Pairwise Learning for Neural Link Prediction

Zhitao Wang
WeChat Pay, Tencent
zhitaowang@tencent.com

Yong Zhou
WeChat Search, Tencent
joyceyzhou@tencent.com

Litao Hong
WeChat Pay, Tencent
brianlthong@tencent.com

Yuanhang Zou
WeChat Search, Tencent
yuanhangzou@tencent.com

Hanjing Su
WeChat Pay, Tencent
justinsu@tencent.com

Shouzhi Chen
WeChat Pay, Tencent
easychen@tencent.com

Abstract

In this paper, we aim at providing an effective **P**airwise **L**earning for **N**eural **L**ink **P**rediction (PLNLP) framework. The framework treats link prediction as a pairwise learning to rank problem and consists of four main components, i.e., neighborhood encoder, link predictor, negative sampler and objective function. The framework is flexible that any generic graph neural convolutions or link prediction specific neural architectures could be employed as neighborhood encoder. For link predictor, we design different scoring functions, which could be selected based on different types of graphs. In negative sampler, we provide several sampling strategies, which are problem specific. As for objective function, we propose to use an effective ranking loss, which approximately maximizes the standard ranking metric AUC. We evaluate the proposed PLNLP framework on 4 link property prediction datasets of Open Graph Benchmark (OGB), including `ogbl-ddi`, `ogbl-collab`, `ogbl-ppa` and `ogbl-ciation2`. PLNLP achieves **top 1** performance on `ogbl-ddi` and `ogbl-collab`, and **top 2** performance on `ogbl-ciation2` only with basic neural architecture. The experimental results demonstrate the effectiveness of PLNLP.

1 Introduction

With a variety of real-world applications, link prediction has been recognized of great importance and attracted increasing attention from the research community in past decade [Lü and Zhou, 2011, Martínez et al., 2017]. For instance, link prediction methods could help infer potential protein-protein interactions to efficiently save human effort on blind checking [Airoldi et al., 2008]. Also, link prediction techniques could be used to predict new friendships between uses on social media, or to discover potential user-to-item relationships on E-commerce sites, such that user experience could be improved [Adamic and Adar, 2003, Koren et al., 2009].

In the literature, heuristic-based methods are probably the most representative link prediction algorithms. The key idea of most existing heuristic methods is to measure the similarity of two target nodes based on their neighborhood information. The success of these heuristics has demonstrated the importance of neighborhood information of target node-pair. However, heuristic methods often have weak applicability and expressiveness in dealing with different types of networks for its simple-form and hand-crafted information of neighborhood. A previous survey found that all of heuristics methods failed to perform consistently across multiple networks [Lü and Zhou, 2011]. The needs of prior

Preprint.

knowledge or expensive trial and error are inevitable in choosing appropriate heuristics for different networks. Thanks to effective feature learning ability of neural networks, a series of neural link prediction models [Zhang and Chen, 2017, Kipf and Welling, 2016b, Wang et al., 2020, 2019, 2021] were proposed, of which the generalization ability was successfully improved.

Existing neural link prediction methods pay much attention on designing more expressive neural architectures, while some basic properties of the problem are often neglected. For example, most neural models treat link prediction as a binary classification problem and naturally adopt a cross entropy loss function. However, this learning schema seems not to be suitable for the link prediction problem. First, link classification is extremely imbalanced due to the natural sparsity of most graphs. Although under-sampling could be adopted, there would be information loss during sampling process and what ratio of sampling is hard to decide. Second, most link prediction evaluation protocols do not aim at labeling positive pairs as 1 while negative pairs as 0, but ask for ranking positive pairs higher than negative pairs. Therefore, employing cross-entropy function seems not to be so direct to the objective of the link prediction task.

Based on above understanding and our previous research [Wang et al., 2020, 2019, 2021], we provide an effective and generic pairwise learning neural link prediction framework in this paper, named **PLNLP**. The framework adopts a pairwise learning to rank schema and consists of four main components, i.e., neighborhood encoder, link predictor, negative sampler and objective function. The neighborhood encoder aims at extracting expressive neighborhood information of input node-pair. Any generic graph neural convolution, such as GCN [Kipf and Welling, 2016a] and SAGE [Hamilton et al., 2017], or link prediction specific neural architecture, such as SEAL [Zhang and Chen, 2018], NANs [Wang et al., 2020] and HalpNet [Wang et al., 2021], could be employed as neighborhood encoder. For link predictor, we design different scoring functions, which could be selected based on different types of graphs. In negative sampler, we provide several negative sampling strategies, which are problem specific. As for objective function, we propose to use an effective ranking loss, which approximately maximizes the standard ranking metric AUC. We evaluate the proposed PLNLP framework on 4 link property prediction datasets of Open Graph Benchmark (OGB) [Hu et al., 2020], including `ogbl-ddi`, `ogbl-collab`, `ogbl-ppa` and `ogbl-ciation2`. PLNLP with basic neural architecture achieves **top 1** performance on `ogbl-ddi` and `ogbl-collab`, and **top 2** performance on `ogbl-ciation2`. The performance demonstrates the effectiveness of PLNLP.

2 Related Work

Existing link prediction approaches can be categorized into three families: heuristic feature based, latent embedding based and neural network based.

Heuristic Methods: Most heuristics measure node similarity with neighborhood information. Popular heuristics include first-order methods common neighbors, Jaccard index [Salton and McGill, 1986] and preferential attachment [Liben-Nowell and Kleinberg, 2007]; second-order methods, i.e., Adamic-Adar [Adamic and Adar, 2003], resource allocation [Zhou et al., 2009]; and high-order heuristic SimRank [Jeh and Widom, 2002]. These heuristics often fail to capture complex latent formation features.

Embedding-based Methods: Embedding based methods aim at learning latent node features. The most classical one is matrix factorization (MF) method [Menon and Elkan, 2011], which aims at reconstructing adjacency matrix. Besides, a series of unsupervised network representation learning models [Perozzi et al., 2014, Tang et al., 2015, Grover and Leskovec, 2016, Hamilton et al., 2017], are also applicable for link prediction. These methods learn generic latent embeddings by preserving structure proximities from a probabilistic view and predict links by composing node embeddings as edge features. PNRL [Wang et al., 2017] is a state-of-the-art link prediction specific embedding method, which simultaneously preserves proximities of observed structure and infers hidden links.

NN-based Methods: Recently, some neural network-based link prediction models were developed, which explore non-linear deep structural features with neural layers. Variational graph auto-encoders [Kipf and Welling, 2016b] predict links by encoding graph with graph convolutional layer [Kipf and Welling, 2016a]. Another two state-of-the-art neural models WLNM [Zhang and Chen, 2017] and SEAL [Zhang and Chen, 2018] use graph labeling algorithm to transfer union neighborhood of two nodes (enclosing subgraph) as meaningful matrix and employ convolutional neural layer or a novel graph neural layer DGCNN [Zhang et al., 2018] for encoding.

Besides, in our previous work, we proposed a series of neighborhood attention neural networks [Wang et al., 2020, 2019, 2021], in which different attention mechanisms were designed to encode neighborhood information specific for link prediction problem. For instance, in [Wang et al., 2020, 2019], we proposed cross neighborhood attention and interactive attention mechanisms to capture structural interactions between neighborhoods of the target node-pair.

3 Preliminaries

3.1 Graphs

Generally, a graph (network) is represented as $G = (V, E)$, where $V = \{v_1, ..., v_N\}$ is the set of nodes, $E \subseteq V \times V$ is the set of links, and the total number of distinct nodes is N. Also, a graph is often denoted as an adjacency matrix $\mathbf{A}$, where $A_{i,j} = 1$ if there is a link from node v_i to v_j, otherwise $A_{i,j} = 0$. $\mathbf{A}$ will be symmetric, if the graph is undirected.

3.2 Neighborhood of Node

We use $\mathcal{N}^h(v_i)$ to represent the h-hop neighborhood of node $v_i \in V$, which is the set of nodes whose distance to v_i (represented as $d(v_i, v_j)$) is not greater than h. In this paper, we focus on unweighted graph, thus the distance function $d(v_i, v_j)$ is directly computed as the length of the shortest path between v_i and v_j. We call v_i the center node and $v_j \in \mathcal{N}^h(v_i)$ the neighboring node within h-hop. To make the neighborhood also include the unique information of the center node, we define that the center node v_i is a neighboring node of itself, such that $v_i \in \mathcal{N}^h(v_i)$.

3.3 Neighborhood Subgraph of Node-Pair

We use $\mathcal{G}^h(v_i, v_j)$ to represent the h-hop neighborhood subgraph of the node pair (v_i, v_j), which is extracted from the whole graph $\mathcal{G}$. Formally, for any node v_k in the neighborhood subgraph $\mathcal{G}^h(v_i, v_j)$, it should satisfy $d(v_k, v_i) \leq h$ and $d(v_k, v_j) \leq h$, i.e., $v_k \in \mathcal{N}^h(v_i) \cup \mathcal{N}^h(v_j)$.

3.4 Link Prediction

Link prediction problems are categorized as temporal link prediction which predicts potential new links on an evolving network, and structural link prediction which infers missing links on a static network. In this paper, we focus on *structural link prediction*. Given the partially observed structure of a network, the goal of it is to predict the unobserved links. Formally, given a partially observed network $G = (V, E)$, we represent the set of node-pairs with unknown link status as $E^? = V \times V - E$, then the goal of structural link prediction is to infer link status of node-pairs in $E^?$.

4 PLNLP Framework

The proposed framework is illustrated as Figure 1. Given an input graph, negative sampler aims to draw negative samples and form training pairs. A training pair consists of a positive sample, which is a node-pair with an observed edge in input graph, and a negative sample, which is a node-pair drawn by negative sampler. Neighborhood encoder is used to extract neighborhood information of both positive and negative samples as the hidden representations. Given the hidden representations, link predictor will calculate link scores of both samples. With link scores of training pairs, the model parameters will be optimized based on the pairwise ranking objective function.

4.1 Neighborhood Encoder

Neighborhood information has proved crucial for link prediction. Therefore, we propose to use neighborhood neural encoder to extract structural information of input samples. We consider two kinds of neighborhood encoder in this paper. One is Node Neighborhood Encoder (NNE), which encodes the two nodes of a input sample with their own neighborhood as two hidden representations, separately. Any generic graph neural networks (GNN), e.g, GCN, GraphSAGE and GAT, could be employed as NNE. Assume that input sample is (v_i, v_j), NNEs aim to extract hidden representations

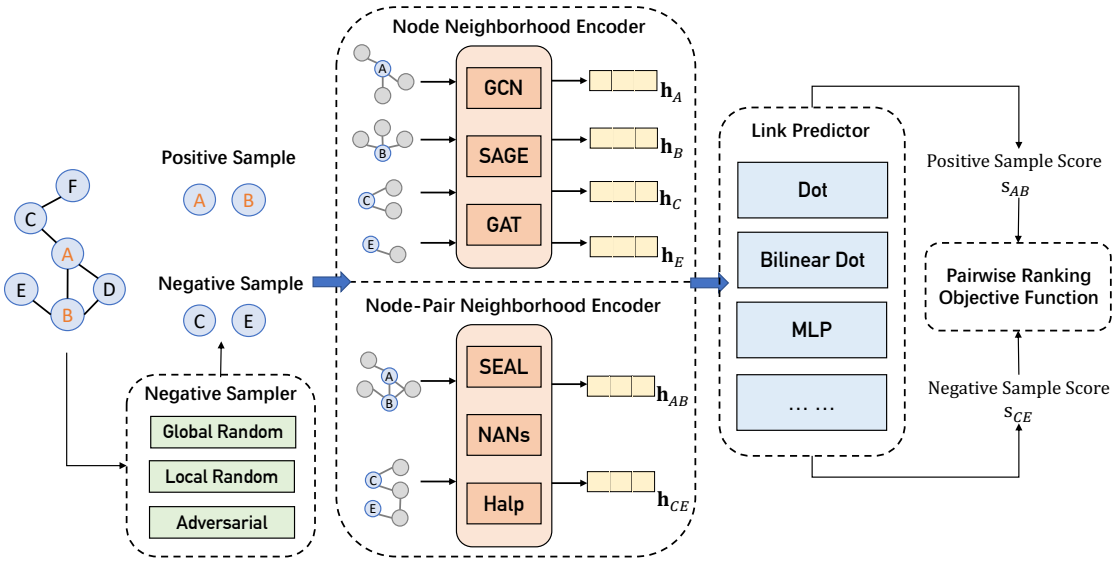

Figure 1: PLNLP Framework

of the input sample as follows:

$$\mathbf{h}_i = \text{NNE}(\mathbf{x}_i, \{\mathbf{x}_k \,|v_k \in \mathcal{N}^h(v_i)\}), \quad \mathbf{h}_j = \text{NNE}(\mathbf{x}_j, \{\mathbf{x}_l \,|v_l \in \mathcal{N}^h(v_j)\}) \tag{1}$$

where $\mathbf{x}_i$ generally represents the input feature of node v_i. If there is no input features, $\mathbf{x}_i$ represents a embedding vector of node v_i, which is trainable parameter. $\mathbf{x}_i$ could also represent the concatenation of input feature and node embedding. In this framework, we only consider homogeneous graph, which means that all nodes share a same NNE.

The other kind of neighborhood encoder is Edge level Neighborhood Encoder (ENE), or called node-pair neighborhood encoder. Recently, a series of ENEs, such as SEAL, NIAN and HalpNet, were proposed specifically for link prediction problem. The main advantage of ENEs is capturing structural interactions between the neighborhoods, which are ignored in NENs. ENEs often consider the neighborhood subgraph of a sample (node-pair) as input, and encode it as one hidden representation. Assume that input sample is (v_i, v_j), ENEs derive a hidden representation of the input sample as follows:

$$\mathbf{h}_{ij} = \text{ENE}(\mathbf{x}_i, \mathbf{x}_j, \{\mathbf{x}_k \,|v_k \in \mathcal{G}^h(v_i, v_j)\}) \tag{2}$$

Similarly, $\mathbf{x}_i$ represents the input feature, or trainable embedding, or the concatenation of input feature and embedding.

4.2 Link Score Predictor

After deriving the hidden representations either in node level or node-pair (edge) level, the framework will calculate a linking score of the input sample. We provide several selections of the scoring function.

Dot Predictor. If we use NNE to derive $\mathbf{h}_i$ and $\mathbf{h}_j$ of the input sample (v_i, v_j), we can simply use a dot operator to derive the score:

$$s_{ij} = \mathbf{h}_i \cdot \mathbf{h}_j \tag{3}$$

Bilinear Dot Predictor. Dot operator can be only used for undirected graph due to its commutative property. For directed graph, we can adopt bilinear dot operator to make the scoring function not commutative:

$$s_{ij} = \mathbf{h}_i \mathbf{W} \mathbf{h}_j \tag{4}$$

where $\mathbf{W}$ is a learnable matrix.

MLP Predictor. We can also employ a multi-layer perceptron (MLP) as the link predictor. If we use ENEs to obtain hidden representation of the input sample, the predictor is as follow:

$$s_{ij} = \text{MLP}(\mathbf{h}_{ij}) \tag{5}$$

If we use NNEs to obtain hidden representations, there are several possible forms of MLP's input. If the graph is undirected, we can adopt a widely used commutative operator, i.e, hadamard product $\odot$:

$$s_{ij} = \text{MLP}(\mathbf{h}_i \odot \mathbf{h}_j) \tag{6}$$

If the graph is directed, we would prefer a non-commutative operator, such as concatenation $||$:

$$s_{ij} = \text{MLP}(\mathbf{h}_i || \mathbf{h}_j) \tag{7}$$

4.3 Pairwise Learning with Ranking Objective

Due to the sparsity of networks, there often exists extreme imbalance between linked pairs and non-linked pairs. Meanwhile, most link prediction tasks do not aim at labeling positive pairs as 1 while negative pairs as 0, but ask for ranking positive pairs higher than negative pairs. To be consistent with the general objective of link prediction, we adopt the ranking idea for model learning, which can be formalized as:

$$s_{ij} > s_{kl}, \forall (v_i, v_j) \in E \text{ and } \forall (v_k, v_l) \in E^- \tag{8}$$

where s_{ij} and s_{kl} are the output scores of link predictor, E^- is the set of true non-linked pairs. In fact, the above learning objective is equivalent to maximize the Area Under the Curve (AUC), which is interpreted as the probability of a positive sample ranking higher than a negative sample. The empirical AUC value is defined as follow:

$$\text{AUC} = \sum_{(v_i,v_j)\in E} \sum_{(v_k,v_l)\in E^-} \frac{\mathbb{1}[f_\theta(v_i, v_j) > f_\theta(v_k, v_l)]}{|V \times V|} \tag{9}$$

where $\mathbb{1}[\cdot]$ is an indicator function that equals to 1 if $f_\theta(v_i, v_j) > f_\theta(v_k, v_l)$, otherwise equals to 0. $f_\theta(v_i, v_j) = s_{ij}$ represents the output of the neural link prediction model, where θ denotes all parameters of the model. Optimizing AUC is not straightforward since the gradient of this function is either zero or not defined. Various techniques have been proposed to approximate the AUC with a surrogate function. There are several possible selections of surrogate functions, such as pairwise hinge loss, logistic loss or exponential loss. In this paper, we simply select the squared least surrogate loss, which is proved consistent with AUC theoretically [Gao and Zhou, 2015]. Our framework is flexible to adopt any other surrogate function that approximates AUC. The base AUC-optimization objective function is defined as follow:

$$O_{\text{AUC}} = \min_\theta \sum_{(v_i,v_j)\in E,(v_i,v_k)\in E^-} \left(1 - f_\theta\left(v_i, v_j\right) + f_\theta\left(v_i, v_k\right)\right)^2 + \frac{\lambda}{2}||\theta||^2 \tag{10}$$

The above function forces the margin between positive samples and negative samples to be 1. In some situations, this constraint is too strict for the optimization. It can be relaxed by combining the squared hinge loss with above function:

$$O_{\text{Hinge-AUC}} = \min_\theta \sum_{(v_i,v_j)\in E,(v_i,v_k)\in E^-} \left(\max\left(0, 1 - f_\theta(v_i, v_j) + f_\theta(v_i, v_k)\right)\right)^2 + \frac{\lambda}{2}||\theta||^2 \tag{11}$$

The above function only forces the margin between positive samples and negative samples to be larger than 1.

Furthermore, the margin may not be fixed as 1 if weights on training edges (positive sample) are expected to be modeled. A straightforward way of introducing sample weights is as follows:

$$O_{\text{Weighted-Hinge-AUC}} = \min_\theta \sum_{(v_i,v_j)\in E,(v_i,v_k)\in E^-} \gamma_{ij} \left(\max\left(0, \gamma_{ij} - f_\theta(v_i, v_j) + f_\theta(v_i, v_k)\right)\right)^2 + \frac{\lambda}{2}||\theta||^2 \tag{12}$$

where γ is an adaptive margin, which may correspond to normalized weights of training edges.

In above objective functions, to prevent over-fitting problem, we use the L2 regularization on parameters with a weight λ. Given a positive pair (v_i, v_j) and a sampled negative pair (v_k, v_l), the parameters θ of the model are optimized by the stochastic gradient descent (SGD) method. For most cases, we use the basic objective function in Eq. 10. When sample weights are considered, the objective function of Eq. 12 is used.

4.4 Negative Sampling

In practice, true non-linked set E^- is not available in the training data. A conventional strategy is randomly sampling a negative node-pair (v_k, v_j), which has unknown link status and is assumed as negative samples. For different problems or types of graphs, we may have different sampling strategies.

Global Sampling. Global sampling represents that, for each positive sample, we uniformly sample a negative node-pair from the set $E^? = V \times V - E$. This strategy is suitable for the problem seeking for global ranking performance. For example, in the protein-protein interaction, we are interested in potential node-pairs, which are worth performing further analysis on, among all possible node-pairs.

Local Sampling. Local sampling represents that, for a positive sample (v_i, v_j), we firstly select an anchor node saying v_i, then uniformly sample a node v_k and regard (v_i, v_k) as the negative sample. Instead of uniform distribution, other distribution, e.g, the power of node degrees, can be applied to sample the negative node v_k. This strategy is appropriate to the situation that aims to obtain good ranking for individual nodes. For example, in a recommendation system, we would like to recommend a good ranking list of items to each individual user.

Adversarial Sampling. The performance of random sampling strategy is not always stable due to complete randomness. Similar problems of random negative sampling have also been found in other tasks, e.g., knowledge graph embedding [Wang et al., 2018, Cai and Wang, 2017] and image retrieval [Wu et al., 2017]. In our previous work [Wang et al., 2020], we proposed to use adversarial learning technique to generate negative samples instead of random sampling. We designed a generative model to generate high quality negative samples, which aims at making difficulties to link prediction model. In this way, link prediction model and negative sample generator play an adversarial game. By continuously providing high quality negative samples, adversarial sampling more robust than random sampling. We leave the evaluation of adversarial sampling on ogb datasets as future work.

Negative Sample Sharing. Since the framework adopts pairwise schema, each negative sample can only be used for one positive sample once, which is not efficient. To make better use of negative samples, we propose a negative sample sharing mechanism. As shown in Figure 2, assume that the total number of positive sample is m, we firstly draw m negative samples and construct m training pairs with same indexes. Given the negative samples, the sharing mechanism will random permute the indexes of negative samples, and form m new training pairs. The hyper-parameter `num_neg` indicates the mechanism will random permute `(num_neg-1)` times of negative samples. By using this sharing mechanism, we could create $m \times$ `num_neg` training pairs by only sampling m negative samples at each training epoch.

4.5 Data Augmentation with Random Walk

In some graphs, high-order structure information play a important role. Although increasing the number of GNN layers could model the high-order information, it also may leads to over-smoothing problem and low efficiency. To this end, we propose to use data augmentation to introduce high-order information at the input. A general technique to sample high order information is the Random Walk. Given all nodes in the graph, we use the basic random walk method to sample the high-order pairs. Assume the start point node is v_i and its random walk is $\text{RW}(v_i) = \{v_{k+1}, ..., v_{k+l}\}$, where l represents the walk length , then the set of positive samples is augmented as :

$$E_{\text{aug}} = E \cup \{(v_i, v_j) | v_j \in \text{RW}(v_i), \forall v_i \in V\} \tag{13}$$

Meanwhile, the augmented pairs are associated with weights based on the steps of walks. For example, in the walk, $\text{RW}(v_i) = \{v_{k+1}, ..., v_{k+l}\}$, the augmented pair (v_i, v_{k+l}) is associated with the weight $1/l$. With different weights of augmented pairs in E_{aug}, we find that using the weight-adaptive objective function in Eq. 12 is more effective. Therefore, it is suggested using this objective function when random walk augmentation is adopted.

5 Evaluation on OGB

Our code for evaluation is available at `https://github.com/zhitao-wang/PLNLP`.

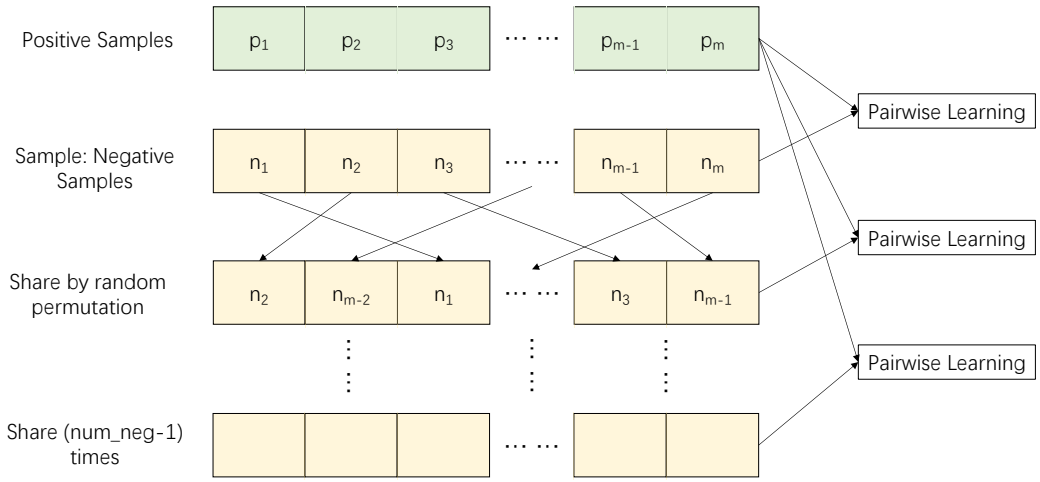

Figure 2: Negative Sample Sharing Mechanism

5.1 Datasets and Evaluation Metrics

We evaluate the link prediction ability of PLNLP on Open Graph Benchmark (OGB) data [Hu et al., 2020]. Four data sets with different graph types are evaluated, including ogbl-ddi, ogbl-collab, ogbl-citation2 and ogbl-ppa.

ogbl-ddi: The dataset is a homogeneous, unweighted, undirected graph, representing the drug-drug interaction network. Each node represents a drug. Edges represent interactions between drugs.

The task is to predict drug-drug interactions given information on already known drug-drug interactions. The performance is evaluated by Hits@20: each true drug interaction is ranked among a set of approximately 100,000 randomly-sampled negative drug interactions, and count the ratio of positive edges that are ranked at 20-place or above.

ogbl-collab: The dataset is an undirected graph, representing a subset of the collaboration network between authors indexed by MAG. Each node represents an author and edges indicate the collaboration between authors. All nodes come with 128-dimensional features.

The task is to predict the future author collaboration relationships given the past collaborations. Evaluation metric is Hits@50, where each true collaboration is ranked among a set of 100,000 randomly-sampled negative collaborations.

ogbl-ppa: The dataset is an undirected, unweighted graph. Nodes represent proteins from 58 different species, and edges indicate biologically meaningful associations between proteins.

The task is to predict new association edges given the training edges. Evaluation metric is Hits@100, where each positive edge is ranked among 3,000,000 randomly-sampled negative edges.

ogbl-citation2: The dataset is a directed graph, representing the citation network between a subset of papers extracted from MAG. Each node is a paper with 128-dimensional word2vec features.

The task is to predict missing citations given existing citations. The evaluation metric is Mean Reciprocal Rank (MRR), where the reciprocal rank of the true reference among 1,000 negative candidates is calculated for each source paper, and then the average is taken over all source papers.

Table 1: Settings of PLNLP on OGB Datasets

	Loss Func.	Encoder	Predictor	Neg. Sampler	Other Parameters
ddi	O_{AUC}	SAGE layer = 2 dim = 512 dropout = 0.3	MLP layer = 2 dim = 512 dropout = 0.3	GLOBAL num_neg = 3	node emb. = 512 node feat. = no lr = 0.001 epochs = 500
collab	$O_{\text{Weighted-Hinge-AUC}}$	SAGE layer = 1 dim = 256 dropout = 0.3	DOT	GLOBAL num_neg = 1	node emb. = 256 node feat. = no lr = 0.001 epochs = 800 random walk aug. = yes walk length = 10
ppa	O_{AUC}	SAGE layer = 2 dim = 256 dropout = 0.3	DOT	GLOBAL num_neg = 3	node emb. = 256 node feat. = yes lr = 0.001 epochs = 200
citation2	O_{AUC}	GCN layer = 2 dim = 200 dropout = 0.0	MLP layer = 2 dim = 200 dropout = 0.0	LOCAL num_neg = 3	node emb. = 50 node feat. = yes lr = 0.001 epochs = 100

5.2 Evaluation Settings and Results

The detailed settings of PLNLP in this paper are shown in Table 1. We only employ basic node level neighborhood encoder, e.g., GCN or SAGE, to demonstrate the effectiveness of the proposed framework. Some well-design edge level neighborhood encoder specific for link prediction, such as SEAL, NANs and HalpNet, may further improve the performance. But due to low efficiency of edge level neighborhood encoders, we leave this part in the future work. We treat all training datasets as unweighted and undirected graphs. As for MLP predictor, we use hadamard product to get the input of MLP. It is worth noting that we use validation set for training on `ogbl-collab`, which is allowed by OGB. Meanwhile, we employ the trick from HOP-REC that we only use training edges after year 2010 in `ogbl-collab`. Furthermore, we use random walk augmentation with a walk length 10 for `ogbl-collab`.

Following OGB rules, we evaluate PLNLP with 10 runs, without fixing random seed. As for other state-of-the-art methods, we just copy the results from OGB official leader board.

The averaged results with standard deviation are reported in the Table 2. Only with basic graph neural architectures, PLNLP achieves top 1 performance on `ogbl-ddi` and `ogbl-collab`, and top 2 performance on `ogbl-ciation2`. This significantly demonstrates the effectiveness of PLNLP.

5.3 Ablation Study

Furthermore, we compare PLNLP against the generic classification learning framework where the loss function is cross-entropy. In this ablation study, we keep same neural architecture (same encoder, predictor with same parameters as reported in Table 1) in the two frameworks. We use basic AUC objective function Eq.10 for all datasets and do not use random walk augmentation for `ogbl-collab` in this study. To guarantee fairness, we use same negative sampling strategies and use the same number of negative samples at each epoch. The results are shown in Table 3. It is found that PLNLP remarkably outperforms the generic classification learning schema, which indicates the proposed pairwise learning could maximize the performance of graph neural models on link prediction problem.

Acknowledgments

The authors greatly thank the great support for advanced research from departments of WeChat Pay and WeChat Search.

Table 2: Link Prediction Performance on OGB (Test Performance)

	ogbl-ddi Hits@20(%)	ogbl-collab Hits@50(%)	ogbl-ppa Hits@100(%)	ogbl-citation2 MRR(%)
CN	17.73 ± 0.00	61.37 ± 0.00	27.65 ± 0.00	51.47 ± 0.00
AA	18.61 ± 0.00	64.17 ± 0.00	32.45 ± 0.00	51.89 ± 0.00
RA	–	–	49.33 ± 0.00	–
AA+Proposal Set	–	65.48 ± 0.00	–	–
RA+Proposal Set	–	–	$\mathbf{53.24 \pm 0.00}$	–
MF	13.68 ± 4.75	38.86 ± 0.29	32.29 ± 0.94	51.86 ± 4.43
DeepWalk	22.46 ± 2.90	50.37 ± 0.34	23.02 ± 1.63	–
Node2vec	23.26 ± 2.09	48.88 ± 0.54	22.26 ± 0.83	–
HOP-REC	–	70.12 ± 0.16	–	–
SAGE	53.90 ± 4.74	54.63 ± 1.12	16.55 ± 2.40	82.60 ± 0.36
GCN	37.07 ± 5.07	47.14 ± 1.45	18.67 ± 1.32	84.74 ± 0.31
SEAL	30.56 ± 3.86	64.74 ± 0.33	48.80 ± 3.16	$\mathbf{87.67 \pm 0.32}$
SAGE+Proposal Set	74.95 ± 3.17	–	–	–
CFLP (w/ JKNet)	86.08 ± 1.98	–	–	–
SAGE+Edge Attr	87.81 ± 4.47	–	–	–
PLNLP	$\mathbf{90.88 \pm 3.13}$	$\mathbf{70.59 \pm 0.29}$	32.38 ± 2.58	$\mathbf{84.92 \pm 0.29}$

Table 3: PLNLP vs. Classification Schema

	ogbl-ddi Hits@20(%)		ogbl-collab Hits@50(%)		ogbl-ppa Hits@100(%)		ogbl-citation2 MRR(%)	
	Test	Valid	Test	Valid	Test	Valid	Test	Valid
Classification	70.70	68.02	64.97	99.17	16.55	17.24	84.64	84.73
PLNLP	**90.88**	**82.42**	**68.72**	**100.00**	**32.38**	**31.62**	**84.92**	**84.90**

References

Lada A Adamic and Eytan Adar. Friends and neighbors on the web. *Social networks*, 25(3):211–230, 2003.

Edoardo M Airoldi, David M Blei, Stephen E Fienberg, and Eric P Xing. Mixed membership stochastic blockmodels. *Journal of Machine Learning Research*, 9(Sep):1981–2014, 2008.

Liwei Cai and William Yang Wang. Kbgan: Adversarial learning for knowledge graph embeddings. *arXiv preprint arXiv:1711.04071*, 2017.

Wei Gao and Zhi-Hua Zhou. On the consistency of auc pairwise optimization. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864. ACM, 2016.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pages 1024–1034, 2017.

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 preprint arXiv:2005.00687*, 2020.

Glen Jeh and Jennifer Widom. Simrank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 538–543. ACM, 2002.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016a.

Thomas N Kipf and Max Welling. Variational graph auto-encoders. *NIPS Workshop on Bayesian Deep Learning*, 2016b.

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

David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.

Linyuan Lü and Tao Zhou. Link prediction in complex networks: A survey. *Physica A: statistical mechanics and its applications*, 390(6):1150–1170, 2011.

Víctor Martínez, Fernando Berzal, and Juan-Carlos Cubero. A survey of link prediction in complex networks. *ACM Computing Surveys (CSUR)*, 49(4):69, 2017.

Aditya Krishna Menon and Charles Elkan. Link prediction via matrix factorization. In *Joint european conference on machine learning and knowledge discovery in databases*, pages 437–452. Springer, 2011.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014.

Gerard Salton and Michael J McGill. Introduction to modern information retrieval. 1986.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077. International World Wide Web Conferences Steering Committee, 2015.

PeiFeng Wang, Shuangyin Li, and Rong Pan. Incorporating gan for negative sampling in knowledge representation learning. In *AAAI*, 2018.

Zhitao Wang, Chengyao Chen, and Wenjie Li. Predictive network representation learning for link prediction. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 969–972. ACM, 2017.

Zhitao Wang, Yu Lei, and Wenjie Li. Neighborhood interaction attention network for link prediction. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2153–2156, 2019.

Zhitao Wang, Yu Lei, and Wenjie Li. Neighborhood attention networks with adversarial learning for link prediction. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

Zhitao Wang, Wenjie Li, and Hanjing Su. Hierarchical attention link prediction neural network. *Knowledge-Based Systems*, 232:107431, 2021. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2021.107431. URL `https://www.sciencedirect.com/science/article/pii/S0950705121006936`.

Chao-Yuan Wu, R Manmatha, Alexander J Smola, and Philipp Krahenbuhl. Sampling matters in deep embedding learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2840–2848, 2017.

Muhan Zhang and Yixin Chen. Weisfeiler-lehman neural machine for link prediction. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 575–583. ACM, 2017.

Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems*, pages 5165–5175, 2018.

Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

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