12	82.11 \pm 2.74	32.53 \pm 4.23	63.32 \pm 3.84	60.72 \pm 1.85	28.48 \pm 3.42	48.88 \pm 0.54
	<i>DeepWalk</i>	48.62 \pm 2.82	73.80 \pm 1.98	81.24 \pm 2.38	31.97 \pm 3.92	64.18 \pm 3.98	60.58 \pm 2.24	27.49 \pm 3.08	50.37 \pm 0.34
	<i>LINE</i>	52.40 \pm 2.02	74.82 \pm 3.40	82.45 \pm 2.75	34.39 \pm 3.86	63.96 \pm 2.83	61.90 \pm 1.93	27.52 \pm 2.98	53.91 \pm 0.00
GNNs	<i>GCN</i>	57.32 \pm 4.52	82.14 \pm 1.99	80.33 \pm 0.73	35.16 \pm 1.60	60.69 \pm 8.56	69.16 \pm 4.61	32.70 \pm 1.97	44.75 \pm 1.07
	<i>SAGE</i>	42.14 \pm 5.62	82.85 \pm 4.01	78.34 \pm 1.08	35.76 \pm 2.97	31.44 \pm 8.24	22.87 \pm 22.53	14.53 \pm 6.28	48.10 \pm 0.81
	<i>SEAL</i>	67.32 \pm 2.71	91.76 \pm 1.17	82.50 \pm 2.08	60.35 \pm 5.72	65.23 \pm 2.08	71.83 \pm 1.44	35.80 \pm 1.38	63.37 \pm 0.69
	<i>ELPH</i>	66.06 \pm 3.00	88.16 \pm 1.21	78.92 \pm 0.78	59.50 \pm 1.89	67.84 \pm 1.27	69.60 \pm 1.22	33.64 \pm 0.77	64.58 \pm 0.32
	<i>NCNC</i>	60.42 \pm 1.89	83.22 \pm 0.82	73.11 \pm 2.07	57.13 \pm 0.66	65.73 \pm 2.57	72.87 \pm 1.80	37.17 \pm 1.86	65.97\pm1.03
DAs	<i>Edge Proposal</i>	70.19 \pm 2.95	86.35 \pm 1.35	81.59 \pm 0.51	36.20 \pm 2.61	62.44 \pm 2.68	70.34 \pm 2.89	33.76 \pm 2.08	65.48 \pm 0.00
	<i>CFLP</i>	54.36 \pm 3.41	89.09 \pm 1.12	73.57 \pm 1.06	50.62 \pm 3.33	OOM	OOM	OOM	OOM
	<i>Node Drop</i>	68.76 \pm 2.77	90.79 \pm 1.40	81.45 \pm 3.10	61.76 \pm 5.72	64.80 \pm 2.52	70.51 \pm 1.87	35.94 \pm 2.30	62.57 \pm 0.96
	<i>Edge Drop</i>	66.92 \pm 4.29	92.12 \pm 0.96	81.92 \pm 1.94	59.66 \pm 7.18	67.27 \pm 1.64	72.52 \pm 1.88	36.91 \pm 0.94	63.20 \pm 0.88
Ours	<i>Complete Only</i>	<u>72.10\pm1.70</u>	91.84 \pm 1.23	82.70 \pm 2.20	63.18 \pm 4.01	67.06 \pm 1.01	71.83 \pm 1.44	35.80 \pm 1.38	63.57 \pm 0.48
	<i>Reduce Only</i>	70.22 \pm 3.69	<u>92.35\pm0.95</u>	<u>84.22\pm1.58</u>	<u>65.40\pm2.27</u>	67.79 \pm 1.50	<u>74.73\pm2.12</u>	<u>37.88\pm1.10</u>	64.24 \pm 0.60
	CORE	76.34\pm1.65	93.14\pm1.09	84.67\pm1.13	65.64\pm1.28	69.67\pm1.36	74.73\pm2.12	37.88\pm1.10	<u>65.62\pm0.50</u>
<i>p</i> -values		0.0001**	0.0394**	0.0096**	0.0105**	0.0060**	0.0486**	0.3126	-

**Figure 3: CORE can enhance the graph structure and even boost heuristics link predictors (Hits@50).**

one of the most expressive GNNs, ensuring that the learned representation can encode structural information and guide downstream data augmentation. The results are presented in upper half of Table 3. We observe that, with *GIN* as the backbone, CORE can still improve link prediction performance over the baseline, yielding a 1% to 10% improvement. It indicates that CORE can be effectively integrated with other backbone models.

Information constraints and stochastic sampling. We further investigate the necessity of retaining the information constraint term in the objective function and the stochastic sampling component in the augmentation process. The results are displayed in the lower half of

Table 3. By setting $\beta = 0$, the Reduce stage of our method loses the ability to constrain the information flow from the inflated graph. This leads to significant performance degradation, suggesting that the regularization term helps prevent the model from overfitting. Additionally, the performance declines when removing the stochastic sampling component and directly applying the attention score as the edge weight. This demonstrates that incorporating sampling in data augmentation can potentially expose the link prediction model to a wider range of augmented data variations.

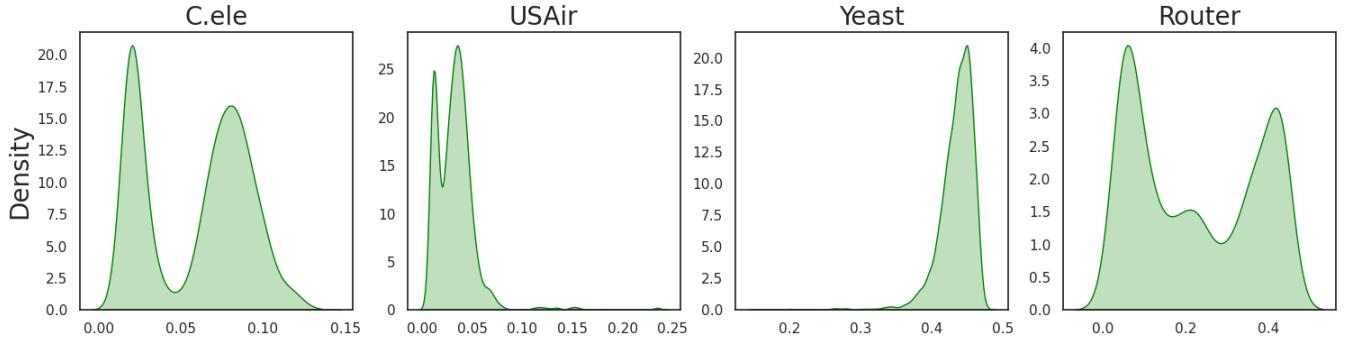


Figure 4: Histogram representing the standard deviations (std) of the learned edge mask ω for each edge within subgraphs associated with different target links. The frequent occurrence of larger std values implies substantial disagreement on the optimal DAs when focusing on different target links.

Table 2: Results of adversarial robustness for different models on *C.ele* and *USAir* datasets. The attack rates of 10%, 30%, and 50% represent the respective ratios of edges subjected to adversarial flips by CLGA [65].

Datasets	Methods	No Adv	10%	30%	50%
C.ele	GCN	57.32 \pm 4.52	59.63 \pm 3.41	54.97 \pm 3.08	46.76 \pm 3.90
	SAGE	42.14 \pm 5.62	31.98 \pm 6.26	35.15 \pm 3.38	28.32 \pm 5.74
	SEAL	67.32 \pm 2.71	60.93 \pm 2.23	58.55 \pm 1.46	51.00 \pm 2.32
	ELPH	66.06 \pm 3.00	62.28 \pm 2.48	56.62 \pm 2.48	50.40 \pm 0.85
	NCNC	60.42 \pm 1.89	52.10 \pm 1.29	53.40 \pm 1.40	50.26 \pm 0.93
	Edge Proposal	70.19 \pm 2.95	64.71 \pm 1.86	58.60 \pm 2.14	50.93 \pm 2.00
	CFLP	54.36 \pm 3.41	51.75 \pm 2.79	46.49 \pm 3.30	42.83 \pm 5.16
	CORE	76.34\pm1.65	72.03\pm3.19	63.78\pm2.24	58.16\pm1.52
USAir	GCN	82.14 \pm 1.99	84.87 \pm 1.22	83.06 \pm 1.73	80.19 \pm 0.77
	SAGE	82.85 \pm 4.01	78.21 \pm 2.81	74.82 \pm 3.28	73.88 \pm 3.65
	SEAL	91.76 \pm 1.17	85.51 \pm 1.70	84.80 \pm 2.95	81.53 \pm 3.95
	ELPH	88.16 \pm 1.21	86.71 \pm 0.94	85.08 \pm 0.96	84.54 \pm 0.50
	NCNC	83.22 \pm 0.82	83.88 \pm 0.78	83.44 \pm 0.50	83.18 \pm 0.53
	Edge Proposal	86.35 \pm 1.35	86.42 \pm 1.34	84.95 \pm 0.74	80.94 \pm 1.66
	CFLP	89.09 \pm 1.12	86.53 \pm 1.74	77.26 \pm 4.24	80.32 \pm 2.44
	CORE	92.69\pm0.75	89.72\pm1.06	88.02\pm1.13	86.71\pm2.06

4.3 Different DAs for different target links

One of the unique designs of our methods is to augment each target link in a separate environment of its own. Here, we empirically investigate the necessity of isolating the DAs. We collect the edge mask ω for each edge within the subgraphs but across different target links. Then, for a set of such edge masks of the same edge, we calculate their standard deviations to indicate how much the learned edge masks ω agree or disagree with each other when augmenting different target links. The results are presented in Figure 4.

As it shows, while a portion of edges may have similar augmentation (small standard deviations), a significant part of them conflicts with each other (large standard deviations). On **C.ele** and **Router**, CORE will learn different DAs for nearly half of the edges associated with different target links. On **Yeast**, the majority of edges are augmented differently by CORE. This result indicates that it is necessary to isolate the DA effect for each target link.

Table 3: Ablation study. The upper half of the table presents results for CORE with GIN as the backbone model, while the bottom half investigates the impact of the balancing hyper-parameter β and edge sampling in the proposed framework.

Methods	C.ele	USAir	Router	Yeast
<i>GIN as the backbone model</i>				
GIN	62.77 \pm 2.33	87.22 \pm 2.70	60.22 \pm 2.09	75.38 \pm 2.23
Complete Only	71.03 \pm 2.18	88.12 \pm 1.47	60.22 \pm 2.09	75.38 \pm 2.23
Reduce Only	64.13 \pm 2.84	88.71 \pm 1.60	62.81 \pm 2.46	78.40 \pm 1.34
CORE	72.33\pm2.62	88.71\pm1.60	62.81\pm2.46	78.40\pm1.34
<i>CORE without sampling or info constraint</i>				
NoSample	75.20 \pm 1.71	91.51 \pm 1.65	64.35 \pm 2.49	84.59\pm1.16
$\beta = 0$	74.02 \pm 2.45	91.81 \pm 1.53	63.90 \pm 2.16	83.33 \pm 2.05
NoSample- $\beta = 0$	73.48 \pm 2.52	91.45 \pm 1.73	65.09 \pm 1.41	84.47 \pm 1.49
CORE	76.34\pm1.65	92.69\pm0.75	65.47\pm2.44	84.22 \pm 1.58

4.4 Additional ablation studies

To further substantiate the efficacy of our proposed DA method, we carry out extensive ablation studies. Due to page constraints, these detailed investigations are presented in the appendix. They include an analysis on the impact of different components of the Reduce stage (see Appendix C.3), a study on parameter sensitivity (see Appendix C.4), and evaluations of CORE when integrated with GCN and SAGE backbones (see Appendix C.5).

5 CONCLUSION

In this paper, we have introduced CORE, a novel data augmentation technique specifically designed for link prediction tasks. Leveraging the Information Bottleneck principle, CORE effectively eliminates noisy and spurious edges while recovering missing edges in the graph, thereby enhancing the generalizability of link prediction models. Our approach yields graph structures that reveal the fundamental relationships inherent in the graph. Extensive experiments on various benchmark datasets have demonstrated the effectiveness and superiority of CORE over competing methods, highlighting its potential as a leading approach for robust link prediction in graph representation learning.

REFERENCES

- [1] Lada A. Adamic and Eytan Adar. 2003. Friends and neighbors on the Web. *Social Networks* 25, 3 (2003), 211–230. [https://doi.org/10.1016/S0378-8733\(03\)00009-1](https://doi.org/10.1016/S0378-8733(03)00009-1)
- [2] Alexander A. Alemi, Ian Fischer, Joshua V. Dillon, and Kevin Murphy. 2023. Deep Variational Information Bottleneck. <https://openreview.net/forum?id=HxQzBceg>
- [3] Martin Arjovsky, Léon Bottou, Ishaaq Gulrajani, and David Lopez-Paz. 2019. Invariant Risk Minimization. <https://arxiv.org/abs/1907.02893v3>
- [4] Vladimir Batagelj and Andrej Mrvar. 2006. Pajek datasets website. <http://vlado.fmf.uni-lj.si/pub/networks/data/>
- [5] Mikhail Belkin. 2021. Fit without fear: remarkable mathematical phenomena of deep learning through the prism of interpolation. <https://doi.org/10.48550/arXiv.2105.14368> [cs, math, stat].
- [6] Sergey Brin and Lawrence Page. 1998. The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer Networks* 30 (1998), 107–117. <http://www-db.stanford.edu/backrub/google.html>
- [7] Benjamin Paul Chamberlain, Sergey Shirokov, Emanuele Rossi, Fabrizio Frasca, Thomas Markovich, Niki Yannick Hammerla, Michael M. Bronstein, and Max Hansmire. 2022. Graph Neural Networks for Link Prediction with Subgraph Sketching. <https://openreview.net/forum?id=m1oqEOAozQU>
- [8] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. 2002. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research* 16 (June 2002), 321–357. <https://doi.org/10.1613/jair.953> arXiv:1106.1813 [cs].
- [9] Yu Chen, Lingfei Wu, and Mohammed Zaki. 2020. Iterative Deep Graph Learning for Graph Neural Networks: Better and Robust Node Embeddings. In *Advances in Neural Information Processing Systems*, Vol. 33. Curran Associates, Inc., 19314–19326. <https://proceedings.neurips.cc/paper/2020/hash/e05c7ba4e087beea9410929698dc41a6-Abstract.html>
- [10] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. 2019. AutoAugment: Learning Augmentation Policies from Data. <https://doi.org/10.48550/arXiv.1805.09501> arXiv:1805.09501 [cs, stat].
- [11] Terrance DeVries and Graham W. Taylor. 2017. Improved Regularization of Convolutional Neural Networks with Cutout. <https://doi.org/10.48550/arXiv.1708.04552> arXiv:1708.04552 [cs].
- [12] Kaiwen Dong, Yijun Tian, Zhihun Guo, Yang Yang, and Nitesh Chawla. 2022. FakeEdge: Alleviate Dataset Shift in Link Prediction. <https://openreview.net/forum?id=QDN0jSXuvT>
- [13] Paul Erdos. 1959. On random graphs. *Publicationes mathematicae* 6 (1959), 290–297.
- [14] Steven Y. Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. 2021. A Survey of Data Augmentation Approaches for NLP. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. Association for Computational Linguistics, Online, 968–988. <https://doi.org/10.18653/v1/2021.findings-acl.84>
- [15] Matthias Fey and Jan E. Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.
- [16] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural Message Passing for Quantum Chemistry. *CoRR* abs/1704.01212 (2017). <http://arxiv.org/abs/1704.01212> arXiv: 1704.01212.
- [17] William L. Hamilton, Rex Ying, and Jure Leskovec. 2018. Inductive Representation Learning on Large Graphs. *arXiv:1706.02216* [cs, stat] (Sept. 2018). <http://arxiv.org/abs/1706.02216> arXiv: 1706.02216
- [18] Paul Holland, Kathryn B. Laskey, and Samuel Leinhardt. 1983. Stochastic block-models: First steps. *Social Networks* 5 (1983), 109–137. <https://api.semanticscholar.org/CorpusID:34098453>
- [19] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. 1989. Multilayer feed-forward networks are universal approximators. *Neural Networks* 2, 5 (Jan. 1989), 359–366. [https://doi.org/10.1016/0893-6080\(89\)90020-8](https://doi.org/10.1016/0893-6080(89)90020-8)
- [20] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2021. Open Graph Benchmark: Datasets for Machine Learning on Graphs. *arXiv:2005.00687* [cs, stat] (Feb. 2021). <https://arxiv.org/abs/2005.00687> arXiv: 2005.00687
- [21] EunJeong Hwang, Veronika Thost, Shib Sankar Dasgupta, and Tengfei Ma. 2022. An Analysis of Virtual Nodes in Graph Neural Networks for Link Prediction (Extended Abstract). <https://openreview.net/forum?id=dI6KBKNP7>
- [22] Eric Jang, Shixiang Gu, and Ben Poole. 2023. Categorical Reparameterization with Gumbel-Softmax. <https://openreview.net/forum?id=rkE3y85ee>
- [23] Leo Katz. 1953. A new status index derived from sociometric analysis. *Psychometrika* 18, 1 (March 1953), 39–43. <https://doi.org/10.1007/BF02289026>
- [24] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. *arXiv:1312.6114* [cs, stat] (May 2014). <http://arxiv.org/abs/1312.6114> arXiv: 1312.6114.
- [25] Thomas N. Kipf and Max Welling. 2016. Variational Graph Auto-Encoders. eprint: 1611.07308.
- [26] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv:1609.02907* [cs, stat] (Feb. 2017). <https://arxiv.org/abs/1609.02907> arXiv: 1609.02907.
- [27] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37. Publisher: IEEE.
- [28] David Krueger, Ethan Caballero, Joern-Henrik Jacobsen, Amy Zhang, Jonathan Binas, Dinghuai Zhang, Remi Le Priol, and Aaron Courville. 2020. Out-of-Distribution Generalization via Risk Extrapolation (REx). <https://arxiv.org/abs/2003.00688v5>
- [29] Lin Li and Michael Spratling. 2023. Data Augmentation Alone Can Improve Adversarial Training. <https://arxiv.org/abs/2301.09879v1>
- [30] Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. 2020. Distance Encoding: Design Provably More Powerful Neural Networks for Graph Representation Learning. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 4465–4478. <https://proceedings.neurips.cc/paper/2020/file/2f73168bf3656f697507752ec592c437-Paper.pdf>
- [31] David Liben-Nowell and Jon Kleinberg. 2003. The link prediction problem for social networks. In *Proceedings of the twelfth international conference on Information and knowledge management (CIKM '03)*. Association for Computing Machinery, New York, NY, USA, 556–559. <https://doi.org/10.1145/956863.956972>
- [32] Linyuan Lu and Tao Zhou. 2011. Link Prediction in Complex Networks: A Survey. *Physica A: Statistical Mechanics and its Applications* 390, 6 (March 2011), 1150–1170. <https://doi.org/10.1016/j.physa.2010.11.027> arXiv:1010.0725 [physics].
- [33] Youzhi Luo, Michael McThrow, Wing Yee Au, Tao Komikado, Kanji Uchino, Koji Maruhashi, and Shuiwang Ji. 2023. Automated Data Augmentations for Graph Classification. <https://doi.org/10.48550/arXiv.2202.13248> arXiv:2202.13248 [cs].
- [34] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2023. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. <https://openreview.net/forum?id=Sj5E5L5gl>
- [35] Siqi Miao, Miaoyuan Liu, and Pan Li. 2022. Interpretable and Generalizable Graph Learning via Stochastic Attention Mechanism. <https://doi.org/10.48550/arXiv.2201.12987> arXiv:2201.12987 [cs].
- [36] Mark EJ Newman. 2006. Finding community structure in networks using the eigenvectors of matrices. *Physical review E* 74, 3 (2006), 036104. Publisher: APS.
- [37] Liming Pan, Cheng Shi, and Ivan Dokmanić. 2022. Neural Link Prediction with Walk Pooling. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=CCuGRcUMwK0>
- [38] Pál András Papp, Karoli Martinkus, Lukas Faber, and Roger Wattenhofer. 2021. DropGNN: Random Dropouts Increase the Expressiveness of Graph Neural Networks. <http://arxiv.org/abs/2111.06283> arXiv:2111.06283 [cs].
- [39] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. 2020. DropEdge: Towards Deep Graph Convolutional Networks on Node Classification. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=Hkx1qkrKPr>
- [40] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2019. Pitfalls of Graph Neural Network Evaluation. <https://doi.org/10.48550/arXiv.1811.05868> arXiv:1811.05868 [cs, stat].
- [41] Connor Shorten and Taghi M. Khoshgoftaar. 2019. A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data* 6, 1 (July 2019), 60. <https://doi.org/10.1186/s40537-019-0197-0>
- [42] 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. <https://doi.org/10.48550/arXiv.2106.15810> arXiv:2106.15810 [cs].
- [43] Krishna Kumar Singh, Hao Yu, Aron Sarmasi, Gautam Pradeep, and Yong Jae Lee. 2018. Hide-and-Seek: A Data Augmentation Technique for Weakly-Supervised Localization and Beyond. <https://doi.org/10.48550/arXiv.1811.02545> arXiv:1811.02545 [cs].
- [44] Neil Spring, Ratul Mahajan, and David Wetherall. 2002. Measuring ISP topologies with Rocketfuel. *ACM SIGCOMM Computer Communication Review* 32, 4 (2002), 133–145. Publisher: ACM New York, NY, USA.
- [45] Qingyun Sun, Jianxin Li, Hao Peng, Jia Wu, Xingcheng Fu, Cheng Ji, and Philip S. Yu. 2021. Graph Structure Learning with Variational Information Bottleneck. <http://arxiv.org/abs/2112.08903> arXiv:2112.08903 [cs].
- [46] Susheel Suresh, Pan Li, Cong Hao, and Jennifer Neville. 2021. Adversarial Graph Augmentation to Improve Graph Contrastive Learning. <https://doi.org/10.48550/arXiv.2106.05819> arXiv:2106.05819 [cs].
- [47] Damian Szkłarczyk, Annika L. Gable, David Lyon, Alexander Junge, Stefan Wyder, Jaime Huerta-Cepas, Milan Simonovic, Nadezhda T. Doncheva, John H. Morris, Peer Bork, Lars J. Jensen, and Christian von Mering. 2019. STRING v11: protein-protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets. *Nucleic Acids Research* 47, D1 (Jan. 2019), D607–D613. <https://doi.org/10.1093/nar/gky1131>
- [48] Naftali Tishby, Fernando C. Pereira, and William Bialek. 2000. The information bottleneck method. <https://doi.org/10.48550/arXiv.physics/0004057> arXiv:physics/0004057.
- [49] Naftali Tishby and Noga Zaslavsky. 2015. Deep learning and the information bottleneck principle. In *2015 IEEE Information Theory Workshop (ITW)*. 1–5. <https://doi.org/10.1109/ITW.2015.7133169>

- [50] Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. 2022. Understanding over-squashing and bottlenecks on graphs via curvature. *arXiv:2111.14522 [cs, stat]* (March 2022). <http://arxiv.org/abs/2111.14522> arXiv: 2111.14522.
- [51] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, Vol. 30. Curran Associates, Inc. https://papers.nips.cc/paper/2017/hash_3f5ee243547dee91fb0d053c1c4a845aa-Abstract.html
- [52] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. *arXiv:1710.10903 [cs, stat]* (Feb. 2018). <http://arxiv.org/abs/1710.10903> arXiv: 1710.10903.
- [53] Christian Von Mering, Roland Krause, Berend Snel, Michael Cornell, Stephen G Oliver, Stanley Fields, and Peer Bork. 2002. Comparative assessment of large-scale data sets of protein–protein interactions. *Nature* 417, 6887 (2002), 399–403. Publisher: Nature Publishing Group.
- [54] Xiyuan Wang, Haotong Yang, and Muhan Zhang. 2023. Neural Common Neighbor with Completion for Link Prediction. <https://doi.org/10.48550/arXiv.2302.00890> arXiv:2302.00890 [cs].
- [55] Yiwei Wang, Wei Wang, Yuxuan Liang, Yujun Cai, Juncheng Liu, and Bryan Hooi. 2020. NodeAug: Semi-Supervised Node Classification with Data Augmentation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '20)*. Association for Computing Machinery, New York, NY, USA, 207–217. <https://doi.org/10.1145/3394486.3403063>
- [56] Duncan J. Watts and Steven H. Strogatz. 1998. Collective dynamics of ‘small-world’ networks. *Nature* 393 (1998), 440–442. <https://api.semanticscholar.org/CorpusID:3034643>
- [57] Qitian Wu, Hengrui Zhang, Junchi Yan, and David Wipf. 2022. Handling Distribution Shifts on Graphs: An Invariance Perspective. <https://openreview.net/forum?id=FQOC5u-1egI>
- [58] Tailin Wu, Hongyu Ren, Pan Li, and Jure Leskovec. 2020. Graph Information Bottleneck. <http://arxiv.org/abs/2010.12811> arXiv:2010.12811 [cs, stat].
- [59] Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V. Le. 2020. Unsupervised Data Augmentation for Consistency Training. <https://doi.org/10.48550/arXiv.1904.12848> arXiv:1904.12848 [cs, stat].
- [60] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How Powerful are Graph Neural Networks? *CoRR abs/1810.00826* (2018). <http://arxiv.org/abs/1810.00826> arXiv: 1810.00826.
- [61] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2021. Graph Contrastive Learning with Augmentations. <https://doi.org/10.48550/arXiv.2010.13902> arXiv:2010.13902 [cs].
- [62] Junchi Yu, Tingyang Xu, Yu Rong, Yatao Bian, Junzhou Huang, and Ran He. 2020. Graph Information Bottleneck for Subgraph Recognition. <https://doi.org/10.48550/arXiv.2010.05563> arXiv:2010.05563 [cs, stat].
- [63] Muhan Zhang and Yixin Chen. 2018. Link Prediction Based on Graph Neural Networks. In *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.), Vol. 31. Curran Associates, Inc. <https://proceedings.neurips.cc/paper/2018/file/53fd7c537d99b3824f0f99d62ea2428-Paper.pdf>
- [64] Muhan Zhang, Pan Li, Yinglong Xia, Kai Wang, and Long Jin. 2021. Labeling Trick: A Theory of Using Graph Neural Networks for Multi-Node Representation Learning. In *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, and J. Wortman Vaughan (Eds.), Vol. 34. Curran Associates, Inc., 9061–9073. <https://proceedings.neurips.cc/paper/2021/file/4be49c79f233b4f407094825c323733-Paper.pdf>
- [65] Sixiao Zhang, Hongxu Chen, Xiangguo Sun, Yicong Li, and Guandong Xu. 2022. Unsupervised Graph Poisoning Attack via Contrastive Loss Back-propagation. In *Proceedings of the ACM Web Conference 2022*, 1322–1330. <https://doi.org/10.1145/3485447.3512179> arXiv:2201.07986 [cs].
- [66] Xiang Zhang, Junbo Zhao, and Yann LeCun. 2016. Character-level Convolutional Networks for Text Classification. <https://doi.org/10.48550/arXiv.1509.01626> arXiv:1509.01626 [cs].
- [67] Tong Zhao, Wei Jin, Yozhen Liu, Yingheng Wang, Gang Liu, Stephan Günnemann, Neil Shah, and Meng Jiang. 2023. Graph Data Augmentation for Graph Machine Learning: A Survey. <https://doi.org/10.48550/arXiv.2202.08871> arXiv: 2202.08871 [cs].
- [68] Tong Zhao, Gang Liu, Daheng Wang, Wenhao Yu, and Meng Jiang. 2022. Learning from Counterfactual Links for Link Prediction. In *Proceedings of the 39th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 162)*, Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (Eds.). PMLR, 26911–26926. <https://proceedings.mlr.press/v162/zha022e.html>
- [69] 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 *Proceedings of the 37th International Conference on Machine Learning*. PMLR, 11458–11468. <https://proceedings.mlr.press/v119/zheng20d.html> ISSN: 2640-3498.
- [70] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. 2017. Random Erasing Data Augmentation. <https://doi.org/10.48550/arXiv.1708.04896> arXiv:1708.04896 [cs].
- [71] 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. Publisher: Springer.
- [72] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. 2018. Adversarial Attacks on Neural Networks for Graph Data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '18)*. Association for Computing Machinery, New York, NY, USA, 2847–2856. <https://doi.org/10.1145/3219819.3220078> event-place: London, United Kingdom.

A RELATED WORKS

Data augmentation. The efficacy of data augmentation strategies in enhancing model generalization is well-documented across various domains. Traditional DA techniques, such as oversampling, undersampling, and interpolation methods [8], have proven to be instrumental in mitigating issues related to learning from imbalanced datasets. In recent years, DA has found extensive application in computer vision and natural language processing. Within the realm of computer vision, techniques such as horizontal flipping, random erasing [70], Hide-and-Seek [43], and Cutout [11] have demonstrated their ability to bolster model performance. On the other hand, in natural language processing, DA is often achieved through lexical substitution strategies, where words are replaced with their semantically equivalent counterparts [66]. UDA [59] introduces a novel approach that leverages TF-IDF metrics for keyword feature augmentation in documents.

Graph data augmentation. Graph-structured data, with its heterogeneous information modalities and complex properties, presents a more intricate landscape for DA compared to conventional image or text data. Typically, graph data augmentation can be bifurcated into two primary approaches: perturbation of graph structure and enhancement of node attributes.

In the realm of semi-supervised node classification, several innovative techniques have been proposed. Drop Edge [39], for instance, introduces random edge dropping to mitigate the oversmoothing problem prevalent in GNNs. Similarly, SDRF [50] leverages graph structure rewiring to address the over-squashing issue in GNNs. NodeAug [55] proposes a more holistic approach by simultaneously augmenting both the graph structure (via edge addition/deletion) and node attributes (via feature replacement).

As for the graph classification task, AD-GCL [46] pioneers an adversarial augmentation technique to boost the training of graph contrastive learning. Concurrently, JOAO [61] and GraphAug [33] automates the selection of augmentations from a predefined pool, incorporating both edge perturbation and node attribute masking.

However, the domain of link prediction has seen relatively limited exploration of DA. Notable exceptions include Distance Encoding [30] and Node labeling [64], which enhance GNNs' expressiveness by infusing distance information. Moreover, Hwang et al. [21] proposes to improve both model expressiveness and node impact by incorporating a virtual node as a message-passing hub for link prediction.

Information bottleneck principle. The Information Bottleneck (IB) principle has been increasingly incorporated into deep learning models to enhance learning robustness. DeepVIB [2], for instance, firstly introduces the application of the IB principle in this domain. To overcome the intractable computation posed by the mutual information term in IB, DeepVIB devises a variational approximation akin to the approach used in Variational Autoencoders (VAEs) [24]. The principle of IB has also been leveraged within the realm of graph representation learning. GIB [58] was among the first to integrate IB into graph learning, aiming to protect GNNs from adversarial attacks. VIB-GSL [45] expanded upon this by applying the IB principle to graph structure learning, demonstrating its robustness in graph classification tasks. GSAT [35], which is the

most related to our work, employs IB to extract the most rationale components from graphs for interpretation purposes. Similarly, IB-subgraph [62] uses IB in conjunction with a bi-level optimization process to identify the most representative subgraph components.

A.1 Comparison to GSAT [35]

Scope & Objective. While GSAT has significantly influenced our research, our study introduces a novel graph DA method designed specifically for link prediction tasks, distinguishing itself from GSAT's focus on graph classification interpretability.

Strategic Design. Unlike the direct application of GIB/GSAT to link prediction, we strategically designed CORE's components:

- **Isolated DA Design:** GSAT when directly applied to the link prediction task resulted in conflicts between optimal DAs for varying target links. We address this by adopting a subgraph-based approach for isolating DA effects per target link. Figure 4 in Section 4.3 showcases how the learned edge masking differs for various downstream target links. The frequent occurrence of larger standard deviations of edge maskings of the same edge for different target links implies substantial disagreement on the optimal DAs. This verifies the necessity of our subgraph-based approach for isolating DA effects per target link.
- **Complete before Reduce:** Recognizing that traditional IB-based methods often overlook data instance information recovery prior to compression, we introduced Complete stage. By recovering missing data, the complete stage can attempt to fulfill the critical assumption (2) in Theorem 1 so that the final DA is predictive and concise. The performance comparison in Table 1 necessitates the need for the Complete stage.

Furthermore, we introduced several improvements to achieve better performance and stability (See the ablation study in Appendix C.3):

- **Attention Mechanism:** When pruning the noisy edges, GIB/GSAT assumes that knowing the edge representation itself is sufficient to determine whether it is a critical substructure. However, in our link prediction task, one edge may be critical for one target link but not for another. We propose using target link representation to discern the criticality of an edge for the said link with attention mechanism, which contrasts with GSAT's approach. (Equation 10)
- **Edge Label:** To differentiate original edges from those introduced in the complete stage, we have adopted a labeling strategy based on their scores to discern their relative importance. (Appendix B.2)
- **Unbiased KL Loss:** In GSAT implementation, the KL loss term is not an unbiased empirical estimator. In a mini-batch B during training, the KL loss term in GSAT is minimized through $\mathbb{E}_G[\text{KL}] \approx \frac{\sum_{G \in B} \text{KL}}{\sum_{G \in B} |E_G|}$. That is, their minimization treats each edge in the batch of graphs as a data instance. However, the unbiased estimator should be $\frac{\sum_{G \in B} \text{KL}}{|B|}$, which average across the subgraphs. The KL loss estimator in GSAT is not that troublesome because the number of edges in their benchmark graphs is similar. However, the number of edges in the subgraphs of the target link varies a lot. The

performance comparison in the ablation study necessitates the unbiased KL loss estimator.

Distinct Utility.

- **Transferability of Graph Structures:** Our method goes beyond previous IB-based graph ML approaches by verifying that our refined graph structures can serve as inputs to other link prediction models like CN, AA or RA. (See Section 4.2)
- **Robustness to Adversarial Attack:** Our method exhibits robustness to adversarial perturbations targeted at link prediction tasks, further validating the capability of our method to capture critical substructures. (See Section 4.2)

In essence, while GSAT inspired our work, our contributions are substantial, addressing nuances of the link prediction task and refining the DA approach for better performance. The ablation studies (Appendix C.3) suggest that applying IB upon link prediction task is a non-trivial work. It requires a dedicated design to suit the unique challenge of link prediction. These distinctions highlight CORE’s value and uniqueness in the graph learning domain.

B IMPLEMENTATION DETAILS

B.1 Marginal distribution

We discuss the marginal distribution term $r(G_{(i,j)}^\pm)$ in Equation 5. Since $\tilde{G}_{(i,j)}$ is sampled based on $G_{(i,j)}^+$ through $\tilde{\omega}_{(u,v)} \sim \text{Bern}(\gamma)$, we can write $r(G_{(i,j)}^\pm) = \sum_{G_{(i,j)}^+} \mathbb{P}(\tilde{\omega}|G_{(i,j)}^+) \mathbb{P}(G_{(i,j)}^+)$. Because $\tilde{\omega}$ is independent from the the inflated graph $G_{(i,j)}^+$ given its size n^\pm , $r(G_{(i,j)}^\pm) = \sum_{n^\pm} \mathbb{P}(\tilde{\omega}|n^\pm) \mathbb{P}(n^\pm) = \mathbb{P}(n^\pm) \prod_{u,v} \mathbb{P}(\tilde{\omega}_{(u,v)})$. Thus, the KL-divergence term in Equation 6 can be written as:

$$\begin{aligned} \text{KL}(p_\phi(G_{(i,j)}^\pm | G_{(i,j)}^+) || r(G_{(i,j)}^\pm)) &= \\ \sum_{(u,v) \in E_{(i,j)}^+} p_{(u,v)} \log \frac{p_{(u,v)}}{\gamma} + (1 - p_{(u,v)}) \log \frac{1 - p_{(u,v)}}{1 - \gamma} \\ + \text{Constant}, \end{aligned} \quad (12)$$

where $p_{(u,v)} = \text{sigmoid}(a_{(u,v)})$ and the constant term accounts for the terms of node pairs $(u,v) \notin E_{(i,j)}^+$ without any trainable parameters.

In practice, we further allow the constraint hyperparameter γ to be different for the original edges in the inflated graph $G_{(i,j)}^+$ and those added in the Complete stage. Namely, we have γ_{ori} and γ_{ext} such that:

$$\begin{aligned} \text{KL}(p_\phi(G_{(i,j)}^\pm | G_{(i,j)}^+) || r(G_{(i,j)}^\pm)) &= \\ \sum_{(u,v) \in E_{(i,j)}^+ \cap E} p_{(u,v)} \log \frac{p_{(u,v)}}{\gamma_{\text{ori}}} + (1 - p_{(u,v)}) \log \frac{1 - p_{(u,v)}}{1 - \gamma_{\text{ori}}} \\ + \sum_{(u,v) \in E_{(i,j)}^+ \cap E_{\text{ext}}} p_{(u,v)} \log \frac{p_{(u,v)}}{\gamma_{\text{ext}}} + (1 - p_{(u,v)}) \log \frac{1 - p_{(u,v)}}{1 - \gamma_{\text{ext}}} \\ + \text{Constant}. \end{aligned} \quad (13)$$

B.2 Awareness of edges scores from the Complete stage

During the Reduce stage, when obtaining the edge representation, $\mathbf{h}(u,v)$, we enrich this representation by appending an additional

encoding, $\tilde{\mathbf{h}}(u,v)$. This supplementary encoding serves to inform the model about the edge’s origin – whether it’s a component of the original graph or an edge added in the Complete stage. However, this encoding strategy does not provide insights into the relative importance of the newly added edges.

To address this, we incorporate a ranking mechanism, assigning a rank label to each added edge based on its relative importance. Specifically, we segregate the added edges into ten equal-sized buckets, determined by their respective scores. Each edge is then assigned a label corresponding to its bucket number. This method of score-aware edge representation enables the model to make more informed use of the added edges, and to discern the most informative edges from the others.

B.3 Augmentation during inference

During the training phase, CORE reduces the inflated graph by implementing edge sampling. However, for the testing phase, we do not employ sampling to obtain the augmented graph.

Given that the entire model can be conceptualized as a probabilistic model, the inference stage requires us to compute the expectation of the random variables. As such, for each edge (u,v) , we use its expected value, $p_{(u,v)}$, as the edge weight during the inference process.

B.4 Nodewise sampling

The implementation of CORE applies an attention mechanism on each edge (u,v) to get $a_{(u,v)}$, which can consume a huge amount of GPU memory when operating on large-scale graphs. As a compromised modification, we follow [35] to sample the node in the inflated graph instead of the edges. More specifically, after we get the node representation, we apply the attention to the node and the subgraph representation to get the importance score a_u for each node. Then, each node is still sampled through a Bernoulli distribution $\omega_u \sim \text{Bern}(\text{sigmoid}(a_u))$. In the end, the edge mask is obtained by $\omega_{(u,v)} = \omega_u * \omega_v$.

B.5 Hyperparameter details

In the Complete stage of our experiments, we employ GCN and SAGE as link predictors to inject potential missing edges into the four non-attributed graphs. This choice is driven by the smaller sizes of these graphs. The parameter k , representing the number of top-scored edges, is searched within the range [1000, 2000].

For the remaining four attributed graphs, we limit our search to heuristic link predictors, namely CN, AA, and RA. The k parameter for these graphs is explored within the range [16000, 32000].

In the Reduce stage, we conduct a hyperparameter search for both γ and β . Specifically, we search for the parameters γ_{ori} and γ_{ext} within the set [0.8, 0.5, 0.2]. The parameter β is searched within the range [1, 0.1, 0.01].

B.6 Software and hardware details

Our implementation leverages the PyTorch Geometric library [15] and the SEAL [63] framework. All experiments were conducted on a Linux system equipped with an NVIDIA P100 GPU with 16GB of memory.

B.7 Time complexity

The time complexity of our method is primarily similar to that of SEAL. However, there are two additional computational requirements: (1) an extra node representation encoding is needed for edge pruning, and (2) a probability score must be assigned to each edge. Consequently, the overall computation complexity of our method is $O(t(d^{l+1}F'' + d^{l+1}F''^2))$, where t represents the number of target links, d is the maximum node degree, l corresponds to the number of hops of the subgraph, and F'' indicates the dimension of the representation.

C SUPPLEMENTARY EXPERIMENTS

C.1 Baseline methods

CN [31]. Common Neighbor (CN) is a widely-used link predictor that posits a node pair with more common neighbors is more likely to connect. The score is computed as $CN(i, j) = |\mathcal{N}_i \cap \mathcal{N}_j|$.

AA [1]. Adamic-Adar (AA) extends the CN approach, emphasizing that common neighbors with fewer connections are more important than those with many connections. The score is calculated as $AA(i, j) = \sum_{z \in \mathcal{N}_i \cap \mathcal{N}_j} \frac{1}{\log|\mathcal{N}_z|}$.

RA [71]. Resource Allocation (RA) modifies the weight decay of AA on common neighbors. It computes the score as $RA(i, j) = \sum_{z \in \mathcal{N}_i \cap \mathcal{N}_j} \frac{1}{|\mathcal{N}_z|}$.

GCN [26]. Graph Convolutional Networks (GCN) propose a graph convolution operation using the first-degree neighbors. Owing to its computational efficiency and high performance, GCN is a popular architecture for GNNs.

SAGE [17]. GraphSAGE (SAGE) proposes a scalable approach to applying GNNs on large graphs. The encoder part of SAGE uses two distinct weights for the center node representation and its surrounding neighbors.

GIN [60]. Graph Isomorphism Network (GIN) is a 1-WL expressive GNN widely used for graph classification problems. In our experiment, we use the zero-one labeling trick [64] in GIN to distinguish between the target node pair and the remaining nodes in the target link's local neighborhood.

SEAL [63]. SEAL is a state-of-the-art link prediction model. It uses GNNs and a node labeling trick [64] to enhance expressiveness for link prediction. We found that SEAL's expected performance on the **Collab** dataset is approximately 5% higher than initially reported.

CFLP [?]. CFLP introduces counterfactual links into the graph to enable causal inference for link prediction. Despite its efficiency for smaller graphs, CFLP can cause out-of-memory issues for larger graphs due to its preprocessing step to find counterfactual node pairs.

Edge Proposal [42]. Edge Proposal augments the graph by adding potential missing or future links to complement the graph for enhanced link prediction performance.

Node Drop [38]. Node Drop, also known as DropGNN, randomly drops nodes in the graph. This exposes the model to multiple views of the graph, enhancing its expressiveness.

Edge Drop [39]. Edge Drop, also known as DropEdge in their work, introduces a stochastic approach to edge removal as a regularization method to solve the oversmoothing issue in GNNs.

C.2 Benchmark datasets

The following graph datasets were utilized in our experiments:

Non-attributed graph datasets:

- (1) **USAir** [4]: This dataset contains a representation of US airlines, encapsulating the connectivity between different airports.
- (2) **Yeast** [53]: This dataset includes a protein-protein interaction network within yeast cells, providing insights into the complex interplay of biological components.
- (3) **C.elegans** [56]: This dataset represents the neural network of the nematode *Caenorhabditis elegans*, one of the most studied organisms in neuroscience.
- (4) **Router** [44]: This dataset encompasses Internet connectivity at the router-level, providing a snapshot of the web's underlying infrastructure.

Attributed graph datasets:

- (1) **CS** [40]: This dataset provides a snapshot of the collaboration network in the computer science domain, highlighting co-authorship relationships.
- (2) **Physics** [40]: This dataset depicts a collaboration network within the field of physics, offering insights into academic partnerships.
- (3) **Computers** [40]: This dataset presents a segment of the co-purchase network on Amazon, reflecting purchasing behavior related to computer products.
- (4) **Collab** [58]: This dataset presents a large-scale collaboration network, showcasing a wide array of interdisciplinary partnerships.

Comprehensive statistics for these datasets are detailed in Table 4. Note that when we perform the train test split, we ensure that the split is the same for all different methods on each dataset.

C.3 More ablation study

We further conduct ablation studies on three unique components we specifically design for link prediction tasks, to enhance the performance of our proposed methods. The results are presented in Table 6. As the results suggest, all three components can boost the performance of the proposed DA method by varying degrees. For instance, CORE with *No Attention* hampers the performance from 0.6 to 2.5 in Hits@50. CORE with *No Edge Label* also drops the effectiveness of DAs by from 0.5 to 3. The *Biased KL Loss* mostly hurts the performance of CORE on **USAir** and **Physics**, which we assume that the degree distribution of these two datasets is more skewed compared to others.

C.4 Parameter sensitivity

We also conducted an experiment to examine the sensitivity of the hyperparameters in CORE. We focused on the Reduce stage only, as this is the core component of our method. As per our hyperparameter search procedure, we evaluated the model performance for β

Table 4: Statistics of benchmark datasets.

Dataset	#Nodes	#Edges	Avg. node deg.	Max. node deg.	Density	Attr. Dimension
C.ele	297	4296	14.46	134	9.7734%	-
USAir	332	4252	12.81	139	7.7385%	-
Yeast	2375	23386	9.85	118	0.8295%	-
Router	5022	12516	2.49	106	0.0993%	-
CS	18333	163788	8.93	136	0.0975%	6805
Physics	34493	495924	14.38	382	0.0834%	8415
Computers	13752	491722	35.76	2992	0.5201%	767
Collab	235868	2358104	10.00	671	0.0085%	128

Table 5: Results of adversarial robustness for different models on the rest of datasets. The attack rates of 10%, 30%, and 50% represent the respective ratios of edges subjected to adversarial flips by CLGA [65].

Methods	Yeast					Router					CS					Physics					Computers				
	No Adv	10%	30%	50%	No Adv	10%	30%	50%	No Adv	10%	30%	50%	No Adv	10%	30%	50%	No Adv	10%	30%	50%	No Adv	10%	30%	50%	
GCN	80.33	76.69	68.30	57.96	35.16	28.55	20.75	15.86	60.69	64.24	57.49	45.28	69.16	68.85	60.16	46.76	32.70	30.16	24.33	20.07	14.53	10.42	4.16	4.34	
SAGE	78.34	74.33	67.01	56.23	35.76	35.13	29.01	25.11	31.44	53.92	42.72	29.16	22.87	61.69	50.20	36.83	33.64	34.30	29.35	23.23	37.17	35.74	33.59	31.53	
ELPH	78.92	76.74	69.05	65.58	59.50	57.07	52.83	47.65	67.84	65.92	60.28	50.09	69.60	64.67	58.83	47.51	35.80	33.84	31.84	28.84	37.88	36.28	34.66	31.85	
NCNC	73.11	71.34	66.34	59.60	57.13	55.04	52.01	47.47	65.73	63.13	53.93	36.66	72.87	69.27	58.74	45.31	34.30	29.35	23.23	20.07	14.53	10.42	4.16	4.34	
SEAL	82.50	78.24	69.24	62.84	60.35	51.84	48.75	44.02	65.23	60.31	58.97	42.38	71.83	64.28	59.84	44.17	35.80	33.84	31.84	28.84	37.17	35.74	33.59	31.53	
CORE	84.67	81.94	74.70	68.96	65.64	59.20	56.58	51.02	69.67	66.87	61.18	50.49	74.73	70.78	62.58	50.37	37.88	36.28	34.66	31.85					

Table 6: Ablation study evaluated by Hits@50. The best-performing components are highlighted in bold, while the second-best performance is underlined.

Models	C.ele	USAir	Router	Yeast	CS	Physics	Computers	Collab
No Attention (Equation 10)	74.41 \pm 1.75	92.38 \pm 1.07	63.60 \pm 2.92	82.81 \pm 0.94	65.69 \pm 2.38	73.10\pm0.78	36.85 \pm 1.17	70.09 \pm 0.94
No Edge Label (Appendix B.2)	74.27 \pm 4.73	91.49 \pm 0.81	63.07 \pm 3.90	<u>83.79\pm2.01</u>	<u>65.84\pm1.40</u>	71.71 \pm 2.72	36.79 \pm 2.88	69.64 \pm 1.32
Biased KL Loss	<u>75.14\pm2.18</u>	90.66 \pm 3.69	<u>64.80\pm2.25</u>	81.91 \pm 1.70	64.22 \pm 3.38	69.32 \pm 5.05	<u>37.31\pm2.47</u>	69.61 \pm 1.13
CORE	76.34\pm1.65	92.69\pm0.75	65.47\pm2.44	84.22\pm1.58	68.15\pm0.78	74.73\pm2.12	37.88\pm1.10	70.64\pm0.51

Table 7: CORE with GCN and SAGE as backbone models. The best-performing methods are highlighted in bold, while the second-best performance is underlined.

Methods	C.ele	USAir	Router	Yeast
<i>GCN as the backbone model</i>				
GCN	62.21 \pm 6.13	83.20 \pm 3.88	43.37 \pm 9.75	81.30 \pm 1.96
Complete Only	65.20 \pm 3.76	85.84 \pm 1.62	<u>53.18\pm6.92</u>	81.54 \pm 1.10
Reduce Only	65.52 \pm 2.95	86.42 \pm 4.03	47.26 \pm 8.37	82.82 \pm 0.96
CORE	68.79\pm2.48	87.96\pm1.03	55.35\pm4.53	82.82\pm0.96
<i>SAGE as the backbone model</i>				
SAGE	70.91 \pm 1.05	80.38 \pm 7.18	56.71 \pm 2.59	84.70 \pm 2.01
Complete Only	71.59 \pm 2.15	83.60 \pm 2.98	58.60 \pm 2.79	84.91 \pm 1.33
Reduce Only	72.12 \pm 1.84	87.32 \pm 3.83	59.54 \pm 2.69	85.47 \pm 0.96
CORE	73.36\pm2.08	89.76\pm2.25	61.75\pm1.07	85.61\pm0.98

Table 8: Number of node pairs with at least one common neighbor as a positive instance in the testing sets.

Dataset	# Node pairs with CN	# Node pairs	Ratio
C.ele	344	429	80.17%
USAir	387	425	91.05%
Yeast	1700	2338	72.71%
Router	114	1251	9.11%
CS	11306	16378	69.03%
Physics	42061	49592	84.81%
Computers	45676	49172	92.89%

across [10, 5, 2, 1, 0.1, 0.01] and γ across [0.8, 0.5, 0.2]. The results are depicted in Figure 5. Our method consistently enhanced the model performance across various hyperparameter settings. However, we also notice that the performance of CORE will drop if we increase the graph compression term β or decrease the edge-preserving term γ . This is because sparsifying the graph too much will result in the

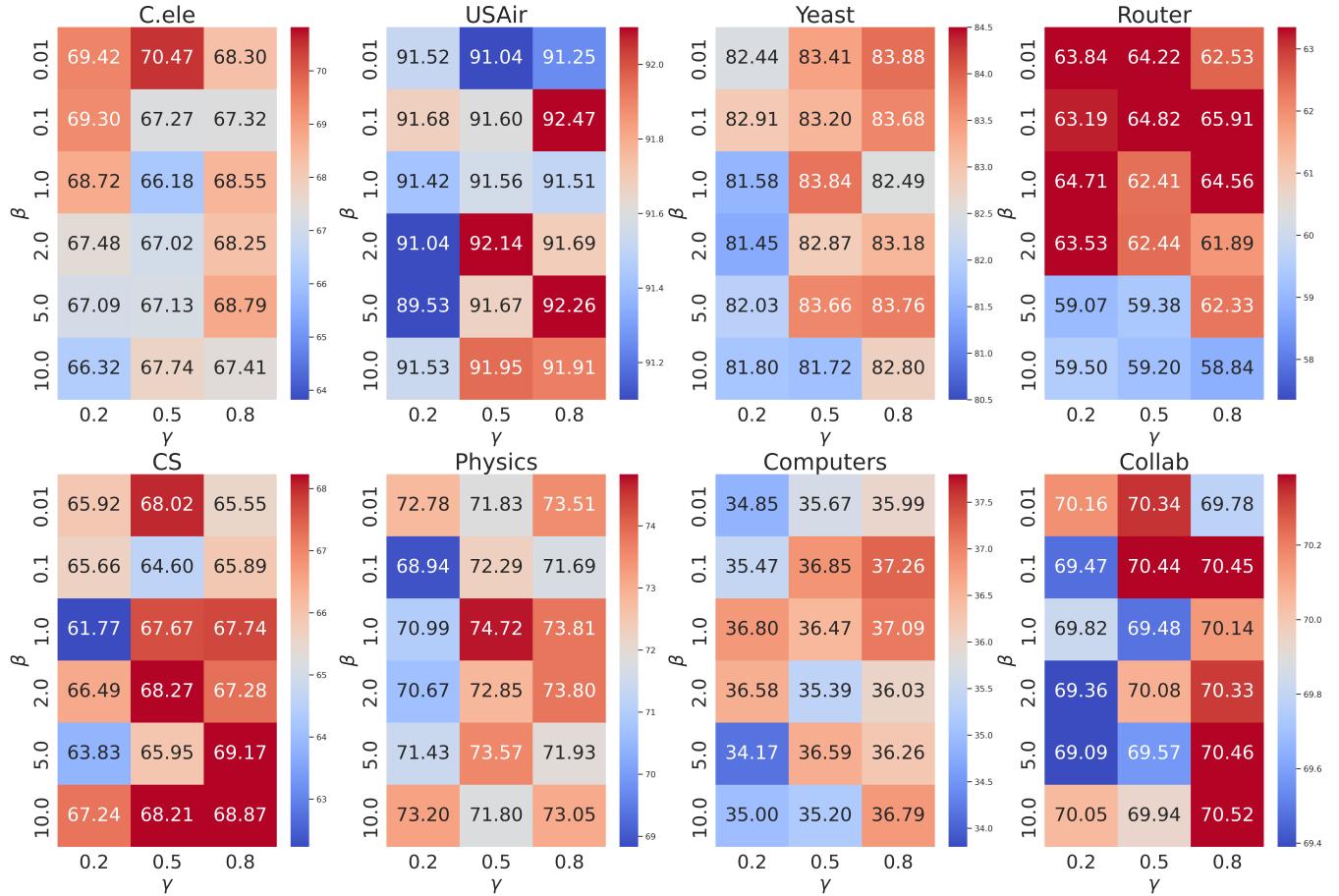


Figure 5: CORE can improve LP performance in various hyperparameter settings measured by Hits@50. Warmer colors indicate improved performance over the baseline, whereas cooler colors signify the contrary.

loss of critical information. Thus, balanced β and γ are crucial for a robust CORE performance.

Besides, the parameter γ acts as a regulatory mechanism that influences each edge's probability, nudging it towards the behavior of a random graph. A lower γ tends to suppress non-essential edges in predictions. Based on our observations, values between $[0.5, 0.8]$ tend to be optimal. That being said, in our studies, the balancing factor β has exhibited a more pronounced effect on model performance than γ .

C.5 CORE with GCN and SAGE as backbones

We further investigate CORE with GCN and SAGE as backbones. The results are presented in Table 7. It shows that CORE can consistently improve model performance with various GNNs as the backbones.

C.6 Complete stage only considering node pairs with common neighbors

When we score the potential node pairs to be added into the graph at Complete stage, we only consider those with at least one common

neighbor. While this seems to limit the capability of recovering missing edges, it is actually an effective approach with balanced computational efficiency. We empirically investigate the number of node pairs in the testing set that have at least one common neighbor, presented in Table 8.

With the exception of the Router dataset, our benchmarks consistently indicate that positive testing edges are inclined to have at least one common neighbor. Therefore, our scoring process at the Complete stage encompasses the majority of the testing edges. This observation underscores that our edge addition strategy aligns well with the community-like nature observed in real-world datasets.

D VARIATIONAL BOUNDS FOR THE GIB OBJECTIVE IN EQUATION 4 AND EQUATION 5

The objective from Equation 3 is:

$$-I(G_{(i,j)}^\pm, Y) + \beta I(G_{(i,j)}^\pm, G_{(i,j)}^+).$$

Since those two terms are computationally intractable, we introduce two variational bounds.

For $I(G_{(i,j)}^\pm, Y)$, by the definition of mutual information:

$$\begin{aligned} I(G_{(i,j)}^\pm, Y) &= \mathbb{E}[\log \frac{p(Y|G_{(i,j)}^\pm)}{p(Y)}] \\ &= \mathbb{E}[\log \frac{q_\theta(Y|G_{(i,j)}^\pm)}{p(Y)}] \\ &\quad + \mathbb{E}[\text{KL}(p(Y|G_{(i,j)}^\pm)||q_\theta(Y|G_{(i,j)}^\pm))] \\ &\geq \mathbb{E}[\log \frac{q_\theta(Y|G_{(i,j)}^\pm)}{p(Y)}] \\ &= \mathbb{E}[\log q_\theta(Y|G_{(i,j)}^\pm)] + H(Y), \end{aligned}$$

where the KL-divergence term is nonnegative.

For the second term $I(G_{(i,j)}^\pm, G_{(i,j)}^+)$, by definition:

$$\begin{aligned} I(G_{(i,j)}^\pm, G_{(i,j)}^+) &= \mathbb{E}[\log \frac{p(G_{(i,j)}^\pm|G_{(i,j)}^+)}{p(G_{(i,j)}^\pm)}] \\ &= \mathbb{E}[\log \frac{p_\phi(G_{(i,j)}^\pm|G_{(i,j)}^+)}{r(G_{(i,j)}^\pm)}] \\ &\quad - \mathbb{E}[\text{KL}(p(G_{(i,j)}^\pm)||r(G_{(i,j)}^\pm))] \\ &\leq \mathbb{E}[\text{KL}(p_\phi(G_{(i,j)}^\pm|G_{(i,j)}^+)||r(G_{(i,j)}^\pm))]. \end{aligned}$$

E PROOF FOR THEOREM 1

We restate Theorem 1: Assume that: (1) The existence Y of a link (i, j) is solely determined by its local neighborhood $G_{(i,j)}^*$ in a way such that $p(Y) = f(G_{(i,j)}^*)$, where f is a deterministic invertible function; (2) The inflated graph contains sufficient structures for prediction $G_{(i,j)}^* \in \mathbb{G}_{\text{sub}}(G_{(i,j)}^+)$. Then $G_{(i,j)}^\pm = G_{(i,j)}^*$ minimizes the objective in Equation 3.

PROOF. We can follow a similar derivation as in [35]. Consider the following steps:

$$\begin{aligned} &- I(G_{(i,j)}^\pm; Y) + \beta I(G_{(i,j)}^\pm; G_{(i,j)}^+) \\ &\quad (\text{Start with the original objective Equation 3}) \\ &= -I(G_{(i,j)}^+, G_{(i,j)}^\pm; Y) + I(G_{(i,j)}^+; Y|G_{(i,j)}^\pm) + \beta I(G_{(i,j)}^\pm; G_{(i,j)}^+) \\ &\quad (\text{Expand the first term via chain rule of mutual information}) \\ &= -I(G_{(i,j)}^+, G_{(i,j)}^\pm; Y) + (1 - \beta)I(G_{(i,j)}^+; Y|G_{(i,j)}^\pm) \\ &\quad + \beta I(G_{(i,j)}^+; G_{(i,j)}^\pm|Y), \end{aligned}$$

$$\begin{aligned} &\quad (\text{Split the third term proportionally}) \\ &= -I(G_{(i,j)}^+; Y) + (1 - \beta)I(G_{(i,j)}^+; Y|G_{(i,j)}^\pm) + \beta I(G_{(i,j)}^+; G_{(i,j)}^\pm|Y) \\ &\quad (\text{Because } G_{(i,j)}^\pm \text{ is a subgraph of } G_{(i,j)}^+) \\ &= (\beta - 1)I(G_{(i,j)}^+; Y) - (\beta - 1)I(G_{(i,j)}^+; Y|G_{(i,j)}^\pm) \\ &\quad + \beta I(G_{(i,j)}^+; G_{(i,j)}^\pm|Y), \end{aligned}$$

(Split the last term and rearrange terms)

Since $I(G_{(i,j)}^+; Y)$ does not involve trainable parameters, we focus on the last two terms. If $\beta \in [0, 1]$, the $G_{(i,j)}^\pm$ that minimizes Equation 3 also minimizes $-(\beta - 1)I(G_{(i,j)}^+; Y|G_{(i,j)}^\pm) + \beta I(G_{(i,j)}^+; G_{(i,j)}^\pm|Y)$. Given that mutual information is nonnegative, the lower bound of $(1 - \beta)I(G_{(i,j)}^+; Y|G_{(i,j)}^\pm) + \beta I(G_{(i,j)}^+; G_{(i,j)}^\pm|Y)$ is 0.

Next, we show that $G_{(i,j)}^\pm = G_{(i,j)}^*$ can make Equation 3 reach its lower bound. Since $p(Y) = f(G_{(i,j)}^*)$, $I(G_{(i,j)}^+; Y|G_{(i,j)}^*) = 0$. That is, there is no mutual information between $G_{(i,j)}^+$ and Y when knowing $G_{(i,j)}^+; Y$. Similarly, because f is invertible, there is no more mutual information between $G_{(i,j)}^+$ and $G_{(i,j)}^\pm$ when knowing Y , yielding $I(G_{(i,j)}^+; G_{(i,j)}^\pm|Y) = 0$. Therefore, $G_{(i,j)}^\pm = G_{(i,j)}^*$ minimizes Equation 3. \square

F LIMITATIONS

In this section, we address the limitations of our proposed method. Firstly, while our model, CORE, delivers superior performance through the application of distinct augmentations for each link prediction, this practice significantly increases the computational burden. This is due to the requirement of independently calculating the augmentation for each link. Attempts to implement a universal augmentation across all links simultaneously resulted in a significant performance drop. Thus, future work may explore efficient methods to balance computational overhead with performance gains.

Secondly, our backbone model, SEAL, utilizes node labeling to determine proximity to the target link for nodes within the neighborhood. Due to computational constraints, this labeling process is performed on the CPU during preprocessing. Despite the capacity of the Reduce stage to alter the local structure of the target link, the node labels remain unchanged post-graph pruning, potentially leading to information leakage about each node's position in the unaltered graph. Future research could investigate methods to fully conceal this information, enabling link prediction to be purely dependent on the pruned graph.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009