Table 6: Statistics of heterogeneous graph datasets.

Data	# User	# Item	# Edges	Split ratio
Gowalla	29, 858	40, 981	1, 027, 370	70/10/20
Yelp2018	31, 668	38, 048	1, 561, 406	70/10/20
Amazon-Book	52, 643	91, 599	2, 984, 108	70/10/20

Table 5: Statistics of homogeneous graph datasets.

Data	# Nodes	# Edges	# Features	Split ratio
Cora	2, 708	5, 429	1, 433	85/5/15
CiteSeer	3, 312	4, 660	3, 703	85/5/15
PubMed	19, 717	44, 338	500	85/5/15
ogbl-ddi	4, 267	1, 334, 889	-	80/10/10
ogbl-collab	235, 868	1, 285, 465	128	92/4/4
ogbl-ppa	576, 289	30, 326, 273	58	70/20/10

A DATASET DETAILS

In this section, we introduce the details of applied datasets as bellow.

- **Cora, CiteSeer**, and **PubMed**: They are the most popular benchmark citation networks used in graph domain. Nodes correspond to documents and edges correspond to citations. Each node has a bag-of-words feature vector according to the paper abstract. Labels are defined as the academic topics.
- **ogbl-ddi**: This is a drug-drug interaction network. Each node represents an FDA-approved or experimental drug. Edges represent interactions between drugs. Node features are not available, in experiments, following [15], we randomly initialize 256-dimensional embedding vector for each node.
- **ogbl-collab**: This is a challenging author collaboration network from KDD Cup 2021. Each node is an author and edges indicate the collaboration between authors. All nodes come with 128-dimensional features, obtained by averaging the word embeddings of papers that are published by the authors.
- **ogbl-ppa**: This is a protein-protein association network. Nodes represent proteins from 58 different species, and edges indicate biologically meaningful associations between proteins. In experiments, we 58-dimensional one hot vectors as node features.

In addition to the aforementioned six homogeneous graphs, we also consider three popular recommendation datasts.

- **Gowalla**: This is the check-in dataset obtained from Gowalla, where users share their locations by checking-in. To ensure the qualify of the dataset, following [35], we use the 10-core setting, i.e., retaining users and items with at least ten interactions.
- **Yelp2018**: This dataset is adopted from the 2018 edition of the Yelp challenge. It describes the relationships between customers and items like restaurants and bars. We use the same 10-core setting in order to ensure data quality.
- **Amazon-book**[1]: is one of the widely used dataset for product recommendation. Similarly, we use the 10-core setting to ensure that each user and item have at least ten interactions.

We split all datasets above into the training/validation/testing sets according to common practice [15, 18, 35] and the specific splitting ratios are summarized in Table 5 and 6.

B MODEL DETAILS

In this section, we provide more details of the proposed PS2 methods from the neural architecture, hyper-parameter, and hardware perspectives.

[1]https://jmcauley.ucsd.edu/data/amazon/

B.1 Details of the Neural Architecture

Recall that our model consists of a personalized subgraph selector g_θ, the GNN encoder $_w$, and the link predictor q_w. The personalized subgraph selector is parameterized by a two-layer MLPs with hidden dimension D and output dimension 1. The GNN encoder is a K-layer GCN [17] modules, which varies from different downstream models. For example, when combing our PS2 with GAE and GraphSage, the default GNN module is GCN [17] and SAGE [11]. The link predictor is initialized as another three-layer MLPs. The hidden activation function in all neural networks is ReLU.

B.2 Hyperparameter Configuration

To provide fair comparison with state-of-the-art link prediction methods, we generally follow the same parameter settings across different baselines in terms of two different applications. In general, our model is optimized based on minibatch training. Following common practice for link prediction training, in each step, we sample a minibatch of positive edges from the training loader and then randomly generate one negative sample for each positive edge to construct the minibatch training set. Notice that, we don't conduct subgraph sampling for node representation as done in [11], we feed the whole adjacency matrix into the model for graph convolution.

Specifically, for planetoid datasets (Cora, CiteSeer, and PubMed), we adopt a three-layer GNN module with dimension 32. We set the batch size to 1024 and fix the learning rate to 0.01. For OGB datasts, we adopt a three GNN layer with hidden dimension 256. The learning rate and batch size are fixed as 0.001 and 10 * 1024 as suggested in [15] [2]. For recommendation datasets, we adopt a three GNN layer with hidden dimension 64 according to [13]. The batch size and learning rate are fixed as 1024 and 0.001, respectively. For different datasets, we search the hidden dimension D of subgraph selector MLP layer from the set $\{64, 128, 256, 512, 1024\}$. The best options for three Planetoid and other datasets are 256 and 512, respectively.

All the experiments are run 10 times, and we report the mean and the standard deviation.

B.3 Hardware

We conduct all the experiments on a server with 48 Intel(R) Xeon(R) Silver 4116 CPU @ 2.10GHz processors, 188 GB memory, and four NVIDIA GeForce RTX 3090 GPUs.

C GRADIENT APPROXIMATION FOR UPPER-LEVEL OPTIMIZATION

With w fixed, the upper-level optimization updates $\boldsymbol{\theta}$ according to the validation performance as:

$$\theta' = \theta - \lambda \nabla_\theta \mathcal{L}_{valid}(w^*(\theta), \theta). \tag{10}$$

[2]https://github.com/snap-stanford/ogb/tree/master/examples/linkproppred

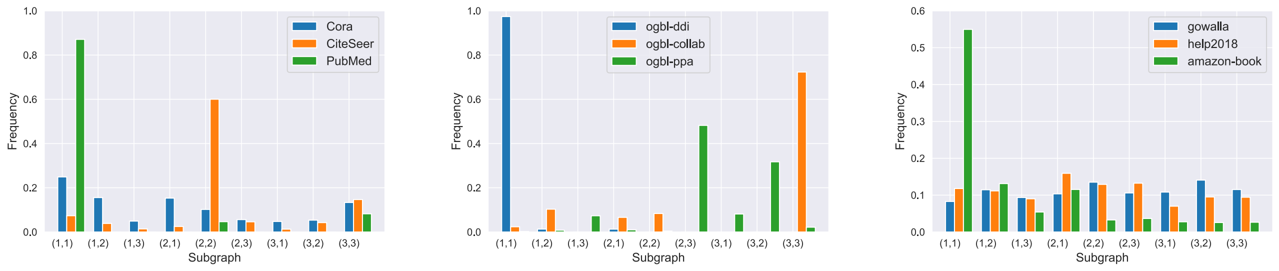

Figure 7: Subgraph distribution of GAE-PS2 on the Plantetoid (left), OGB (middle), and Recommendation (right) datasets.

Table 7: Finetune vs. train from scratch on Planetoid dataset.

	Cora	CiteSeer	PubMed
GAE	91.0 ± 0.01	89.5 ± 0.04	96.4 ± 0.01
GAE-PS2-scratch	90.1 ± 0.37	90.2 ± 0.9	97.7 ± 0.09
GAE-PS2	92.3 ± 0.71	92.2 ± 0.19	98.3 ± 0.10
GraphSage	86.3 ± 1.06	85.2 ± 2.56	87.6 ± 0.87
GraphSage-PS2-scratch	92.5 ± 0.89	88.9 ± 2.30	90.9 ± 0.44
GraphSage-PS2	93.8 ± 0.01	93.4 ± 1.20	93.5 ± 0.22

However, evaluating the gradient *w.r.t.* θ exactly is computationally prohibitive, since it requires solving for the optimal $w^*(\theta)$ whenever θ gets updated. To approximate the optimal solution $w^*(\theta)$, we propose to take one step of gradient descent update for w, without solving the lower-level optimization completely by training until convergence. Applying the chain rule, the approximated gradient yields:

$$\nabla_\theta \mathcal{L}_{valid}(w', \theta) - \lambda \nabla^2_{\theta,w} \mathcal{L}_{train}(w, \theta) \nabla_w \mathcal{L}_{valid}(w', \theta), \quad (11)$$

where $w' = w - \lambda \nabla_w \mathcal{L}_{train}(w, \theta)$ is the weights for one-step forward model. The second term in Eq. (11) contains an expensive matrix-vector product, which requires $O(|\theta||w|)$ complexity. To further accelerate the optimization, we approximate the second term using the finite difference approximation, defined as:

$$\nabla^2_{\theta,w} \mathcal{L}_{train}(w, \theta) \nabla_{w'} \mathcal{L}_{valid}(w', \theta) \approx \frac{\nabla_\theta \mathcal{L}_{train}(w^+, \theta) - \nabla_\theta \mathcal{L}_{train}(w^-, \theta)}{2\epsilon}, \quad (12)$$

Based on this approximation, we only need two forward passes for w and two backward passes for θ, therefore, the complexity is reduced from $O(|\theta||w|)$ to $O(|\theta| + |w|)$. The final result is

$$\nabla_\theta \mathcal{L}_{valid}(w^*(\theta), \theta) \approx \nabla_\theta \mathcal{L}_{valid}(w', \theta) - \lambda \frac{\nabla_\theta \mathcal{L}_{train}(w^+, \theta) - \nabla_\theta \mathcal{L}_{train}(w^-, \theta)}{2\epsilon}, \quad (13)$$

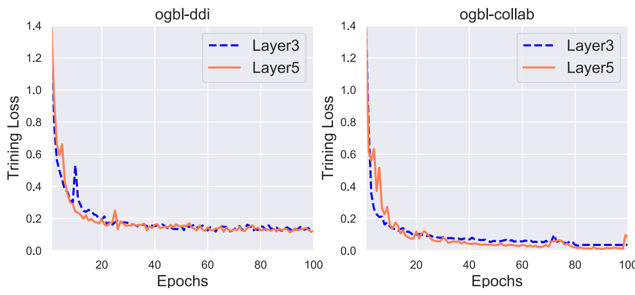

Figure 8: Empirical training curves of GAE-PS2 on datasets ogbL-ddi and ogbn-collab with different GCN [17] layers.