# Pure Message Passing Can Estimate Common Neighbor for Link Prediction

**Kaiwen Dong**[1,2] **Zhichun Guo**[1,2] **Nitesh V. Chawla**[1,2]
[1]Computer Science and Engineering, University of Notre Dame
[2]Lucy Family Institute for Data and Society, University of Notre Dame
{kdong2, zguo5, nchawla}@nd.edu

## Abstract

Message Passing Neural Networks (MPNNs) have emerged as the *de facto* standard in graph representation learning. However, when it comes to link prediction, they are not always superior to simple heuristics such as Common Neighbor (CN). This discrepancy stems from a fundamental limitation: while MPNNs excel in node-level representation, they stumble with encoding the joint structural features essential to link prediction, like CN. To bridge this gap, we posit that, by harnessing the orthogonality of input vectors, pure message-passing can indeed capture joint structural features. Specifically, we study the proficiency of MPNNs in approximating CN heuristics. Based on our findings, we introduce the Message Passing Link Predictor (MPLP), a novel link prediction model. MPLP taps into quasi-orthogonal vectors to estimate link-level structural features, all while preserving the node-level complexities. We conduct experiments on benchmark datasets from various domains, where our method consistently outperforms the baseline methods, establishing new state-of-the-arts.

## 1 Introduction

Link prediction is a cornerstone task in the field of graph machine learning, with broad-ranging implications across numerous industrial applications. From identifying potential new acquaintances on social networks [1] to predicting protein interactions [2], from enhancing recommendation systems [3] to completing knowledge graphs [4], the impact of link prediction is felt across diverse domains. Recently, with the advent of Graph Neural Networks (GNNs) [5] and more specifically, Message-Passing Neural Networks (MPNNs) [6], these models have become the primary tools for tackling link prediction tasks. Despite the resounding success of MPNNs in the realm of node and graph classification tasks [5, 7–9], it is intriguing to note that their performance in link prediction does not always surpass that of simpler heuristic methods [10].

Zhang et al. [11] highlights the limitations of GNNs/MPNNs for link prediction tasks arising from its intrinsic property of permutation invariance. Owing to this property, isomorphic nodes invariably receive identical representations. This poses a challenge when attempting to distinguish links whose endpoints are isomorphic nodes. As illustrated in Figure 1a, nodes $v_1$ and $v_3$ share a Common Neighbor $v_2$, while nodes $v_1$ and $v_5$ do not. Ideally, due to their disparate local structures, these two links $(v_1, v_3)$ and $(v_1, v_5)$ should receive distinct predictions. However, the permutation invariance of MPNNs results in identical representations for nodes $v_3$ and $v_5$, leading to identical predictions for the two links. As Zhang et al. [11] asserts, such node-level representation, even with the most expressive MPNNs, **cannot** capture structural link representation such as Common Neighbors (CN), a critical aspect of link prediction.

In this work, we posit that the pure Message Passing paradigm [6] can indeed capture structural link representation by exploiting orthogonality within the vector space. We begin by presenting a

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

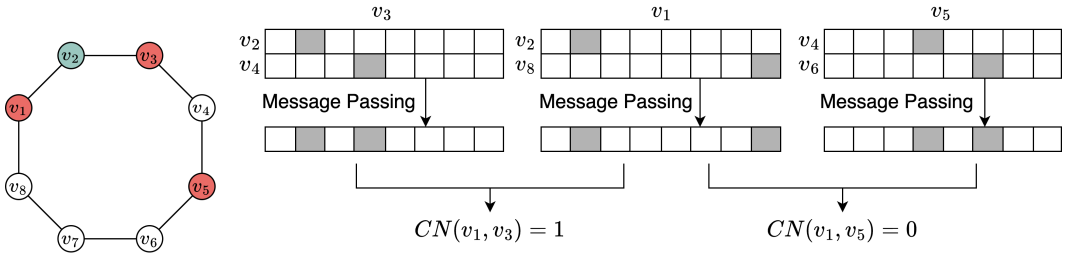



Figure 1: (a) Isomorphic nodes result in identical MPNN node representation, making it impossible to distinguish links such as $(v_1, v_3)$ and $(v_1, v_5)$ based on these representations. (b) MPNN counts Common Neighbor through the inner product of neighboring nodes' one-hot representation.

motivating example, considering a non-attributed graph as depicted in Figure 1a. In order to fulfill the Message Passing's requirement for node vectors as input, we assign a one-hot vector to each node $v_i$, such that the $i$-th dimension has a value of one, with the rest set to zero. These vectors, viewed as *signatures* rather than mere permutation-invariant node representations, can illuminate pairwise relationships. Subsequently, we execute a single iteration of message passing as shown in Figure 1b, updating each node's vector by summing the vector of its neighbors. This process enables us to compute CN for any node pair by taking the inner product of the vectors of the two target nodes.

At its core, this naive method employs an orthonormal basis as the node signatures, thereby ensuring that the inner product of distinct nodes' signatures is consistently zero. While this approach effectively computes CN, its scalability poses a significant challenge, given that its space complexity is quadratically proportional to the size of the graph. To overcome this, we draw inspiration from DotHash [12] and capitalize on the premise that the family of vectors almost orthogonal to each other swells exponentially, even with just linearly scaled dimensions [13]. Instead of relying on the orthogonal basis, we can propagate these quasi-orthogonal (QO) vectors and utilize the inner product to estimate the joint structural information of any node pair.

In sum, our paper presents several pioneering advances in the realm of GNNs for link prediction:

- We are the first, both empirically and theoretically, to delve into the proficiency of GNNs in approximating heuristic predictors like CN for link prediction. This uncovers a previously uncharted territory in GNN research.
- Drawing upon the insights gleaned from GNNs' capabilities in counting CN, we introduce **MPLP**, a novel link prediction model. Uniquely, MPLP discerns joint structures of links and their associated substructures within a graph, setting a new paradigm in the field.
- Our empirical investigations provide compelling evidence of MPLP's dominance. Benchmark tests reveal that MPLP not only holds its own but outstrips state-of-the-art models in link prediction performance.

## 2 Preliminaries and Related Work

**Notations.** Consider an undirected graph $G = (V, E, \boldsymbol{X})$, where $V$ represents the set of nodes with cardinality $n$, indexed as $\{1, \ldots, n\}$, $E \subseteq V \times V$ denotes the observed set of edges, and $\boldsymbol{X}_{i,:} \in \mathbb{R}^{F_x}$ encapsulates the attributes associated with node $i$. Additionally, let $\mathcal{N}_v$ signify the neighborhood of a node $v$, that is $\mathcal{N}_v = \{u | \text{SPD}(u, v) = 1\}$ where the function $\text{SPD}(\cdot, \cdot)$ measures the shortest path distance between two nodes. Furthermore, the node degree of $v$ is given by $d_v = |\mathcal{N}_v|$. To generalize, we introduce the shortest path neighborhood $\mathcal{N}_v^s$, representing the set of nodes that are $s$ hops away from node $v$, defined as $\mathcal{N}_v^s = \{u | \text{SPD}(u, v) = s\}$.

**Link predictions.** Alongside the observed set of edges $E$, there exists an unobserved set of edges, which we denote as $E_c \subseteq V \times V \setminus E$. This unobserved set encompasses edges that are either absent from the original observation or are anticipated to materialize in the future within the graph $G$.

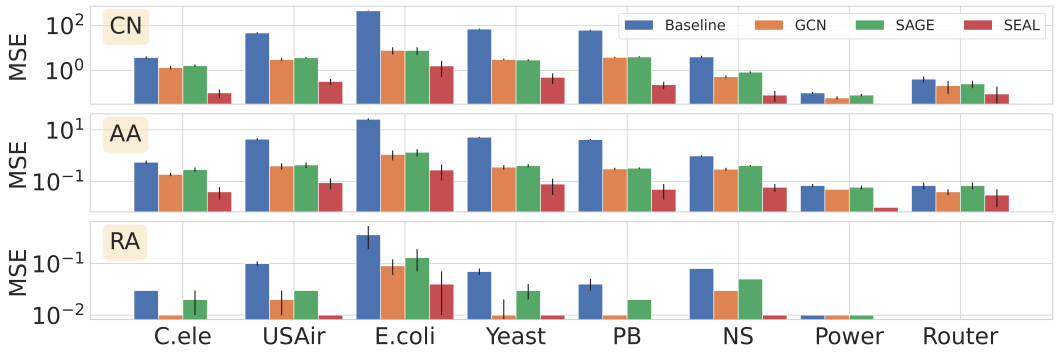



Figure 2: GNNs estimate CN, AA and RA via MSE regression, using the mean value as a Baseline. Lower values are better.

Consequently, we can formulate the link prediction task as discerning the unobserved set of edges $E_c$. Heuristics link predictors include Common Neighbor (CN) [1], Adamic-Adar index (AA) [14], and Resource Allocation (RA) [15]. CN is simply counting the cardinality of the common neighbors, while AA and RA count them weighted to reflect their relative importance as a common neighbor.

$$\text{CN}(u,v) = \sum_{k \in \mathcal{N}_u \bigcap \mathcal{N}_v} 1; \;\; \text{AA}(u,v) = \sum_{k \in \mathcal{N}_u \bigcap \mathcal{N}_v} \frac{1}{\log d_k}; \;\; \text{RA}(u,v) = \sum_{k \in \mathcal{N}_u \bigcap \mathcal{N}_v} \frac{1}{d_k}. \quad (1)$$

Though heuristic link predictors are effective across various graph domains, their growing computational demands clash with the need for low latency. To mitigate this, approaches like ELPH [16] and DotHash [12] propose using estimations rather than exact calculations for these predictors. Our study, inspired by these works, seeks to further refine techniques for efficient link predictions. A detailed comparison with related works and our method is in Appendix A.

**GNNs for link prediction.** The advent of graphs incorporating node attributes has caused a significant shift in research focus toward methods grounded in GNNs. Most practical GNNs follow the paradigm of the Message Passing [6]. It can be formulated as:

$$\boldsymbol{m}_v^{(l)} = \text{AGGREGATE}\left(\{\boldsymbol{h}_v^{(l)}, \boldsymbol{h}_u^{(l)}, \forall u \in \mathcal{N}_v\}\right), \;\; \boldsymbol{h}_v^{(l+1)} = \text{UPDATE}\left(\{\boldsymbol{h}_v^{(l)}, \boldsymbol{m}_v^{(l)}\}\right), \quad (2)$$

where $\boldsymbol{h}_v^{(l)}$ represents the vector of node $v$ at layer $l$ and $\boldsymbol{h}_v^{(0)} = \boldsymbol{X}_{v,:}$. For simplicity, we use $\boldsymbol{h}_v$ to represent the node vector at the last layer. The specific choice of the neighborhood aggregation function, AGGREGATE$(\cdot)$, and the updating function, UPDATE$(\cdot)$, dictates the instantiation of the GNN model, with different choices leading to variations of model architectures. In the context of link prediction tasks, the GAE model [17] derives link representation, $\boldsymbol{h}(i,j)$, as a Hadamard product of the target node pair representations, $\boldsymbol{h}_{(i,j)} = \boldsymbol{h}_i \odot \boldsymbol{h}_j$. Despite its seminal approach, the SEAL model [18], which labels nodes based on proximity to target links and then performs message-passing for each target link, is hindered by computational expense, limiting its scalability. Efficient alternatives like ELPH [16] estimate node labels, while NCNC [19] directly learns edgewise features by aggregating node representations of common neighbors.

## 3 Can Message Passing count Common Neighbor?

In this section, we delve deep into the potential of MPNNs for heuristic link predictor estimation. We commence with an empirical evaluation to recognize the proficiency of MPNNs in approximating link predictors. Following this, we unravel the intrinsic characteristics of 1-layer MPNNs, shedding light on their propensity to act as biased estimators for heuristic link predictors and proposing an unbiased alternative. Ultimately, we cast light on how successive rounds of message passing can estimate the number of walks connecting a target node pair with other nodes in the graph. All proofs are provided in Appendix G.

### 3.1 Estimation via Mean Squared Error Regression

To explore the capacity of MPNNs in capturing the overlap information inherent in heuristic link predictors, such as CN, AA and RA, we conduct an empirical investigation, adopting the GAE

framework [17] with GCN [5] and SAGE [7] as representative encoders. SEAL [18], known for its proven proficiency in capturing heuristic link predictors, serves as a benchmark in our comparison. Additionally, we select a non-informative baseline estimation, simply using the mean of the heuristic link predictors on the training sets. The datasets comprise eight non-attributed graphs (more details in Section 5). Given that GNN encoders require node features for initial representation, we have to generate such features for our non-attributed graphs. We achieved this by sampling from a high-dimensional Gaussian distribution with a mean of 0 and standard deviation of 1. Although one-hot encoding is frequently employed for feature initialization on non-attributed graphs, we choose to forgo this approach due to the associated time and space complexity.

To evaluate the ability of GNNs to estimate CN information, we adopt a training procedure analogous to a conventional link prediction task. However, we reframe the task as a regression problem aimed at predicting heuristic link predictors, rather than a binary classification problem predicting link existence. This shift requires changing the objective function from cross-entropy to Mean Squared Error (MSE). Such an approach allows us to directly observe GNNs' capacity to approximate heuristic link predictors.

Our experimental findings, depicted in Figure 2, reveal that GCN and SAGE both display an ability to estimate heuristic link predictors, albeit to varying degrees, in contrast to the non-informative baseline estimation. More specifically, GCN demonstrates a pronounced aptitude for estimating RA and nearly matches the performance of SEAL on datasets such as C.ele, Yeast, and PB. Nonetheless, both GCN and SAGE substantially lag behind SEAL in approximating CN and AA. In the subsequent section, we delve deeper into the elements within the GNN models that facilitate this approximation of link predictors while also identifying factors that impede their accuracy.

### 3.2 Estimation capabilities of GNNs for link predictors

GNNs exhibit the capability of estimating link predictors. In this section, we aim to uncover the mechanisms behind these estimations, hoping to offer insights that could guide the development of more precise and efficient methods for link prediction. We commence with the following theorem:

**Theorem 3.1.** *Let $G = (V, E)$ be a non-attributed graph and consider a 1-layer GCN/SAGE. Define the input vectors $\boldsymbol{X} \in \mathbb{R}^{N\times F}$ initialized randomly from a zero-mean distribution with standard deviation $\sigma_{node}$. Additionally, let the weight matrix $\boldsymbol{W} \in \mathbb{R}^{F'\times F}$ be initialized from a zero-mean distribution with standard deviation $\sigma_{weight}$. After performing message passing, for any pair of nodes $\{(u, v)|(u, v) \in V \times V \setminus E\}$, the expected value of their inner product is given by:*

$$\text{GCN: } \mathbb{E}(\boldsymbol{h}_u \cdot \boldsymbol{h}_v) = \frac{C}{\sqrt{\hat{d}_u \hat{d}_v}} \sum_{k\in\mathcal{N}_u \bigcap \mathcal{N}_v} \frac{1}{\hat{d}_k}; \quad \text{SAGE: } \mathbb{E}(\boldsymbol{h}_u \cdot \boldsymbol{h}_v) = \frac{C}{\sqrt{d_u d_v}} \sum_{k\in\mathcal{N}_u \bigcap \mathcal{N}_v} 1,$$

*where $\hat{d}_v = d_v + 1$ and $C = \sigma_{node}^2 \sigma_{weight}^2 FF'$.*

The theorem suggests that given proper initialization of input vectors and weight matrices, MPNN-based models, such as GCN and SAGE, can adeptly approximate heuristic link predictors. This makes them apt for encapsulating joint structural features of any node pair. Interestingly, SAGE predominantly functions as a CN estimator, whereas the aggregation function in GCN grants it the ability to weigh the count of common neighbors in a way similar to RA. This particular trait of GCN is evidenced by its enhanced approximation of RA, as depicted in Figure 2.

**Quasi-orthogonal vectors.** The GNN's capability to approximate heuristic link predictors is primarily grounded in the properties of their input vectors in a linear space. When vectors are sampled from a high-dimensional linear space, they tend to be quasi-orthogonal, implying that their inner product is nearly 0 w.h.p. With message-passing, these QO vectors propagate through the graph, yielding in a linear combination of QO vectors at each node. The inner product between pairs of QO vector sets essentially echoes the norms of shared vectors while nullifying the rest. Such a trait enables GNNs to estimate CN through message-passing. A key advantage of QO vectors, especially when compared with orthonormal basis, is their computational efficiency. For a modest linear increment in space dimensions, the number of QO vectors can grow exponentially, given an acceptable margin of error [13]. An intriguing observation is that the orthogonality of QO vectors remains intact even after GNNs undergo linear transformations post message-passing, attributed to the randomized weight matrix initialization. This mirrors the dimension reduction observed in random projection [20].

**Limitations.** While GNNs manifest a marked ability in estimating heuristic link predictors, they are not unbiased estimators and can be influenced by factors such as node pair degrees, thereby

compromising their accuracy. Another challenge when employing such MPNNs is their limited generalization to unseen nodes. The neural networks, exposed to randomly generated vectors, may struggle to transform newly added nodes in the graph with novel random vectors. This practice also violates the permutation-invariance principle of GNNs when utilizing random vectors as node representation. It could strengthen generalizability if we regard these randomly generated vectors as signatures of the nodes, instead of their node features, and circumvent the use of MLPs for them.

**Unbiased estimator.** Addressing the biased element in Theorem 3.1, we propose the subsequent instantiation for the message-passing functions:

$$\boldsymbol{h}_v^{(l+1)} = \sum_{u \in \mathcal{N}_v} \boldsymbol{h}_u^{(l)}. \tag{3}$$

Such an implementation aligns with the SAGE model that employs sum aggregation devoid of self-node propagation. This methodology also finds mention in DotHash [12], serving as a cornerstone for our research. With this kind of message-passing design, the inner product of any node pair signatures can estimate CN impartially:

**Theorem 3.2.** *Let $G = (V, E)$ be a graph, and let the vector dimension be given by $F \in \mathbb{N}_+$. Define the input vectors $\boldsymbol{X} = (X_{i,j})$, which are initialized from a random variable* x *having a mean of* $0$ *and a standard deviation of $\frac{1}{\sqrt{F}}$. Using the 1-layer message-passing in Equation 3, for any pair of nodes $\{(u, v)|(u, v) \in V \times V\}$, the expected value and variance of their inner product are:*

$$\mathbb{E}(\boldsymbol{h}_u \cdot \boldsymbol{h}_v) = \text{CN}(u, v);$$
$$\text{Var}(\boldsymbol{h}_u \cdot \boldsymbol{h}_v) = \frac{1}{F}\left(d_u d_v + \text{CN}(u, v)^2 - 2\text{CN}(u, v)\right) + F\text{Var}\left(\text{x}^2\right)\text{CN}(u, v).$$

Though this estimator provides an unbiased estimate for CN, its accuracy can be affected by its variance. Specifically, DotHash recommends selecting a distribution for input vector sampling from vertices of a hypercube with unit length, which curtails variance given that $\text{Var}\left(\text{x}^2\right) = 0$. However, the variance influenced by the graph structure isn't adequately addressed, and this issue will be delved into in Section 4.

**Orthogonal node attributes.** Both Theorem 3.1 and Theorem 3.2 underscore the significance of quasi orthogonality in input vectors, enabling message-passing to efficiently count CN. Intriguingly, in most attributed graphs, node attributes, often represented as bag-of-words [21], exhibit inherent orthogonality. This brings forth a critical question: In the context of link prediction, do GNNs primarily approximate neighborhood overlap, sidelining the intrinsic value of node attributes? We earmark this pivotal question for in-depth empirical exploration in Appendix E, where we find that random vectors as input to GNNs can catch up with or even outperform node attributes.

### 3.3 Multi-layer message passing

Theorem 3.2 elucidates the estimation of CN based on a single iteration of message passing. This section explores the implications of multiple message-passing iterations and the properties inherent to the iteratively updated node signatures. We begin with a theorem delineating the expected value of the inner product for two nodes' signatures derived from any iteration of message passing:

**Theorem 3.3.** *Under the conditions defined in Theorem 3.2, let $\boldsymbol{h}_u^{(l)}$ denote the vector for node $u$ after the $l$-th message-passing iteration. We have:*

$$\mathbb{E}\left(\boldsymbol{h}_u^{(p)} \cdot \boldsymbol{h}_v^{(q)}\right) = \sum_{k \in V} |\textit{walks}^{(p)}(k, u)||\textit{walks}^{(q)}(k, v)|,$$

*where $|\textit{walks}^{(l)}(u, v)|$ counts the number of length-$l$ walks between nodes $u$ and $v$.*

This theorem posits that the message-passing procedure computes the number of walks between the target node pair and all other nodes. In essence, each message-passing trajectory mirrors the path of the corresponding walk. As such, $\boldsymbol{h}_u^{(l)}$ aggregates the initial QO vectors originating from nodes reachable by length-$l$ walks from node $u$. In instances where multiple length-$l$ walks connect node $k$ to $u$, the associated QO vector $\boldsymbol{X}_{k,:}$ is incorporated into the sum $|\text{walks}^{(l)}(k, u)|$ times.

One might surmise a paradox, given that message-passing calculates the number of walks, not nodes. However, in a simple graph devoid of self-loops, where at most one edge can connect any two nodes, it is guaranteed that $|\text{walks}^{(1)}(u, v)| = 1$ iff $\text{SPD}(u, v) = 1$. Consequently, the quantity of length-1 walks to a target node pair equates to CN, a first-order heuristic. It's essential to recognize, however, that $|\text{walks}^{(l)}(u, v)| \geq 1$ only implies $\text{SPD}(u, v) \leq l$. This understanding becomes vital when employing message-passing for estimating the local structure of a target node pair in Section 4.

# 4 Method

In this section, we introduce our novel link prediction model, denoted as **MPLP**. Distinctively designed, MPLP leverages the pure essence of the message-passing mechanism to adeptly learn joint structural features of the target node pairs.

**Node representation.** While MPLP is specifically designed for its exceptional structural capture, it also embraces the inherent attribute associations of graphs that speak volumes about individual node characteristics. To fuse the attributes (if they exist in the graph) and structures, MPLP begins with a GNN, utilized to encode node $u$'s representation: $\text{GNN}(u) \in \mathbb{R}^{F_x}$. This node representation will be integrated into the structural features when constructing the QO vectors. Importantly, this encoding remains flexible, permitting the choice of any node-level GNN.

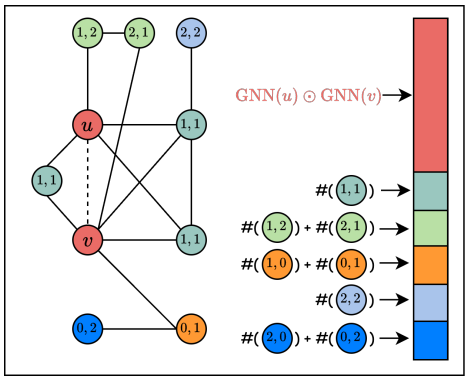



Figure 3: Representation of the target link $(u, v)$ within our model (MPLP), with nodes color-coded based on their distance from the target link.

## 4.1 QO vectors construction

**Probabilistic hypercube sampling.** Though deterministic avenues for QO vector construction are documented [22, 23], our preference leans toward probabilistic techniques for their inherent simplicity. We inherit the sampling paradigm from DotHash [12], where each node $k$ is assigned with a node signature $\boldsymbol{h}_k^{(0)}$, acquired via random sampling from the vertices of an $F$-dimensional hypercube with unit vector norms. Consequently, the sampling space for $\boldsymbol{h}_k^{(0)}$ becomes $\{-1/\sqrt{F}, 1/\sqrt{F}\}^F$.

**Harnessing One-hot hubs for variance reduction.** The stochastic nature of our estimator brings along an inevitable accompaniment: variance. Theorem 3.2 elucidates that a graph's topology can augment estimator variance, irrespective of the chosen QO vector distribution. At the heart of this issue is the imperfectness of quasi-orthogonality. While a pair of vectors might approach orthogonality, the same cannot be confidently said for the subspaces spanned by larger sets of QO vectors.

Capitalizing on the empirical observation that real-world graphs predominantly obey the power-law distribution [24], we propose a strategy to control variance. Leveraging the prevalence of high-degree nodes—or *hubs*—we designate unique one-hot vectors for the foremost hubs. Consider the graph's top-$b$ hubs; while other nodes draw their QO vectors from a hypercube $\{-1/\sqrt{F-b}, 1/\sqrt{F-b}\}^{F-b} \bigtimes \{0\}^b$, these hubs are assigned one-hot vectors from $\{0\}^{F-b} \bigtimes \{0, 1\}^b$, reserving a distinct subspace of the linear space to safeguard orthogonality. Note that when new nodes are added, their QO vectors are sampled the same way as the non-hub nodes, which can ensure a tractable computation complexity.

**Norm rescaling to facilitate weighted counts.** Theorem 3.1 alludes to an intriguing proposition: the estimator's potential to encapsulate not just CN, but also RA. Essentially, RA and AA are nuanced heuristics translating to weighted enumerations of shared neighbors, based on their node degrees. In Theorem 3.2, such counts are anchored by vector norms during dot products. MPLP enhances this count methodology by rescaling node vector norms, drawing inspiration from previous works [12, 25].

This rescaling is determined by the node's representation, GNN$(u)$, and its degree $d_u$. The rescaled vector is formally expressed as:

$$\tilde{\boldsymbol{h}}_k^{(0)} = f(\text{GNN}(k)||[d_k]) \cdot \boldsymbol{h}_k^{(0)}, \tag{4}$$

where $f\colon \mathbb{R}^{F_x+1} \to \mathbb{R}$ is an MLP mapping the node representation and degree to a scalar, enabling the flexible weighted count paradigm.

### 4.2 Structural feature estimations

**Node label estimation.** The estimator in Theorem 3.2 can effectively quantify CN. Nonetheless, solely relying on CN fails to encompass diverse topological structures embedded within the local neighborhood. To offer a richer representation, we turn to Distance Encoding (DE) [26]. DE acts as an adept labeling tool [11], demarcating nodes based on their shortest-path distances relative to a target node pair. For a given pair $(u, v)$, a node $k$ belongs to a node set DE$(p, q)$ iff SPD$(u, k) = p$ and SPD$(v, k) = q$. Unlike its usage as node labels, we opt to enumerate these labels, producing a link feature defined by $\#(p, q) = |\text{DE}(p, q)|$. Our model adopts a philosophy akin to ELPH [16], albeit with a distinct node-estimation mechanism.

Returning to Theorem 3.3, we recall that message-passing as in Equation 3 essentially corresponds to walks. Our ambition to enumerate nodes necessitates a single-layer message-passing alteration, reformulating Equation 3 to:

$$\boldsymbol{\eta}_v^{(s)} = \sum_{k \in \mathcal{N}_v^s} \tilde{\boldsymbol{h}}_k^{(0)}. \tag{5}$$

Here, $\mathcal{N}_v^s$ pinpoints $v$'s shortest-path neighborhoods distanced by the shortest-path $s$. This method sidesteps the duplication dilemma highlighted in Theorem 3.3, ensuring that $\boldsymbol{\eta}_v^{(s)}$ aggregates at most one QO vector per node. Similar strategies are explored in [27, 28].

For a tractable computation, we limit the largest shortest-path distance as $r \geq max(p, q)$. Consequently, to capture the varied proximities of nodes to the target pair $(u, v)$, we can deduce:

$$\#(p,q) = \begin{cases} \mathbb{E}\Big(\boldsymbol{\eta}_u^{(p)} \cdot \boldsymbol{\eta}_v^{(q)}\Big), & r \geq p, q \geq 1 \\ |\mathcal{N}_v^q| - \sum\limits_{1 \leq s \leq r} \#(s,q), & p = 0 \\ |\mathcal{N}_u^p| - \sum\limits_{1 \leq s \leq r} \#(p,s), & q = 0 \end{cases} \tag{6}$$

Concatenating the resulting estimates yields the expressive structural features of MPLP.

**Shortcut removal.** The intricately designed structural features improve the expressiveness of MPLP. However, this augmented expressiveness introduces susceptibility to distribution shifts during link prediction tasks [29]. Consider a scenario wherein the neighborhood of a target node pair contains a node $k$. Node $k$ resides a single hop away from one of the target nodes but requires multiple steps to connect with the other. When such a target node pair embodies a positive instance in the training data (indicative of an existing link), node $k$ can exploit both the closer target node and the link between the target nodes as a shortcut to the farther one. This dynamic ensures that for training-set positive instances, the maximum shortest-path distance from any neighboring node to the target pair is constrained to the smaller distance increased by one. This can engender a discrepancy in distributions between training and testing phases, potentially diminishing the model's generalization capability.

To circumvent this pitfall, we adopt an approach similar to preceding works [18, 30, 19, 31]. Specifically, we exclude target links from the original graph during each training batch, as shown by the dash line in Figure 3. This maneuver ensures these links are not utilized as shortcuts, thereby preserving the fidelity of link feature construction.

**Feature integration for link prediction.** Having procured the structural features, we proceed to formulate the encompassing link representation for a target node pair $(u, v)$ as:

$$\boldsymbol{h}_{(u,v)} = (\text{GNN}(u) \odot \text{GNN}(v))||[\#(1,1), \ldots, \#(r,r)],$$

which can be fed into a classifier for a link prediction between nodes $(u, v)$.

Table 1: Link prediction results on non-attributed benchmarks. The format is average score ± standard deviation. The top three models are colored by **First**, **Second**, **Third**.

| Metric | USAir Hits@50 | NS Hits@50 | PB Hits@50 | Yeast Hits@50 | C.ele Hits@50 | Power Hits@50 | Router Hits@50 | E.coli Hits@50 |
|---|---|---|---|---|---|---|---|---|
| **CN** | 80.52±4.07 | 74.00±1.98 | 37.22±3.52 | 72.60±3.85 | 47.67±10.87 | 11.57±0.55 | 9.38±1.05 | 51.74±2.70 |
| **AA** | 85.51±2.25 | 74.00±1.98 | 39.48±3.53 | 73.62±1.01 | 58.34±2.88 | 11.57±0.55 | 9.38±1.05 | 68.13±1.61 |
| **RA** | 85.95±1.83 | 74.00±1.98 | 38.94±3.54 | 73.62±1.01 | 61.47±4.59 | 11.57±0.55 | 9.38±1.05 | 74.45±0.55 |
| **GCN** | 73.29±4.70 | 78.32±2.57 | 37.32±4.69 | 73.15±2.41 | 40.68±5.45 | 15.40±2.90 | 24.42±4.59 | 61.02±11.91 |
| **SAGE** | 83.81±3.09 | 56.62±9.41 | **47.26±2.53** | 71.06±5.12 | 58.97±4.77 | 6.89±0.95 | 42.25±4.32 | 75.60±2.40 |
| **SEAL** | **90.47±3.00** | 86.59±3.03 | 44.47±2.86 | **83.92±1.17** | **64.80±4.23** | **31.46±3.25** | **61.00±10.10** | 83.42±1.01 |
| **Neo-GNN** | 86.07±1.96 | 83.54±3.92 | 44.04±1.89 | 83.14±0.73 | 63.22±4.32 | 21.98±4.62 | 42.81±4.13 | 73.76±1.94 |
| **ELPH** | 87.60±1.49 | **88.49±2.14** | 46.91±2.21 | 82.74±1.19 | 64.45±3.91 | 26.61±1.73 | **61.07±3.06** | 75.25±1.44 |
| **NCNC** | 86.16±1.77 | 83.18±3.17 | 46.85±3.18 | 82.00±0.97 | 60.49±5.09 | 23.28±1.55 | 52.45±8.77 | **83.94±1.57** |
| **MPLP** | **92.12±2.21** | **90.02±2.04** | **52.55±2.90** | **85.36±0.72** | **74.28±2.09** | **32.66±3.58** | **64.68±3.14** | **86.11±0.83** |
| **MPLP+** | **91.24±2.11** | **88.91±2.04** | **51.81±2.39** | **84.95±0.66** | **72.73±2.99** | **31.86±2.59** | 60.94±2.51 | **87.07±0.89** |

Table 2: Link prediction results on attributed benchmarks. The format is average score ± standard deviation. The top three models are colored by **First**, **Second**, **Third**.

| Metric | CS Hits@50 | Physics Hits@50 | Computers Hits@50 | Photo Hits@50 | Collab Hits@50 | PPA Hits@100 | Citation2 MRR |
|---|---|---|---|---|---|---|---|
| **CN** | 51.04±15.56 | 61.46±6.12 | 21.95±2.00 | 29.33±2.74 | 61.37±0.00 | 27.65±0.00 | 51.47±0.00 |
| **AA** | 68.26±1.28 | 70.98±1.96 | 26.96±2.08 | 37.35±2.65 | 64.35±0.00 | 32.45±0.00 | 51.89±0.00 |
| **RA** | 68.25±1.29 | 72.29±1.69 | 28.05±1.59 | 40.77±3.41 | 64.00±0.00 | 49.33±0.00 | 51.98±0.00 |
| **GCN** | 66.00±2.90 | 73.71±2.28 | 22.95±10.58 | 28.14±7.81 | 35.53±2.39 | 18.67±0.00 | 84.74±0.21 |
| **SAGE** | 57.79±18.23 | 74.10±2.51 | 33.79±3.11 | 46.01±1.83 | 36.82±7.41 | 16.55±0.00 | 82.60±0.36 |
| **SEAL** | 68.50±0.76 | 74.27±2.58 | 30.43±2.07 | 46.08±3.27 | 64.74±0.43 | 48.80±3.16 | **87.67±0.32** |
| **Neo-GNN** | 71.13±1.69 | 72.28±2.33 | 22.76±3.07 | 44.83±3.23 | 57.52±0.37 | 49.13±0.60 | 87.26±0.84 |
| **ELPH**[1] | 72.26±2.58 | 65.80±2.26 | 29.01±2.66 | 43.51±2.37 | 65.94±0.58 | **49.85±0.20** | 87.56±0.11 |
| **NCNC** | **74.65±1.23** | **75.96±1.73** | **36.48±4.16** | **47.98±2.36** | **66.61±0.71** | **61.42±0.73** | **89.12±0.40** |
| **MPLP** | **76.40±1.44** | **76.46±1.95** | **43.47±3.61** | **58.08±3.68** | **67.05±0.51** | OOM | OOM |
| **MPLP+** | **75.55±1.46** | **76.36±1.40** | **42.21±3.56** | **57.76±2.75** | **66.99±0.40** | **65.24±1.50** | **90.72±0.12** |

### 4.3 More scalable estimation

MPLP estimates the cardinality of the distinct node sets with different distances relative to target node pairs in Equation 6. However, this operation requires a preprocessing step to construct the shortest-path neighborhoods $\mathcal{N}_v^s$ for $s \leq r$, which can cause computational overhead on large-scale graph benchmarks. To overcome this issue, we simplify the structural feature estimations as:

$$\#(p, q) = \mathbb{E}\left(\tilde{\boldsymbol{h}}_u^{(p)} \cdot \tilde{\boldsymbol{h}}_v^{(q)}\right), \tag{7}$$

where $\tilde{\boldsymbol{h}}_v^{(l+1)} = \sum_{u \in \mathcal{N}_v} \tilde{\boldsymbol{h}}_u^{(l)}$ follows the message-passing defined in Equation 3. Similar to common GNNs, such a message-passing only requires the one-hop neighborhood $\mathcal{N}_v$, which is provided in a format of adjacency matrices/lists by most graph datasets. Therefore, we can substitute the structural features of MPLP with the estimation in Equation 7. We denote such a model with walk-level features as MPLP+.

### 4.4 Triangular substructure estimation

Our method, primarily designed to encapsulate the local structure of a target node pair, unexpectedly exhibits the capacity for estimating the count of triangles linked to individual nodes. This capability, traditionally considered beyond the reach of GNNs, marks a significant advancement in the field [32]. Although triangle counting is less directly relevant in the context of link prediction, the implications of this capability are noteworthy. To maintain focus, we relegate the detailed discussion on pure message-passing for effective triangle counting to Appendix C.

Table 3: Link prediction results on OGB datasets under HeaRT [33]. The top three models are colored by **First**, **Second**, **Third**.

| Models | Collab | | PPA | | Citation2 | |
|---|---|---|---|---|---|---|
| | MRR | Hits@20 | MRR | Hits@20 | MRR | Hits@20 |
| **CN** | 4.20 | 16.46 | 25.70 | 68.25 | 17.11 | 41.73 |
| **AA** | 5.07 | 19.59 | 26.85 | 70.22 | 17.83 | 43.12 |
| **RA** | 6.29 | 24.29 | 28.34 | 71.50 | 17.79 | 43.34 |
| **GCN** | 6.09 | 22.48 | 26.94 | 68.38 | 19.98 | 51.72 |
| **SAGE** | 5.53 | 21.26 | 27.27 | 69.49 | 22.05 | 53.13 |
| **SEAL** | 6.43 | 21.57 | 29.71 | 76.77 | 20.60 | 48.62 |
| **Neo-GNN** | 5.23 | 21.03 | 21.68 | 64.81 | 16.12 | 43.17 |
| **BUDDY** | 5.67 | 23.35 | 27.70 | 71.50 | 19.17 | 47.81 |
| **NCNC** | 4.73 | 20.49 | 33.52 | 82.24 | 19.61 | 51.69 |
| **MPLP+** | 6.79 | 25.10 | 41.40 | 84.88 | 23.11 | 55.51 |

## 5 Experiments

**Datasets, baselines and experimental setup** We conduct evaluations across a diverse spectrum of 15 graph benchmark datasets, which include 8 non-attributed and 7 attributed graphs [2]. It also includes three datasets from OGB [10] with predefined train/test splits. In the absence of predefined splits, links are partitioned into train, validation, and test sets using a 70-10-20 percent split. Our comparison spans three categories of link prediction models: (1) heuristic-based methods encompassing CN, AA, and RA; (2) node-level models like GCN and SAGE; and (3) link-level models, including SEAL, Neo-GNN [25], ELPH [16], and NCNC [19]. Each experiment is conducted 10 times, with the average score and standard deviations reported. The evaluation metrics are aligned with the standard metrics for OGB datasets, and we utilize Hits@50 for the remaining datasets. We limit the number of hops $r = 2$, which results in a good balance of performance and efficiency. A comprehensive description of the experimental setup is available in Appendix D.

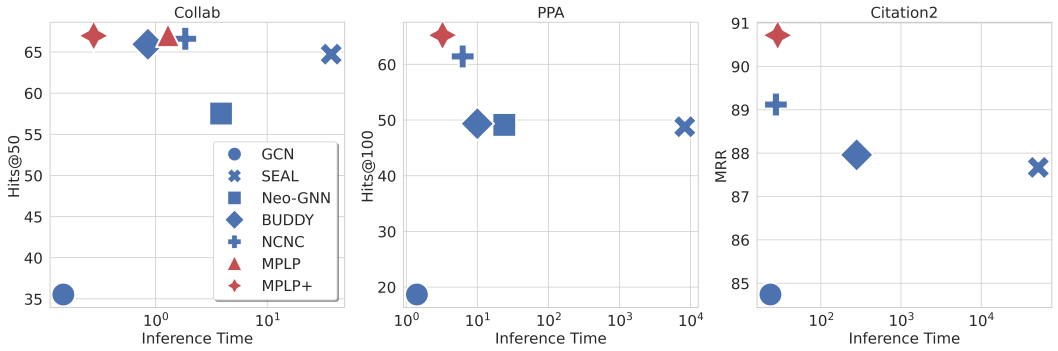



Figure 4: Evaluation of inference time on large-scale OGB datasets. The inference time encompasses the entire cycle within a full-batch inference.

**Results** Performance metrics are shown in Tables 1 and 2. Our methods, MPLP and MPLP+, demonstrate superior performance, surpassing baseline models across all evaluated benchmarks by a significant margin. Notably, MPLP tends to outperform MPLP+ in various benchmarks, suggesting that node-level structural features (Equation 6) might be more valuable for link prediction tasks than the walk-level features (Equation 7). In large-scale graph benchmarks such as PPA and Citation2, MPLP+ sets new benchmarks, establishing state-of-the-art results. For other datasets, our methods

[1] On OGB dataset Collab, PPA, and Citation2, we report the performance of BUDDY [16], a more streamlined version of ELPH. For the rest of the datasets, we report the performance of ELPH, which is empirically better when the computation budget is allowed.

[2] Our code is publicly available at `https://github.com/Barcavin/efficient-node-labelling`.

show a substantial performance uplift, with improvements in Hits@50 ranging from 2% to 10% compared to the closest competitors.

We extend our evaluation of MPLP+ to assess its performance on large-scale datasets under the challenging HeaRT setting proposed by Li et al. [33]. HeaRT introduces a more rigorous and realistic set of negative samples during evaluation, typically resulting in a notable decline in performance across link prediction methods. As detailed in Table 3, MPLP+ consistently outperform all other methods across three OGB graph benchmarks in this demanding context. This underscores the robustness of MPLP+, affirming its ability to maintain superior performance across a variety of graph benchmarks and evaluation settings.

**Time efficiency** We conduct an analysis of the time efficiency of our methods, MPLP and MPLP+, against established baselines using three large-scale OGB datasets. The results, illustrated in Figure 4, demonstrate that our approaches not only deliver superior performance across the graph benchmarks but also set a new benchmark for state-of-the-art time efficiency in full-batch inference. In particular, the primary component underlying our methods is the message-passing operation, which allows their inference speeds to rival that of the baseline GCN. Additionally, the structural feature estimations enhance the models' expressiveness, enabling more accurate representation of graph structures, particularly in the context of link prediction tasks. More details can be found in Appendix D.3.

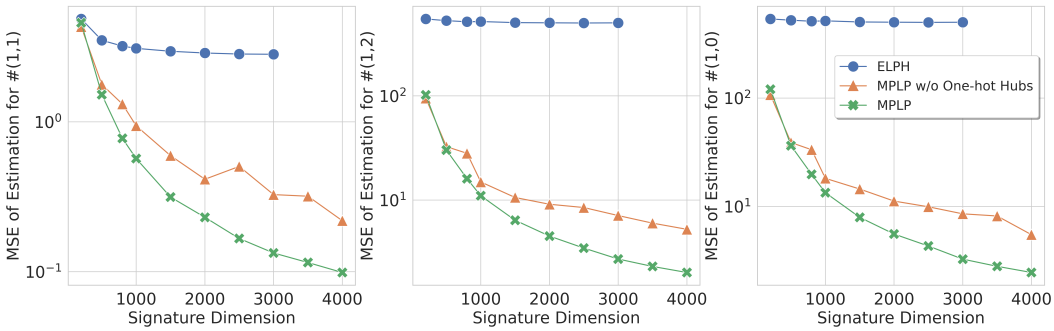



Figure 5: MSE of estimation for $\#(1,1)$, $\#(1,2)$ and $\#(1,0)$ on Collab. Lower values are better.

**Estimation accuracy** We investigate the precision of MPLP in estimating $\#(p,q)$, which denotes the count of node labels, using the Collab dataset. The outcomes of this examination are illustrated in Figure 5. Although ELPH possesses the capability to approximate these counts utilizing techniques like MinHash and Hyperloglog, our method exhibits superior accuracy. Moreover, ELPH runs out of memory when the dimension is larger than 3000. Remarkably, deploying a one-hot encoding strategy for the hubs further bolsters the accuracy of MPLP, concurrently diminishing the variance introduced by inherent graph structures. An exhaustive analysis, including time efficiency considerations, is provided in Appendix F.1.

**Extended ablation studies** Further ablation studies have been carried out to understand the individual contributions within MPLP. These include: (1) an exploration of the distinct components of MPLP in Appendix F.2; (2) an analysis of the performance contributions from different structural estimations in Appendix F.3; and (3) an examination of parameter sensitivity in Appendix F.4.

## 6 Conclusion

We study the potential of message-passing GNNs to encapsulate link structural features. Based on this, we introduce a novel link prediction paradigm that consistently outperforms state-of-the-art baselines across various graph benchmarks. The inherent capability to adeptly capture structures enhances the expressivity of GNNs, all while maintaining their computational efficiency. Our findings hint at a promising avenue for elevating the expressiveness of GNNs through probabilistic approaches.

## 7 Acknowledgements

We would like to thank the anonymous reviewers for their insightful comments and helpful discussions. This research was supported in part by the University of Notre Dame's Lucy Family Institute for Data and Society and the NSF Center for Computer-Assisted Synthesis (C-CAS), under grant number CHE-2202693.

## References

[1] David Liben-Nowell and Jon Kleinberg. The link prediction problem for social networks. In *Proceedings of the twelfth international conference on Information and knowledge management*, CIKM '03, pages 556–559, New York, NY, USA, November 2003. Association for Computing Machinery. ISBN 978-1-58113-723-1. doi: 10.1145/956863.956972. URL http://doi.org/10.1145/956863.956972.

[2] Damian Szklarczyk, 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. STRING v11: protein-protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets. *Nucleic Acids Research*, 47(D1):D607–D613, January 2019. ISSN 1362-4962. doi: 10.1093/nar/gky1131.

[3] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009. Publisher: IEEE.

[4] Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian Tang. Neural bellman-ford networks: A general graph neural network framework for link prediction. *Advances in Neural Information Processing Systems*, 34, 2021.

[5] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv:1609.02907 [cs, stat]*, February 2017. URL http://arxiv.org/abs/1609.02907. arXiv: 1609.02907.

[6] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural Message Passing for Quantum Chemistry. *CoRR*, abs/1704.01212, 2017. URL http://arxiv.org/abs/1704.01212. arXiv: 1704.01212.

[7] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive Representation Learning on Large Graphs. *arXiv:1706.02216 [cs, stat]*, September 2018. URL http://arxiv.org/abs/1706.02216. arXiv: 1706.02216.

[8] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *arXiv:1710.10903 [cs, stat]*, February 2018. URL http://arxiv.org/abs/1710.10903. arXiv: 1710.10903.

[9] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks? *CoRR*, abs/1810.00826, 2018. URL http://arxiv.org/abs/1810.00826. arXiv: 1810.00826.

[10] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open Graph Benchmark: Datasets for Machine Learning on Graphs. *arXiv:2005.00687 [cs, stat]*, February 2021. URL http://arxiv.org/abs/2005.00687. arXiv: 2005.00687.

[11] Muhan Zhang, Pan Li, Yinglong Xia, Kai Wang, and Long Jin. Labeling Trick: A Theory of Using Graph Neural Networks for Multi-Node Representation Learning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 9061–9073. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/4be49c79f233b4f4070794825c323733-Paper.pdf.

[12] Igor Nunes, Mike Heddes, Pere Vergés, Danny Abraham, Alexander Veidenbaum, Alexandru Nicolau, and Tony Givargis. DotHash: Estimating Set Similarity Metrics for Link Prediction and Document Deduplication, May 2023. URL http://arxiv.org/abs/2305.17310. arXiv:2305.17310 [cs].

[13] Paul C. Kainen and Věra Kůrková. Quasiorthogonal dimension of euclidean spaces. *Applied Mathematics Letters*, 6(3):7–10, May 1993. ISSN 0893-9659. doi: 10.1016/0893-9659(93)90023-G. URL https://www.sciencedirect.com/science/article/pii/089396599390023G.

[14] Lada A. Adamic and Eytan Adar. Friends and neighbors on the Web. *Social Networks*, 25(3): 211–230, 2003. ISSN 0378-8733. doi: https://doi.org/10.1016/S0378-8733(03)00009-1. URL https://www.sciencedirect.com/science/article/pii/S0378873303000091.

[15] Tao Zhou, Linyuan Lü, and Yi-Cheng Zhang. Predicting missing links via local information. *The European Physical Journal B*, 71(4):623–630, 2009. Publisher: Springer.

[16] Benjamin Paul Chamberlain, Sergey Shirobokov, Emanuele Rossi, Fabrizio Frasca, Thomas Markovich, Nils Yannick Hammerla, Michael M. Bronstein, and Max Hansmire. Graph Neural Networks for Link Prediction with Subgraph Sketching. September 2022. URL https://openreview.net/forum?id=m1oqEOAozQU.

[17] Thomas N. Kipf and Max Welling. Variational Graph Auto-Encoders, 2016. _eprint: 1611.07308.

[18] Muhan Zhang and Yixin Chen. Link Prediction Based on Graph Neural Networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/53f0d7c537d99b3824f0f99d62ea2428-Paper.pdf.

[19] Xiyuan Wang, Haotong Yang, and Muhan Zhang. Neural Common Neighbor with Completion for Link Prediction, February 2023. URL http://arxiv.org/abs/2302.00890. arXiv:2302.00890 [cs].

[20] William Johnson and Joram Lindenstrauss. Extensions of Lipschitz maps into a Hilbert space. *Contemporary Mathematics*, 26:189–206, January 1984. ISSN 9780821850305. doi: 10.1090/conm/026/737400.

[21] Skye Purchase, Yiren Zhao, and Robert D. Mullins. Revisiting Embeddings for Graph Neural Networks. November 2022. URL https://openreview.net/forum?id=Ri2dzVt_a1h.

[22] Paul C Kainen. Orthogonal dimension and tolerance. *Unpublished report, Washington DC: Industrial Math*, 1992.

[23] Paul C Kainen and Věra Kurkova. Quasiorthogonal dimension. In *Beyond traditional probabilistic data processing techniques: Interval, fuzzy etc. Methods and their applications*, pages 615–629. Springer, 2020.

[24] Albert-László Barabási and Réka Albert. Emergence of Scaling in Random Networks. *Science*, 286(5439):509–512, 1999. doi: 10.1126/science.286.5439.509. URL https://www.science.org/doi/abs/10.1126/science.286.5439.509. _eprint: https://www.science.org/doi/pdf/10.1126/science.286.5439.509.

[25] Seongjun Yun, Seoyoon Kim, Junhyun Lee, Jaewoo Kang, and Hyunwoo J. Kim. Neo-GNNs: Neighborhood Overlap-aware Graph Neural Networks for Link Prediction. November 2021. URL https://openreview.net/forum?id=Ic9vRN3VpZ.

[26] Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. Distance Encoding: Design Provably More Powerful Neural Networks for Graph Representation Learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 4465–4478. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/2f73168bf3656f697507752ec592c437-Paper.pdf.

[27] Ralph Abboud, Radoslav Dimitrov, and Ismail Ilkan Ceylan. Shortest Path Networks for Graph Property Prediction. November 2022. URL https://openreview.net/forum?id=mWzWvMxuFg1.

[28] Jiarui Feng, Yixin Chen, Fuhai Li, Anindya Sarkar, and Muhan Zhang. How Powerful are K-hop Message Passing Graph Neural Networks. May 2022. URL https://openreview.net/forum?id=nN3aVRQsxGd.

[29] Kaiwen Dong, Yijun Tian, Zhichun Guo, Yang Yang, and Nitesh Chawla. FakeEdge: Alleviate Dataset Shift in Link Prediction. December 2022. URL https://openreview.net/forum?id=QDN0jSXuvtX.

[30] Haoteng Yin, Muhan Zhang, Yanbang Wang, Jianguo Wang, and Pan Li. Algorithm and System Co-design for Efficient Subgraph-based Graph Representation Learning. *Proceedings of the VLDB Endowment*, 15(11):2788–2796, July 2022. ISSN 2150-8097. doi: 10.14778/3551793.3551831. URL http://arxiv.org/abs/2202.13538. arXiv:2202.13538 [cs].

[31] Jiarui Jin, Yangkun Wang, Weinan Zhang, Quan Gan, Xiang Song, Yong Yu, Zheng Zhang, and David Wipf. Refined Edge Usage of Graph Neural Networks for Edge Prediction. December 2022. doi: 10.48550/arXiv.2212.12970. URL https://arxiv.org/abs/2212.12970v1.

[32] Zhengdao Chen, Lei Chen, Soledad Villar, and Joan Bruna. Can Graph Neural Networks Count Substructures? *arXiv:2002.04025 [cs, stat]*, October 2020. URL http://arxiv.org/abs/2002.04025. arXiv: 2002.04025.

[33] Juanhui Li, Harry Shomer, Haitao Mao, Shenglai Zeng, Yao Ma, Neil Shah, Jiliang Tang, and Dawei Yin. Evaluating Graph Neural Networks for Link Prediction: Current Pitfalls and New Benchmarking, July 2023. URL http://arxiv.org/abs/2306.10453. arXiv:2306.10453 [cs].

[34] Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, March 1953. ISSN 1860-0980. doi: 10.1007/BF02289026. URL https://doi.org/10.1007/BF02289026.

[35] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., USA, 1986. ISBN 0-07-054484-0.

[36] Sergey Brin and Lawrence Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer Networks*, 30:107–117, 1998. URL http://www-db.stanford.edu/~backrub/google.html.

[37] Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks, November 2021. URL http://arxiv.org/abs/1810.02244. arXiv:1810.02244 [cs, stat].

[38] Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably Powerful Graph Networks. *arXiv:1905.11136 [cs, stat]*, June 2020. URL http://arxiv.org/abs/1905.11136. arXiv: 1905.11136.

[39] Muhan Zhang and Pan Li. Nested Graph Neural Networks, 2021. URL https://arxiv.org/abs/2110.13197.

[40] Fabrizio Frasca, Beatrice Bevilacqua, Michael M. Bronstein, and Haggai Maron. Understanding and Extending Subgraph GNNs by Rethinking Their Symmetries, June 2022. URL http://arxiv.org/abs/2206.11140. arXiv:2206.11140 [cs].

[41] Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Random Features Strengthen Graph Neural Networks, 2021. _eprint: 2002.03155.

[42] Ralph Abboud, Ismail Ilkan Ceylan, Martin Grohe, and Thomas Lukasiewicz. The Surprising Power of Graph Neural Networks with Random Node Initialization, 2021. _eprint: 2010.01179.

[43] Pál András Papp, Karolis Martinkus, Lukas Faber, and Roger Wattenhofer. DropGNN: Random Dropouts Increase the Expressiveness of Graph Neural Networks, November 2021. URL http://arxiv.org/abs/2111.06283. arXiv:2111.06283 [cs].

[44] Vladimir Batagelj and Andrej Mrvar. Pajek datasets website, 2006. URL http://vlado.fmf.uni-lj.si/pub/networks/data/.

[45] Mark EJ Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical review E*, 74(3):036104, 2006. Publisher: APS.

[46] Robert Ackland and others. Mapping the US political blogosphere: Are conservative bloggers more prominent? In *BlogTalk Downunder 2005 Conference, Sydney*, 2005.

[47] Christian Von Mering, Roland Krause, Berend Snel, Michael Cornell, Stephen G Oliver, Stanley Fields, and Peer Bork. Comparative assessment of large-scale data sets of protein–protein interactions. *Nature*, 417(6887):399–403, 2002. Publisher: Nature Publishing Group.

[48] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998. URL https://api.semanticscholar.org/CorpusID:3034643.

[49] Neil Spring, Ratul Mahajan, and David Wetherall. Measuring ISP topologies with Rocketfuel. *ACM SIGCOMM Computer Communication Review*, 32(4):133–145, 2002. Publisher: ACM New York, NY, USA.

[50] Muhan Zhang, Zhicheng Cui, Shali Jiang, and Yixin Chen. Beyond link prediction: Predicting hyperlinks in adjacency space. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[51] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of Graph Neural Network Evaluation, June 2019. URL http://arxiv.org/abs/1811.05868. arXiv:1811.05868 [cs, stat].

[52] Matthias Fey and Jan E. Lenssen. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

# Appendix

## Table of Contents

# A Related work

**Link prediction** Link prediction, inherent to graph data analysis, has witnessed a paradigm shift from its conventional heuristic-based methods to the contemporary, more sophisticated GNNs approaches. Initial explorations in this domain primarily revolve around heuristic methods such as CN, AA, RA, alongside seminal heuristics like the Katz Index [34], Jaccard Index [35], Page Rank [36], and Preferential Attachment [24]. However, the emergence of graphs associated with node attributes has shifted the research landscape towards GNN-based methods. Specifically, these GNN-centric techniques bifurcate into node-level and link-level paradigms. Pioneers like Kipf and Welling introduce the Graph Auto-Encoder (GAE) to ascertain node pair similarity through GNN-generated node representation. On the other hand, link-level models, represented by SEAL [18], opt for subgraph extractions centered on node pairs, even though this can present scalability challenges.

**Amplifying GNN Expressiveness with Randomness** The expressiveness of GNNs, particularly those of the MPNNs, has been the subject of rigorous exploration [9]. A known limitation of MPNNs, their equivalence to the 1-Weisfeiler-Lehman test, often results in indistinguishable representation for non-isomorphic graphs. A suite of contributions has surfaced to boost GNN expressiveness, of which [37–40] stand out. An elegant, yet effective paradigm involves symmetry-breaking through stochasticity injection [41–43]. Although enhancing expressiveness, such random perturbations can occasionally undermine generalizability. Diverging from these approaches, our methodology exploits probabilistic orthogonality within random vectors, culminating in a robust structural feature estimator that introduces minimal estimator variance.

**Link-Level Link Prediction** While node-level models like GAE offer enviable efficiency, they occasionally fall short in performance when compared with rudimentary heuristics [16]. Efforts to build scalable link-level alternatives have culminated in innovative methods such as Neo-GNN [25], which distills structural features from adjacency matrices for link prediction. Elsewhere, ELPH [16] harnesses hashing mechanisms for structural feature representation, while NCNC [19] adeptly aggregates common neighbors' node representation. Notably, DotHash [12], which profoundly influenced our approach, employs quasi-orthogonal random vectors for set similarity computations, applying these in link prediction tasks.

Distinctively, our proposition builds upon, yet diversifies from, the frameworks of ELPH and DotHash. While resonating with ELPH's architectural spirit, we utilize a streamlined, efficacious hashing technique over MinHash for set similarity computations. Moreover, we resolve ELPH's limitations through strategic implementations like shortcut removal and norm rescaling. When paralleled with DotHash, our approach magnifies its potential, integrating it with GNNs for link predictions and extrapolating its applicability to multi-hop scenarios. It also judiciously optimizes variance induced by the structural feature estimator in sync with graph data. We further explore the potential of achieving higher expressiveness with linear computational complexity by estimating the substructure counting [32].

# B Efficient inference at node-level complexity

In addition to its superior performance, MPLP stands out for its practical advantages in industrial applications due to its node-level inference complexity. This design is akin to employing an MLP as the predictor. Our method facilitates offline preprocessing, allowing for the caching of node signatures or representations. Consequently, during online inference in a production setting, MPLP merely requires fetching the relevant node signatures or representations and processing them through an MLP. This approach significantly streamlines the online inference process, necessitating only node-level space complexity and ensuring constant time complexity for predictions. This efficiency in both space and time makes MPLP particularly suitable for real-world applications where rapid, on-the-fly predictions are crucial.

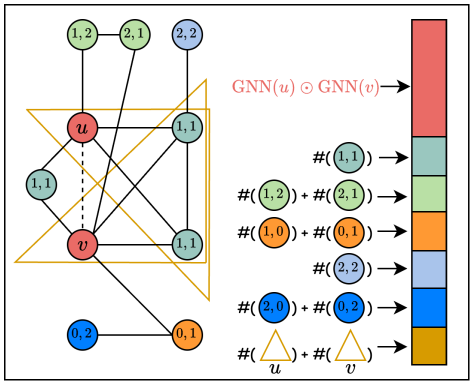
# C Estimate triangular substructures

Not only does MPLP encapsulate the local structure of the target node pair by assessing node counts based on varying shortest-path distances, but it also pioneers in estimating the count of triangles

Table 4: Performance of different GNNs on learning the counts of triangles, measured by MSE divided by variance of the ground truth counts. Shown here are the median (i.e., third-best) performances of each model over five runs with different random seeds.

| Dataset | Erdos-Renyi | Random Regular |
|---|---|---|
| GCN | 8.27E-1 | 2.05 |
| GIN | 1.25E-1 | 4.74E-1 |
| SAGE | 1.48E-1 | 5.21E-1 |
| sGNN | 1.13E-1 | 4.43E-1 |
| 2-IGN | 9.85E-1 | 5.96E-1 |
| PPGN | 2.51E-7 | 3.71E-5 |
| LRP-1-3 | 2.49E-4 | 3.83E-4 |
| Deep LRP-1-3 | 4.77E-5 | 5.16E-6 |
| MPLP | 1.61E-4 | 3.70E-4 |

Figure 6: Representation of the target link $(u, v)$ of MPLP after including the triangular estimation component.




linked to any of the nodes— an ability traditionally deemed unattainable for GNNs [32]. In this section, we discuss a straightforward implementation of the triangle estimation.

### C.1 Method

Constructing the structural feature with DE can provably enhance the expressiveness of the link prediction model [26, 11]. However, there are still prominent cases where labelling trick also fails to capture. Since labelling trick only considers the relationship between the neighbors and the target node pair, it can sometimes miss the subtleties of intra-neighbor relationships. For example, the nodes of DE$(1, 1)$ in Figure 3 exhibit different local structures. Nevertheless, labelling trick like DE tends to treat them equally, which makes the model overlook the triangle substructure shown in the neighborhood. Chen et al. [32] discusses the challenge of counting such a substructure with a pure message-passing framework. We next give an implementation of message-passing to approximate triangle counts linked to a target node pair—equivalent in complexity to conventional MPNNs.

For a triangle to form, two nodes must connect with each other and the target node. Key to our methodology is recognizing the obligatory presence of length-1 and length-2 walks to the target node. Thus, according to Theorem 3.3, our estimation can formalize as:

$$\#(\underset{u}{\triangle}) = \frac{1}{2}\mathbb{E}\left(\tilde{\boldsymbol{h}}_u^{(1)} \cdot \tilde{\boldsymbol{h}}_u^{(2)}\right). \tag{8}$$

Augmenting the structural features with triangle estimates gives rise to a more expressive structural feature set of MPLP.

### C.2 Experiments

Following the experiment in Section 6.1 of [32], we conduct an experiment to evaluate MPLP's ability to count triangular substructures. Similarly, we generate two synthetic graphs as the benchmarks: the Erdos-Renyi graphs and the random regular graphs. We also present the performance of baseline models reported in [32]. Please refer to [32] for details about the experimental settings and baseline models. The results are shown in Table 4.

As the results show, the triangle estimation component of MPLP can estimate the number of triangles in the graph with almost negligible error, similar to other more expressive models. Moreover, MPLP achieves this with a much lower computational cost, which is comparable to 1-WL GNNs like GCN, GIN, and SAGE. It demonstrates MPLP's advantage of better efficiency over more complex GNNs like 2-IGN and PPGN.

Table 5: Statistics of benchmark datasets.

| Dataset | #Nodes | #Edges | Avg. node deg. | Std. node deg. | Max. node deg. | Density | Attr. Dimension |
|---|---|---|---|---|---|---|---|
| **C.ele** | 297 | 4296 | 14.46 | 12.97 | 134 | 9.7734% | - |
| **Yeast** | 2375 | 23386 | 9.85 | 15.50 | 118 | 0.8295% | - |
| **Power** | 4941 | 13188 | 2.67 | 1.79 | 19 | 0.1081% | - |
| **Router** | 5022 | 12516 | 2.49 | 5.29 | 106 | 0.0993% | - |
| **USAir** | 332 | 4252 | 12.81 | 20.13 | 139 | 7.7385% | - |
| **E.coli** | 1805 | 29320 | 16.24 | 48.38 | 1030 | 1.8009% | - |
| **NS** | 1589 | 5484 | 3.45 | 3.47 | 34 | 0.4347% | - |
| **PB** | 1222 | 33428 | 27.36 | 38.42 | 351 | 4.4808% | - |
| **CS** | 18333 | 163788 | 8.93 | 9.11 | 136 | 0.0975% | 6805 |
| **Physics** | 34493 | 495924 | 14.38 | 15.57 | 382 | 0.0834% | 8415 |
| **Computers** | 13752 | 491722 | 35.76 | 70.31 | 2992 | 0.5201% | 767 |
| **Photo** | 7650 | 238162 | 31.13 | 47.28 | 1434 | 0.8140% | 745 |
| **Collab** | 235868 | 2358104 | 10.00 | 18.98 | 671 | 0.0085% | 128 |
| **PPA** | 576289 | 30326273 | 52.62 | 99.73 | 3241 | 0.0256% | 58 |
| **Citation2** | 2927963 | 30561187 | 10.44 | 42.81 | 10000 | 0.0014% | 128 |

# D Experimental details

## D.1 Benchmark datasets

The statistics of each benchmark dataset are shown in Table 5. The benchmarks without attributes are:

- **USAir** [44]: a graph of US airlines;
- **NS** [45]: a collaboration network of network science researchers;
- **PB** [46]: a graph of links between web pages on US political topics;
- **Yeast** [47]: a protein-protein interaction network in yeast;
- **C.ele** [48]: the neural network of Caenorhabditis elegans;
- **Power** [48]: the network of the western US's electric grid;
- **Router** [49]: the Internet connection at the router-level;
- **E.coli** [50]: the reaction network of metabolites in Escherichia coli.

4 out of 7 benchmarks with node attributes come from [51], while Collab, PPA and Citation2 are from Open Graph Benchmark [10]:

- **CS**: co-authorship graphs in the field of computer science, where nodes represent authors, edges represent that two authors collaborated on a paper, and node features indicate the keywords for each author's papers;
- **Physics**: co-authorship graphs in the field of physics with the same node/edge/feature definition as of **CS**;
- **Computers**: a segment of the Amazon co-purchase graph for computer-related equipment, where nodes represent goods, edges represent that two goods are frequently purchased together together, and node features represent the product reviews;
- **Physics**: a segment of the Amazon co-purchase graph for photo-related equipment with the same node/edge/feature definition as of **Computers**;
- **Collab**: a large-scale collaboration network, showcasing a wide array of interdisciplinary partnerships.
- **PPA**: a large-scale protein-protein association network, representing the biological interaction between proteins.

- **Citation2**: a large-scale citation network, with papers as nodes and the citaitons as edges.

Since OGB datasets have a fixed split, no train test split is needed for it. For the other benchmarks, we randomly split the edges into 70-10-20 as train, validation, and test sets. The validation and test sets are not observed in the graph during the entire cycle of training and testing. They are only used for evaluation purposes. For Collab, it is allowed to use the validation set in the graph when evaluating on the test set.

We run the experiments 10 times on each dataset with different splits. For each run, we cache the split edges and evaluate every model on the same split to ensure a fair comparison. The average score and standard deviation are reported in Hits@100 for PPA, MMR for Citation2 and Hits@50 for the remaining datasets.

### D.2 More details in baseline methods

In our experiments, we explore advanced variants of the baseline models **ELPH** and **NCNC**. Specifically, for **ELPH**, Chamberlain et al. [16] propose BUDDY, a link prediction method that preprocesses node representations to achieve better efficiency but compromises its expressiveness. **NCNC** [19] builds upon its predecessor, NCN, by first estimating the complete graph structure and then performing inference. In our experiments, we select the most expressiveness variant to make sure it is a fair comparison between different model architectures. Thus, we select **ELPH** over BUDDY, and **NCNC** over NCN to establish robust baselines in our study. We conduct a thorough hyperparameter tuning for **ELPH** and **NCNC** to select the best-performing models on each benchmark dataset. We follow the hyperparameter guideline of **ELPH** and **NCNC** to search for the optimal structures. For **ELPH**, we run through hyperparameters including dropout rates on different model components, learning rate, batch size, and dimension of node embedding. For **NCNC**, we experiment on dropout rates on different model components, learning rates on different model components, batch size, usage of jumping knowledge, type of encoders, and other model-specific terms like alpha. For **Neo-GNN** and **SEAL**, due to their relatively inferior efficiency, we only tune the common hyperparameters like learning rate, size of hidden dimensions.

### D.3 Evaluation Details: Inference Time

In Figure 4, we assess the inference time across different models on the OGB datasets for a single epoch of test links. Specifically, we clock the wall time taken by models to score the complete test set. This encompasses preprocessing, message-passing, and the actual prediction. For the SEAL model, we employ a dynamic subgraph generator during the preprocessing phase, which dynamically computes the subgraph. We substitute ELPH with BUDDY from [16] in this evaluation, since BUDDY exhibits better time efficiency compared to ELPH. For both BUDDY and our proposed methods, we initially propagate the node features and signatures just once at the onset of inference. These are then cached for subsequent scoring sessions.

### D.4 Software and hardware details

We implement MPLP in Pytorch Geometric framework [52]. We run our experiments on a Linux system equipped with an NVIDIA A100 GPU with 80GB of memory.

### D.5 Time Complexity

The efficiency of MPLP stands out when it comes to link prediction inference. Let's denote $t$ as the number of target links, $d$ as the maximum node degree, $r$ as the number of hops to compute, and $F$ as the dimension count of node signatures.

For preprocessing node signatures, MPLP involves two primary steps:

1. Initially, the algorithm computes all-pairs unweighted shortest paths across the input graph to acquire the shortest-path neighborhood $\mathcal{N}_v^s$ for each node. This can be achieved using a BFS approach for each node, with a time complexity of $O(|V||E|)$.
2. Following this, MPLP propagates the QO vectors through the shortest-path neighborhood, which has a complexity of $O(td^rF)$, and then caches these vectors in memory.

During online scoring, MPLP performs the inner product operation with a complexity of $O(tF)$, enabling the extraction of structural feature estimations.

However, during training, the graph's structure might vary depending on the batch of target links due to the shortcut removal operation. As such, MPLP proceeds in three primary steps:

1. Firstly, the algorithm extracts the $r$-hop induced subgraph corresponding to these $t$ target links. In essence, we deploy a BFS starting at each node of the target links to determine their receptive fields. This process, conceptually similar to message-passing but in a reversed message flow, has a time complexity of $O(tdr)$. Note that, different from SEAL, we extract one $r$-hop subgraph induced from a batch of target links.
2. To identify the shortest-path neighborhood $\mathcal{N}_v^s$, we simply apply sparse-sparse matrix multiplications of the adjacency matrix to get the $s$-power adjacency matrix, where $s = 1, 2, \ldots, r$. Due to the sparsity, this takes $O(|V|d^r)$.
3. Finally, the algorithm engages in message-passing to propagate the QO vectors along the shortest-path neighborhoods, with a complexity of $O(td^rF)$, followed by performing the inner product at $O(tF)$.

Summing up, the overall time complexity for MPLP in the training phase stands at $O(tdr + |V|d^r + td^rF)$.

For MPLP+, it does not require the preprocessing step for the shortest-path neighborhood. Thus, the time complexity is the same as any standard message-passing GNNs, $O(td^rF)$.

### D.6 Hyperparameters

We determine the optimal hyperparameters for our model through systematic exploration. The setting with the best performance on the validation set is selected. The chosen hyperparameters are as follows:

- Number of Hops ($r$): We set the maximum number of hops to $r = 2$. Empirical evaluation suggests this provides an optimal trade-off between accuracy and computational efficiency.
- Node Signature Dimension ($F$): The dimension of node signatures, $F$, is fixed at $1024$, except for Citation2 with $512$. This configuration ensures that MPLP is both efficient and accurate across all benchmark datasets.
- The minimum degree of nodes to be considered as hubs ($b$): This parameter indicates the minimum degree of the nodes which are considered as hubs to one-hot encode in the node signatures. We experiment with values in the set $[50, 100, 150]$.
- Batch Size ($B$): We vary the batch size depending on the graph type: For the 8 non-attributed graphs, we explore batch sizes within $[512, 1024]$. For the 4 attributed graphs coming from [51], we search within $[2048, 4096]$. For OGB datasets, we use $32768$ for Collab and PPA, and $261424$ for Citation2.

More ablation study can be found in Appendix F.4.

## E Exploring Bag-Of-Words Node Attributes

In Section 3, we delved into the capability of GNNs to discern joint structural features, particularly when presented with Quasi-Orthogonal (QO) vectors. Notably, many graph benchmarks utilize text data to construct node attributes, representing them as Bag-Of-Words (BOW). BOW is a method that counts word occurrences, assigning these counts as dimensional values. With a large dictionary, these BOW node attribute vectors often lean towards QO due to the sparse nature of word representations. Consequently, many node attributes in graph benchmarks inherently possess the QO trait. Acknowledging GNNs' proficiency with QO vector input, we propose the question: *Is it the **QO** property or the **information** embedded within these attributes that significantly impacts link prediction in benchmarks?* This section is an empirical exploration of this inquiry.

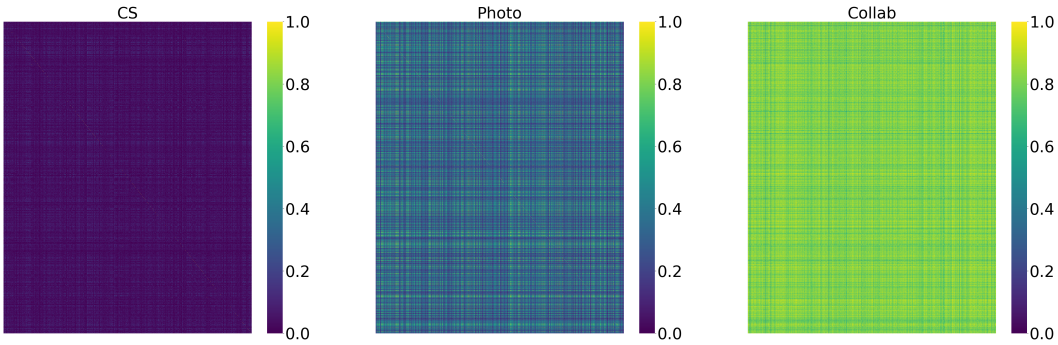



Figure 7: Heatmap illustrating the inner product of node attributes across CS, Photo, and Collab datasets.

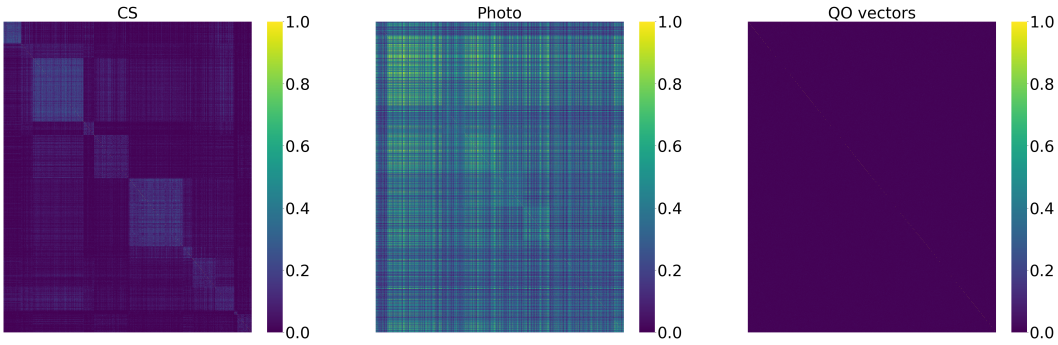



Figure 8: Heatmap illustrating the inner product of node attributes, arranged by node labels, across CS and Photo. The rightmost showcases the inner product of QO vectors.

### E.1 Node Attribute Orthogonality

Our inquiry begins with the assessment of node attribute orthogonality across three attributed graphs: CS, Photo, and Collab. CS possesses extensive BOW vocabulary, resulting in node attributes spanning over $8000$ dimensions. Contrarily, Photo has a comparatively minimal dictionary, encompassing just $745$ dimensions. Collab, deriving node attributes from word embeddings, limits to $128$ dimensions.

For our analysis, we sample $10000$ nodes ($7650$ for Photo) and compute the inner product of their attributes. The results are visualized in Figure 7. Our findings confirm that with a larger BOW dimension, CS node attributes closely follow QO. However, this orthogonality isn't as pronounced in Photo and Collab—especially Collab, where word embeddings replace BOW. Given that increased node signature dimensions can mitigate estimation variance (as elaborated in Theorem 3.2), one could posit GNNs might offer enhanced performance on CS, due to its extensive BOW dimensions. Empirical evidence from Table 2 supports this claim.

Further, in Figure 8, we showcase the inner product of node attributes in CS and Photo, but this time, nodes are sequenced by class labels. This order reveals that nodes sharing labels tend to have diminished orthogonality compared to random pairs—a potential variance amplifier in structural feature estimation using node attributes.

### E.2 Role of Node Attribute Information

To discern the role of embedded information within node attributes, we replace the original attributes in CS, Photo, and Collab with random vectors—denoted as *random feat*. These vectors maintain the original attribute dimensions, though each dimension gets randomly assigned values from $\{-1, 1\}$. The subsequent findings are summarized in Table 6. Intriguingly, even with this "noise" as input, performance remains largely unaltered. CS attributes appear to convey valuable insights for link predictions, but the same isn't evident for the other datasets. In fact, introducing random vectors to Computers and Photo resulted in enhanced outcomes, perhaps due to their original attribute's

Table 6: Performance comparison of GNNs using node attributes versus random vectors (Hits@50). For simplicity, all GNNs are configured with two layers.

| | CS | Physics | Computers | Photo | Collab |
|---|---|---|---|---|---|
| **GCN** | 66.00±2.90 | 73.71±2.28 | 22.95±10.58 | 28.14±7.81 | 35.53±2.39 |
| **GCN(random feat)** | 51.67±2.70 | 69.55±2.45 | 35.86±3.17 | 46.84±2.53 | 17.25±1.15 |
| **SAGE** | 57.79±18.23 | 74.10±2.51 | 1.86±2.53 | 5.70±10.15 | 36.82±7.41 |
| **SAGE(random feat)** | 11.78±1.62 | 64.71±3.65 | 29.23±3.92 | 39.94±3.41 | 28.87±2.36 |
| **Random feat** | | | | | |
| **GCN**($F = 1000$) | 3.73±1.44 | 49.28±2.74 | 36.92±3.36 | 48.72±3.84 | 31.93±2.10 |
| **GCN**($F = 2000$) | 24.97±2.67 | 49.13±4.64 | 40.24±3.04 | 53.49±3.50 | 40.16±1.70 |
| **GCN**($F = 3000$) | 39.51±6.47 | 53.76±3.85 | 42.33±3.82 | 56.27±3.47 | 47.22±1.60 |
| **GCN**($F = 4000$) | 43.23±3.37 | 61.86±4.10 | 42.85±3.60 | 56.87±3.59 | 50.40±1.28 |
| **GCN**($F = 5000$) | 48.25±3.28 | 63.19±4.31 | 44.52±2.78 | 58.13±3.79 | 52.13±1.02 |
| **GCN**($F = 6000$) | 51.44±1.50 | 65.10±4.11 | 44.90±2.74 | 58.10±3.35 | 53.78±0.84 |
| **GCN**($F = 7000$) | 52.00±1.74 | 66.76±3.32 | 45.11±3.69 | 57.41±2.62 | 55.04±1.06 |
| **GCN**($F = 8000$) | 54.21±3.47 | 69.27±2.94 | 44.47±4.11 | 58.67±3.90 | 55.36±1.15 |
| **GCN**($F = 9000$) | 53.16±2.80 | 70.79±2.83 | 45.03±3.13 | 57.15±3.87 | OOM |
| **GCN**($F = 10000$) | 55.91±2.63 | 71.88±3.29 | 45.26±1.94 | 58.12±2.54 | OOM |

insufficient orthogonality hampering effective structural feature capture. Collab shows a performance drop with random vectors, implying that the original word embedding can contribute more to the link prediction than structural feature estimation with merely 128 QO vectors.

### E.3 Expanding QO Vector Dimensions

Lastly, we substitute node attributes with QO vectors of varied dimensions, utilizing GCN as the encoder. The outcomes of this experiment are cataloged in Table 6. What's striking is that GCNs, when furnished with lengthier random vectors, often amplify link prediction results across datasets, with the exception of CS. On Computers and Photo, a GCN even rivals our proposed model (Table 2), potentially attributed to the enlarged vector dimensions. This suggests that when computational resources permit, expanding our main experiment's node signature dimensions (currently set at 1024) could elevate our model's performance. On Collab. the performance increases significantly compared to the experiments which are input with 128-dimensional vectors, indicating that the structural features are more critical for Collab than the word embedding.

## F Additional experiments

### F.1 Node label estimation accuracy and time

In Figure 5, we assess the accuracy of node label count estimation. For ELPH, the node signature dimension corresponds to the number of MinHash permutations. We employ a default hyperparameter setting for Hyperloglog, with $p = 8$, a configuration that has demonstrated its adequacy in [16]. For time efficiency evaluation, we initially propagate and cache node signatures, followed by performing the estimation.

Furthermore, we evaluate the node label count estimation for $\#(2, 2)$ and $\#(2, 0)$. The outcomes are detailed in Figure 9. While MPLP consistently surpasses ELPH in estimation accuracy, the gains achieved via one-hot hubs diminish for $\#(2, 2)$ and $\#(2, 0)$ relative to node counts at a shortest-path distance of 1. This diminishing performance gain can be attributed to our selection criteria for one-hot encoding, which prioritizes nodes that function as hubs within a one-hop radius. However, one-hop hubs don't necessarily serve as two-hop hubs. While we haven't identified a performance drop for these two-hop node label counts, an intriguing avenue for future research would be to refine variance reduction strategies for both one-hop and two-hop estimations simultaneously.

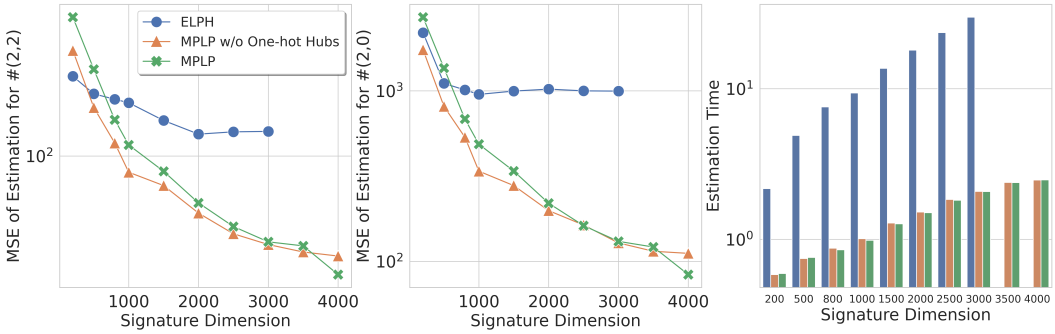



Figure 9: MSE of estimation for $\#(2,2)$, $\#(2,0)$ and estimation time on Collab. Lower values are better.

Table 7: Ablation study on non-attributed benchmarks evaluated by Hits@50. The format is average score ± standard deviation. The top three models are colored by **First**, **Second**, **Third**.

| | USAir | NS | PB | Yeast | C.ele | Power | Router | E.coli |
|---|---|---|---|---|---|---|---|---|
| **w/o Shortcut removal** | 80.94±3.49 | 85.47±2.60 | 49.51±3.57 | 82.62±0.99 | 57.51±2.09 | 19.99±2.54 | 36.67±10.03 | 76.94±1.54 |
| **w/o One-hot hubs** | **84.04±4.53** | **89.45±2.60** | **51.49±2.63** | **85.11±0.62** | **66.85±3.04** | **29.54±1.79** | **50.81±3.74** | **79.07±2.47** |
| **w/o Norm rescaling** | **85.04±2.64** | **89.34±2.79** | **52.50±2.90** | **83.01±1.03** | **66.81±4.11** | **29.00±2.30** | **50.43±3.59** | **79.36±2.18** |
| **MPLP** | **85.19±4.59** | **89.58±2.60** | **52.84±3.39** | **85.11±0.62** | **67.97±2.96** | **29.54±1.79** | **51.04±4.03** | **79.35±2.35** |

Regarding the efficiency of estimation, MPLP consistently demonstrates superior computational efficiency in contrast to ELPH. When we increase the node signature dimension to minimize estimation variance, ELPH's time complexity grows exponentially and becomes impractical. In contrast, MPLP displays a sublinear surge in estimation duration.

It's also worth noting that ELPH exhausts available memory when the node signature dimension surpasses 3000. This constraint arises as ELPH, while estimating structural features, has to cache node signatures for both MinHash and Hyperloglog. Conversely, MPLP maintains efficiency by caching only one type of node signatures.

### F.2 Model enhancement ablation

We investigate the individual performance contributions of three primary components in MPLP: Shortcut removal, One-hot hubs, and Norm rescaling. To ensure a fair comparison, we maintain consistent hyperparameters across benchmark datasets, modifying only the specific component under evaluation. Moreover, node attributes are excluded from the model's input for this analysis. The outcomes of this investigation are detailed in Table 7 and Table 8.

Among the three components, Shortcut removal emerges as the most pivotal for MPLP. This highlights the essential role of ensuring the structural distribution of positive links is aligned between the training and testing datasets [29].

Regarding One-hot hubs, while they exhibited strong results in the estimation accuracy evaluations presented in Figure 5 and Figure 9, their impact on the overall performance is relatively subdued. We

Table 8: Ablation study on attributed benchmarks evaluated by Hits@50. The format is average score ± standard deviation. The top three models are colored by **First**, **Second**, **Third**.

| | CS | Physics | Computers | Photo | Collab |
|---|---|---|---|---|---|
| **w/o Shortcut removal** | 41.63±7.27 | 62.58±2.40 | 32.74±3.03 | 52.09±2.52 | 60.45±1.44 |
| **w/o One-hot hubs** | **65.49±4.28** | **71.58±2.28** | **36.09±4.08** | **55.63±2.48** | **65.07±0.47** |
| **w/o Norm rescaling** | **65.20±2.92** | **67.73±2.54** | **35.83±3.24** | **52.59±3.57** | **63.99±0.59** |
| **MPLP** | **65.70±3.86** | **71.03±3.55** | **37.56±3.57** | **55.63±2.48** | **66.07±0.47** |

Table 9: The mapping between the configuration number and the used structural features in MPLP.

| Configurations | $\#(1,1)$ | $\#(1,2)$ | $\#(1,0)$ | $\#(2,2)$ | $\#(2,0)$ | $\#(\triangle)$ |
|---|---|---|---|---|---|---|
| **(1)** | ✓ | - | - | - | - | - |
| **(2)** | - | ✓ | ✓ | - | - | - |
| **(3)** | - | - | - | ✓ | ✓ | - |
| **(4)** | - | - | - | - | - | ✓ |
| **(5)** | ✓ | ✓ | ✓ | - | - | - |
| **(6)** | ✓ | - | - | ✓ | ✓ | - |
| **(7)** | ✓ | - | - | - | - | ✓ |
| **(8)** | - | ✓ | ✓ | ✓ | ✓ | - |
| **(9)** | - | ✓ | ✓ | - | - | ✓ |
| **(10)** | - | - | - | ✓ | ✓ | ✓ |
| **(11)** | ✓ | ✓ | ✓ | ✓ | ✓ | - |
| **(12)** | ✓ | ✓ | ✓ | - | - | ✓ |
| **(13)** | ✓ | - | - | ✓ | ✓ | ✓ |
| **(14)** | - | ✓ | ✓ | ✓ | ✓ | ✓ |
| **(15)** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

Table 10: Ablation analysis highlighting the impact of various structural features on link prediction. Refer to Table 9 for detailed configurations of the structural features used.

| Configurations | **USAir** | **NS** | **PB** | **Yeast** | **C.ele** | **Power** | **Router** | **E.coli** |
|---|---|---|---|---|---|---|---|---|
| **(1)** | 76.64±26.74 | 75.26±2.79 | 37.48±13.30 | 58.70±30.50 | 46.22±24.84 | 14.40±1.40 | 17.29±3.96 | 60.10±30.80 |
| **(2)** | 82.54±4.61 | 84.76±3.63 | 41.84±15.51 | 80.56±0.65 | 56.22±20.39 | 21.38±1.46 | 48.97±3.34 | 67.78±23.83 |
| **(3)** | 67.76±23.65 | 70.05±2.35 | 44.81±2.63 | 67.02±2.53 | 36.53±19.68 | 25.24±4.07 | 21.32±2.66 | 56.59±1.78 |
| **(4)** | 37.18±37.57 | 25.13±1.99 | 12.35±10.75 | 7.42±10.80 | 30.75±18.69 | 5.47±1.13 | 30.47±3.10 | 34.90±36.63 |
| **(5)** | 86.24±2.70 | 84.91±2.80 | 48.35±3.76 | 84.42±0.56 | 66.69±3.60 | 22.25±1.39 | 49.68±3.79 | 80.94±1.62 |
| **(6)** | 77.41±5.27 | 80.00±2.39 | 46.05±2.76 | 74.70±1.45 | 46.88±5.79 | 27.74±3.23 | 22.37±2.06 | 71.41±2.47 |
| **(7)** | 71.11±25.51 | 76.72±2.37 | 43.57±3.70 | 73.08±1.23 | 54.99±20.14 | 14.50±1.64 | 31.26±2.87 | 80.22±2.09 |
| **(8)** | 80.16±4.82 | 88.67±2.72 | 52.16±2.25 | 82.52±0.85 | 63.82±4.02 | 28.41±2.00 | 50.97±3.57 | 77.26±1.31 |
| **(9)** | 75.13±26.51 | 87.28±3.33 | 48.10±3.43 | 80.84±0.97 | 60.63±4.54 | 23.85±1.37 | 49.78±3.56 | 76.13±1.81 |
| **(10)** | 76.82±4.28 | 77.04±3.70 | 45.42±2.77 | 67.34±3.20 | 41.66±13.47 | 26.95±1.47 | 28.31±2.76 | 70.14±0.77 |
| **(11)** | 82.82±5.52 | 88.91±2.90 | 52.57±3.05 | 84.61±0.67 | 67.11±2.52 | 28.98±1.73 | 50.63±3.72 | 80.16±2.20 |
| **(12)** | 87.29±1.08 | 88.08±2.59 | 48.86±3.42 | 84.59±0.69 | 66.06±3.74 | 23.79±1.87 | 50.06±3.66 | 79.57±2.46 |
| **(13)** | 78.21±2.74 | 88.08±3.27 | 46.00±2.31 | 74.88±2.49 | 54.64±4.99 | 28.82±1.29 | 26.24±2.18 | 74.67±3.96 |
| **(14)** | 80.75±5.02 | 89.14±2.38 | 51.63±2.67 | 82.68±0.67 | 63.01±3.21 | 29.41±1.44 | 51.08±4.12 | 76.88±1.86 |
| **(15)** | 81.06±6.62 | 89.73±2.12 | 53.49±2.66 | 85.06±0.69 | 66.41±3.02 | 28.86±2.40 | 50.63±3.79 | 78.91±2.58 |

hypothesize that, in the context of these sparse benchmark graphs, the estimation variance may not be sufficiently influential on the model's outcomes.

Finally, Norm rescaling stands out as a significant enhancement in MPLP. This is particularly evident in its positive impact on datasets like Yeast, Physics, Photo, and Collab.

### F.3 Structural features ablation

We further examine the contribution of various structural features to the link prediction task. These features include: $\#(1,1)$, $\#(1,2)$, $\#(1,0)$, $\#(2,2)$, $\#(2,0)$, and $\#(\triangle)$. To ensure fair comparison, we utilize only the structural features for link representation, excluding the node representations derived from $\text{GNN}(\cdot)$. Given the combinatorial nature of these features, they are grouped into four categories:

Table 11: Ablation study of Batch Size ($B$) on non-attributed benchmarks evaluated by Hits@50. The format is average score ± standard deviation. The top three models are colored by **First**, **Second**, **Third**.

| | USAir | NS | PB | Yeast | C.ele | Power | Router | E.coli |
|---|---|---|---|---|---|---|---|---|
| **MPLP**($B = 256$) | **90.31**±1.32 | **88.98**±2.48 | **51.14**±2.44 | **84.07**±0.69 | **71.59**±2.83 | **28.92**±1.67 | **56.15**±3.80 | **85.12**±1.00 |
| **MPLP**($B = 512$) | **90.40**±2.47 | **89.40**±2.12 | 49.63±2.08 | **84.17**±0.60 | **71.72**±3.35 | **28.60**±1.66 | **53.25**±6.57 | 84.72±1.04 |
| **MPLP**($B = 1024$) | **90.49**±2.22 | **88.49**±2.34 | 50.60±3.40 | **83.67**±0.57 | **70.61**±4.13 | **28.63**±1.60 | 49.75±5.14 | 84.52±1.03 |
| **MPLP**($B = 2048$) | 81.20±2.80 | 61.79±18.55 | 50.34±3.05 | 76.79±6.79 | 31.79±19.88 | 28.45±1.88 | 49.37±3.89 | 84.43±1.28 |
| **MPLP**($B = 4096$) | 81.20±2.80 | 61.79±18.55 | **52.59**±2.36 | 58.26±7.20 | 31.54±18.53 | 27.25±3.30 | **50.26**±3.89 | **85.15**±1.15 |
| **MPLP**($B = 8192$) | 81.20±2.80 | 56.20±21.34 | **51.91**±2.08 | 24.47±21.12 | 31.79±19.88 | 17.22±3.17 | 38.67±7.78 | **85.67**±0.90 |

- $\#(1,1)$;
- $\#(1,2)$, $\#(1,0)$;
- $\#(2,2)$, $\#(2,0)$;
- $\#(\triangle)$.

The configuration of these structural features and their corresponding results are detailed in Table 9 and Table 10.

Our analysis reveals that distinct benchmark datasets have varied preferences for structural features, reflecting their unique underlying distributions. For example, datasets PB and Power exhibit superior performance with 2-hop structural features, whereas others predominantly favor 1-hop features. Although $\#(1,1)$, which counts Common Neighbors, is often considered pivotal for link prediction, the two other 1-hop structural features, $\#(1,2)$ and $\#(1,0)$, demonstrate a more pronounced impact on link prediction outcomes. Meanwhile, while the count of triangles, $\#(\triangle)$, possesses theoretical significance for model expressiveness, it seems less influential for link prediction when assessed in isolation. However, its presence can bolster link prediction performance when combined with other key structural features.

### F.4 Parameter sensitivity

We perform an ablation study to assess the hyperparameter sensitivity of MPLP, focusing specifically on two parameters: Batch Size ($B$) and Node Signature Dimension ($F$).

Our heightened attention to $B$ stems from its role during training. Within each batch, MPLP executes the shortcut removal. Ideally, if $B = 1$, only one target link would be removed, thereby preserving the local structures of other links. However, this approach is computationally inefficient. Although shortcut removal can markedly enhance performance and address the distribution shift issue (as elaborated in Appendix F.2), it can also inadvertently modify the graph structure. Thus, striking a balance between computational efficiency and minimal graph structure alteration is essential.

Our findings are delineated in Table 11, Table 12, Table 13, and Table 14. Concerning the batch size, our results indicate that opting for a smaller batch size typically benefits performance. However, if this size is increased past a certain benchmark threshold, there can be a noticeable performance drop. This underscores the importance of pinpointing an optimal batch size for MPLP. Regarding the node signature dimension, our data suggests that utilizing longer QO vectors consistently improves accuracy by reducing variance. This implies that, where resources allow, selecting a more substantial node signature dimension is consistently advantageous.

## G Theoretical analysis

### G.1 Proof for Theorem 3.1

We begin by restating Theorem 3.1 and then proceed with its proof:

Let $G = (V, E)$ be a non-attributed graph and consider a 1-layer GCN/SAGE. Define the input vectors $\boldsymbol{X} \in \mathbb{R}^{N \times F}$ initialized randomly from a zero-mean distribution with standard deviation $\sigma_{node}$. Additionally, let the weight matrix $\boldsymbol{W} \in \mathbb{R}^{F' \times F}$ be initialized from a zero-mean distribution with standard deviation $\sigma_{weight}$. After performing message passing, for any pair of nodes $\{(u, v) | (u, v) \in V \times V \setminus E\}$, the expected value of their inner product is given by:

Table 12: Ablation study of Batch Size ($B$) on attributed benchmarks evaluated by Hits@50. The format is average score ± standard deviation. The top three models are colored by **First**, **Second**, **Third**.

| | CS | Physics | Computers | Photo |
|---|---|---|---|---|
| **MPLP**($B = 256$) | 74.96±1.87 | **76.06±1.47** | **43.38±2.83** | **57.58±2.92** |
| **MPLP**($B = 512$) | **75.61±2.25** | **75.38±1.79** | **42.95±2.56** | **57.19±2.51** |
| **MPLP**($B = 1024$) | 74.89±2.00 | 74.89±1.97 | **42.69±2.41** | **56.97±3.20** |
| **MPLP**($B = 2048$) | 75.02±2.68 | **75.47±1.68** | 41.39±2.87 | 55.89±3.03 |
| **MPLP**($B = 4096$) | **75.46±1.78** | 74.88±2.57 | 40.65±2.85 | 55.89±2.88 |
| **MPLP**($B = 8192$) | **75.26±1.91** | 74.14±2.17 | 40.00±3.40 | 55.90±2.52 |

Table 13: Ablation study of Node Signature Dimension ($F$) on non-attributed benchmarks evaluated by Hits@50. The format is average score ± standard deviation. The top three models are colored by **First**, **Second**, **Third**.

| | USAir | NS | PB | Yeast | C.ele | Power | Router | E.coli |
|---|---|---|---|---|---|---|---|---|
| **MPLP**($F = 256$) | **90.64±2.50** | 88.52±3.07 | 50.42±3.86 | 80.63±0.84 | 70.89±4.70 | 25.74±1.59 | 51.84±2.90 | 84.60±0.92 |
| **MPLP**($F = 512$) | **90.49±1.95** | 89.18±2.35 | **51.48±2.63** | 82.41±1.10 | **70.91±4.68** | 27.58±1.80 | 51.98±4.38 | **84.70±1.33** |
| **MPLP**($F = 1024$) | **90.16±1.61** | **89.40±2.12** | 50.60±3.40 | **83.87±1.06** | 70.61±4.13 | **28.88±2.24** | **53.92±2.88** | **84.81±0.85** |
| **MPLP**($F = 2048$) | 90.14±2.24 | **89.36±1.92** | **51.26±1.67** | **84.20±1.02** | **72.24±3.31** | **29.27±1.92** | **54.50±4.52** | 84.58±1.42 |
| **MPLP**($F = 4096$) | 89.95±1.48 | **89.54±2.22** | **51.07±2.87** | **84.89±0.64** | **71.91±3.52** | **29.26±1.51** | **54.71±5.07** | **84.67±0.61** |

For GCN:

$$\mathbb{E}(\boldsymbol{h}_u \cdot \boldsymbol{h}_v) = \frac{C}{\sqrt{\hat{d}_u \hat{d}_v}} \sum_{k \in \mathcal{N}_u \bigcap \mathcal{N}_v} \frac{1}{\hat{d}_k},$$

For SAGE:

$$\mathbb{E}(\boldsymbol{h}_u \cdot \boldsymbol{h}_v) = \frac{C}{\sqrt{d_u d_v}} \sum_{k \in \mathcal{N}_u \bigcap \mathcal{N}_v} 1,$$

where $\hat{d}_v = d_v + 1$ and the constant $C$ is defined as $C = \sigma_{node}^2 \sigma_{weight}^2 FF'$.

*Proof.* Define $\boldsymbol{X}$ as $\left(\boldsymbol{X}_1^\top, \ldots, \boldsymbol{X}_N^\top\right)^\top$ and $\boldsymbol{W}$ as $(\boldsymbol{W}_1, \boldsymbol{W}_2, \ldots, \boldsymbol{W}_F)$.

Using GCN as the MPNN, the node representation is updated by:

$$\boldsymbol{h}_u = \boldsymbol{W} \sum_{k \in \mathcal{N}(u) \cup \{u\}} \frac{1}{\sqrt{\hat{d}_k \hat{d}_u}} \boldsymbol{X}_k,$$

where $\hat{d}_v = d_v + 1$.

Table 14: Ablation study of Node Signature Dimension ($F$) on attributed benchmarks evaluated by Hits@50. The format is average score ± standard deviation. The top three models are colored by **First**, **Second**, **Third**.

| | CS | Physics | Computers | Photo |
|---|---|---|---|---|
| **MPLP**($F = 256$) | 74.90±1.88 | 73.91±1.41 | 40.65±3.24 | 55.13±2.98 |
| **MPLP**($F = 512$) | 74.67±2.63 | 74.49±2.05 | 39.36±2.28 | **55.93±3.31** |
| **MPLP**($F = 1024$) | **75.02±2.68** | **75.27±2.95** | **42.27±3.96** | 55.89±3.03 |
| **MPLP**($F = 2048$) | **75.30±2.14** | **75.82±2.15** | **41.98±3.21** | **57.11±2.56** |
| **MPLP**($F = 4096$) | **76.04±1.57** | **76.17±2.04** | **43.33±2.93** | **58.55±2.47** |

For any two nodes $(u, v)$ from $\{(u, v)|(u, v) \in V \times V \setminus E\}$, we compute:

$$\begin{aligned}
\boldsymbol{h}_u \cdot \boldsymbol{h}_v &= \boldsymbol{h}_u^\top \boldsymbol{h}_v \\
&= \left( \boldsymbol{W} \sum_{a \in \mathcal{N}(u) \cup \{u\}} \frac{1}{\sqrt{\hat{d}_a \hat{d}_u}} \boldsymbol{X}_a \right)^\top \left( \boldsymbol{W} \sum_{b \in \mathcal{N}(v) \cup \{v\}} \frac{1}{\sqrt{\hat{d}_b \hat{d}_v}} \boldsymbol{X}_b \right) \\
&= \sum_{a \in \mathcal{N}(u) \cup \{u\}} \frac{1}{\sqrt{\hat{d}_a \hat{d}_u}} \boldsymbol{X}_a^\top \boldsymbol{W}^\top \boldsymbol{W} \sum_{b \in \mathcal{N}(v) \cup \{v\}} \frac{1}{\sqrt{\hat{d}_b \hat{d}_v}} \boldsymbol{X}_b \\
&= \sum_{a \in \mathcal{N}(u) \cup \{u\}} \frac{1}{\sqrt{\hat{d}_a \hat{d}_u}} \boldsymbol{X}_a^\top \begin{pmatrix} \boldsymbol{W}_1^\top \boldsymbol{W}_1 & \cdots & \boldsymbol{W}_1^\top \boldsymbol{W}_F \\ \vdots & \vdots & \vdots \\ \boldsymbol{W}_F^\top \boldsymbol{W}_1 & \cdots & \boldsymbol{W}_F^\top \boldsymbol{W}_F \end{pmatrix} \sum_{b \in \mathcal{N}(v) \cup \{v\}} \frac{1}{\sqrt{\hat{d}_b \hat{d}_v}} \boldsymbol{X}_b.
\end{aligned}$$

Given that

1. $\mathbb{E}\big(\boldsymbol{W}_i^\top \boldsymbol{W}_j\big) = \sigma_{weight}^2 F'$ when $i = j$,
2. $\mathbb{E}\big(\boldsymbol{W}_i^\top \boldsymbol{W}_j\big) = 0$ when $i \neq j$,

we obtain:

$$\mathbb{E}(\boldsymbol{h}_u \cdot \boldsymbol{h}_v) = \sigma_{weight}^2 F' \sum_{a \in \mathcal{N}(u) \cup \{u\}} \frac{1}{\sqrt{\hat{d}_a \hat{d}_u}} \boldsymbol{X}_a^\top \sum_{b \in \mathcal{N}(v) \cup \{v\}} \frac{1}{\sqrt{\hat{d}_b \hat{d}_v}} \boldsymbol{X}_b.$$

Also the orthogonal of the random vectors guarantee that $\mathbb{E}\big(\boldsymbol{X}_a^\top \boldsymbol{X}_b\big) = 0$ when $a \neq b$. Then, we have:

$$\mathbb{E}(\boldsymbol{h}_u \cdot \boldsymbol{h}_v) = \frac{C}{\sqrt{\hat{d}_u \hat{d}_v}} \sum_{k \in \mathcal{N}_u \bigcap \mathcal{N}_v} \frac{1}{\hat{d}_k}$$

where $C = \sigma_{node}^2 \sigma_{weight}^2 F F'$.

This completes the proof for the GCN variant. A similar approach, utilizing the probabilistic orthogonality of the input vectors and weight matrix, can be employed to derive the expected value for SAGE as the MPNN. □

### G.2 Proof for Theorem 3.2

We begin by restating Theorem 3.2 and then proceed with its proof:

Let $G = (V, E)$ be a graph, and let the vector dimension be given by $F \in \mathbb{N}_+$. Define the input vectors $\boldsymbol{X} = (X_{i,j})$, which are initialized from a random variable x having a mean of 0 and a standard deviation of $\frac{1}{\sqrt{F}}$. Using the message-passing as described by Equation 3, for any pair of nodes $\{(u, v)|(u, v) \in V \times V\}$, the expected value and variance of their inner product are:

$$\begin{aligned}
\mathbb{E}(\boldsymbol{h}_u \cdot \boldsymbol{h}_v) &= \text{CN}(u, v), \\
\text{Var}(\boldsymbol{h}_u \cdot \boldsymbol{h}_v) &= \frac{1}{F} \left( d_u d_v + \text{CN}(u, v)^2 - 2\text{CN}(u, v) \right) + F \text{Var}\big(\text{x}^2\big) \text{CN}(u, v).
\end{aligned}$$

*Proof.* We follow the proof of the theorem in [12]. Based on the message-passing defined in Equation 3:

$$\begin{aligned}
\mathbb{E}(\boldsymbol{h}_u \cdot \boldsymbol{h}_v) &= \mathbb{E}\left( \left( \sum_{k_u \in \mathcal{N}_u} \boldsymbol{X}_{k_u,:} \right) \cdot \left( \sum_{k_v \in \mathcal{N}_v} \boldsymbol{X}_{k_v,:} \right) \right) \\
&= \mathbb{E}\left( \sum_{k_u \in \mathcal{N}_u} \sum_{k_v \in \mathcal{N}_v} \boldsymbol{X}_{k_u,:} \boldsymbol{X}_{k_v,:} \right) \\
&= \sum_{k_u \in \mathcal{N}_u} \sum_{k_v \in \mathcal{N}_v} \mathbb{E}(\boldsymbol{X}_{k_u,:} \boldsymbol{X}_{k_v,:}).
\end{aligned}$$

Since the sampling of each dimension is independent of each other, we get:

$$\mathbb{E}(\boldsymbol{h}_u \cdot \boldsymbol{h}_v) = \sum_{k_u \in \mathcal{N}_u} \sum_{k_v \in \mathcal{N}_v} \sum_{i=1}^{F} \mathbb{E}(X_{k_u,i} X_{k_v,i}).$$

When $k_u = k_v$,

$$\mathbb{E}(X_{k_u,i} X_{k_v,i}) = \mathbb{E}\left(\mathrm{x}^2\right) = \frac{1}{F}.$$

When $k_u \neq k_v$,

$$\mathbb{E}(X_{k_u,i} X_{k_v,i}) = \mathbb{E}(X_{k_u,i})\mathbb{E}(X_{k_v,i}) = 0.$$

Thus:

$$\begin{aligned}\mathbb{E}(\boldsymbol{h}_u \cdot \boldsymbol{h}_v) &= \sum_{k_u \in \mathcal{N}_u} \sum_{k_v \in \mathcal{N}_v} \sum_{i=1}^{F} \mathbf{1}\left(k_u = k_v\right) \frac{1}{F} \\ &= \sum_{k \in \mathcal{N}_u \cap \mathcal{N}_v} 1 = \mathrm{CN}(u, v).\end{aligned}$$

For the variance, we separate the equal from the non-equal pairs of $k_u$ and $k_v$. Note that there is no covariance between the equal pairs and the non-equal pairs due to the independence:

$$\begin{aligned}\mathrm{Var}(\boldsymbol{h}_u \cdot \boldsymbol{h}_v) &= \mathrm{Var}\left(\sum_{k_u \in \mathcal{N}_u} \sum_{k_v \in \mathcal{N}_v} \sum_{i=1}^{F} X_{k_u,i} X_{k_v,i}\right) \\ &= \sum_{i=1}^{F} \mathrm{Var}\left(\sum_{k_u \in \mathcal{N}_u} \sum_{k_v \in \mathcal{N}_v} X_{k_u,i} X_{k_v,i}\right) \\ &= \sum_{i=1}^{F} \left(\mathrm{Var}\left(\sum_{k \in \mathcal{N}_u \cap \mathcal{N}_v} \mathrm{x}^2\right) + \mathrm{Var}\left(\sum_{k_u \in \mathcal{N}_u} \sum_{k_v \in \mathcal{N}_v \setminus \{k_u\}} X_{k_u,i} X_{k_v,i}\right)\right).\end{aligned}$$

For the first term, we can obtain:

$$\mathrm{Var}\left(\sum_{k \in \mathcal{N}_u \cap \mathcal{N}_v} \mathrm{x}^2\right) = \mathrm{Var}\left(\mathrm{x}^2\right)\mathrm{CN}(u, v).$$

For the second term, we further split the variance of linear combinations to the linear combinations of variances and covariances:

$$\begin{aligned}\mathrm{Var}\left(\sum_{k_u \in \mathcal{N}_u} \sum_{k_v \in \mathcal{N}_v \setminus \{k_u\}} X_{k_u,i} X_{k_v,i}\right) = &\sum_{k_u \in \mathcal{N}_u} \sum_{k_v \in \mathcal{N}_v \setminus \{k_u\}} \mathrm{Var}(X_{k_u,i} X_{k_v,i}) + \\ &\sum_{a \in \mathcal{N}_u \setminus \{k_u\}} \sum_{b \in \mathcal{N}_v \setminus \{k_v, a\}} \mathrm{Cov}(X_{k_u,i} X_{k_v,i}, X_{a,i} X_{b,i}).\end{aligned}$$

Note that the $\mathrm{Cov}(X_{k_u,i} X_{k_v,i}, X_{a,i} X_{b,i})$ is $\mathrm{Var}(X_{k_u,i} X_{k_v,i}) = \frac{1}{F^2}$ when $(k_u, k_v) = (b, a)$, and otherwise 0.

Thus, we have:

$$\mathrm{Var}\left(\sum_{k_u \in \mathcal{N}_u} \sum_{k_v \in \mathcal{N}_v \setminus \{k_u\}} X_{k_u,i} X_{k_v,i}\right) = \frac{1}{F^2}\left(d_u d_v + \mathrm{CN}(u, v)^2 - 2\mathrm{CN}(u, v)\right),$$

and the variance is:

$$\mathrm{Var}(\boldsymbol{h}_u \cdot \boldsymbol{h}_v) = \frac{1}{F}\left(d_u d_v + \mathrm{CN}(u, v)^2 - 2\mathrm{CN}(u, v)\right) + F\mathrm{Var}\left(\mathrm{x}^2\right)\mathrm{CN}(u, v).$$

□

### G.3 Proof for Theorem 3.3

We begin by restating Theorem 3.3 and then proceed with its proof:

Under the conditions defined in Theorem 3.2, let $\boldsymbol{h}_u^{(l)}$ denote the vector for node $u$ after the $l$-th message-passing iteration. We have:

$$\mathbb{E}\Big(\boldsymbol{h}_u^{(p)} \cdot \boldsymbol{h}_v^{(q)}\Big) = \sum_{k \in V} |\text{walks}^{(p)}(k,u)||\text{walks}^{(q)}(k,v)|,$$

where $|\text{walks}^{(l)}(u,v)|$ counts the number of length-$l$ walks between nodes $u$ and $v$.

*Proof.* Reinterpreting the message-passing described in Equation 3, we can equivalently express it as:

$$\text{ms}_v^{(l+1)} = \bigcup_{u \in \mathcal{N}_v} \text{ms}_u^{(l)}, \boldsymbol{h}_v^{(l+1)} = \sum_{u \in \text{ms}_v^{(l+1)}} \boldsymbol{h}_u^{(0)}, \tag{9}$$

where $\text{ms}_v^{(l)}$ refers to a multiset, a union of multisets from its neighbors. Initially, $\text{ms}_v^{(0)} = \{\{v\}\}$. The node vector $\boldsymbol{h}_v^{(l)}$ is derived by summing the initial QO vectors of the multiset's elements.

We proceed by induction: Base Case ($l = 1$):

$$\text{ms}_v^{(1)} = \bigcup_{u \in \mathcal{N}_v} \text{ms}_u^{(0)} = \bigcup_{u \in \mathcal{N}_v} \{\{u\}\} = \{\{k | \omega \in \text{walks}^{(1)}(k,v)\}\}$$

Inductive Step ($l \geq 1$): Let's assume that $\text{ms}_v^{(l)} = \{\{k | \omega \in \text{walks}^{(l)}(k,v)\}\}$ holds true for an arbitrary $l$. Utilizing Equation 9 and the inductive hypothesis, we deduce:

$$\text{ms}_v^{(l+1)} = \bigcup_{u \in \mathcal{N}_v} \{\{k | \omega \in \text{walks}^{(l)}(k,u)\}\}.$$

If $k$ initiates the $l$-length walks terminating at $v$ and if $v$ is adjacent to $u$, then $k$ must similarly initiate the $l$-length walks terminating at $u$. This consolidates our inductive premise.

With the induction established:

$$\mathbb{E}\Big(\boldsymbol{h}_u^{(p)} \cdot \boldsymbol{h}_v^{(q)}\Big) = \mathbb{E}\left(\sum_{k_u \in \text{ms}_u^{(p)}} \boldsymbol{h}_{k_u}^{(0)} \cdot \sum_{k_v \in \text{ms}_v^{(q)}} \boldsymbol{h}_{k_v}^{(0)}\right)$$

The inherent independence among node vectors concludes the proof. □

## H Limitations

Despite the promising capabilities of MPLP, there are distinct limitations that warrant attention:

1. Training cost vs. inference cost: The computational cost during training significantly outweighs that of inference. This arises from the necessity to remove shortcut edges for positive links in the training phase, causing the graph structure to change across different batches. This, in turn, mandates a repeated computation of the shortest-path neighborhood. Even though MPLP+ can avoid the computation of the shortest-path neighborhood for each batch, it shows suboptimal performance compared to MPLP. A potential remedy is to consider only a subset of links in the graph as positive instances and mask them, enabling a single round of preprocessing. Exploring this approach will be the focus of future work.
2. Estimation variance influenced by graph structure: The structure of the graph itself can magnify the variance of our estimations. Specifically, in dense graphs or those with a high concentration of hubs, the variance can become substantial, thereby compromising the accuracy of structural feature estimation.
3. Optimality of estimating structural features: Our research demonstrates the feasibility of using message-passing to derive structural features. However, its optimality remains undetermined. Message-passing, by nature, involves sparse matrix multiplication operations, which can pose challenges in terms of computational time and space, particularly for exceedingly large graphs.

# I Broader Impact

Our study is centered on creating a more efficient and expressive method for link prediction, with the goal of significantly advancing graph machine learning. The potential applications of our method are diverse and impactful, extending to recommendation systems, social network analysis, and biological interaction networks, among others. While we have not identified any inherent biases in our method, we acknowledge the necessity of rigorous bias assessments, particularly when integrating our method into industrial-scale applications.

## NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims made in abstract and introduction clearly reflect the contribution.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitation is discussed in Appendix H.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: The proof is rigorously shown in the Appendix.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The experimental details are extensively discussed in the main body and appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code and data are publicly available. The code is open source.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The experimental settings are discussed in the main body and the appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The error bars are clearly reported in the experimental results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The compute resource is discussed in Appendix D.4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Yes, the research conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Broader impact is discussed in Appendix I.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to

generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: This study is not feasible for safeguards.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: It is all properly cited.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: No new asset is introduced.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: No crowdsourcing experiments or research with human subjects are used in this study.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: No human participants in this study.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.